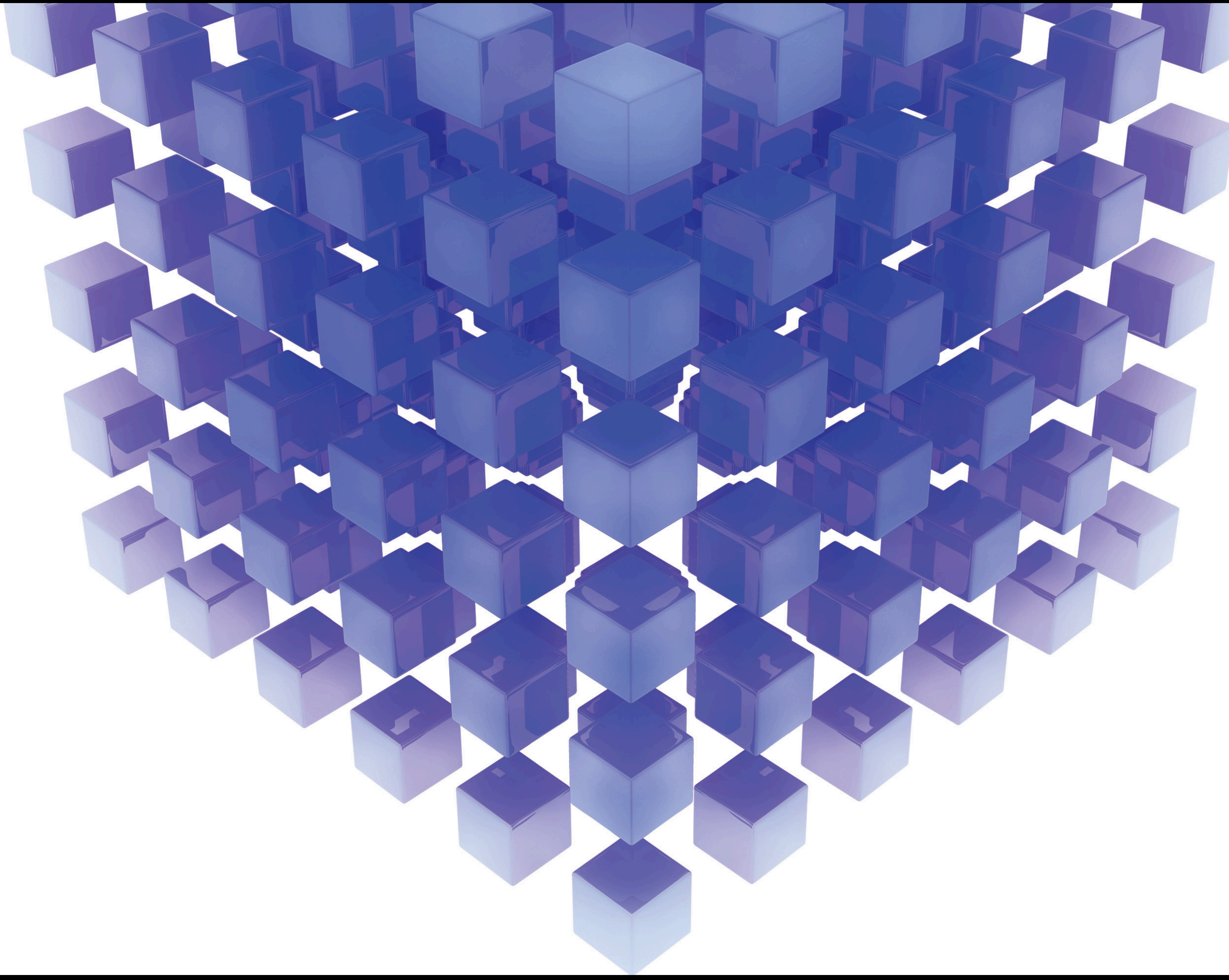
Mathematical Problems in Engineering

Structure, Dynamics, and Applications of Complex Networks in Software Engineering

Lead Guest Editor: Weifeng Pan
Guest Editors: Hua Ming and Chunlai Chai



Structure, Dynamics, and Applications of Complex Networks in Software Engineering

Mathematical Problems in Engineering

Structure, Dynamics, and Applications of Complex Networks in Software Engineering

Lead Guest Editor: Weifeng Pan
Guest Editors: Hua Ming and Chunlai Chai

Alberto Borboni, Italy
Marco Bortolini, Italy
Paolo Boscariol, Italy
Daniela Boso, Italy
Guillermo Botella-Juan, Spain
Boulaïd Boulkroune, Belgium
Abdesselem Boulkroune, Algeria
Fabio Bovenga, Italy
Francesco Braghin, Italy
Ricardo Branco, Portugal
Maurizio Brocchini, Italy
Julien Bruchon, France
Matteo Bruggi, Italy
Michele Brun, Italy
Maria Elena Bruni, Italy
Vasilis Burganos, Greece
Maria Angela Butturi, Italy
Raquel Caballero-Águila, Spain
Guillermo Cabrera-Guerrero, Chile
Filippo Cacace, Italy
Pierfrancesco Cacciola, United Kingdom
Salvatore Caddemi, Italy
zuowei cai, China
Roberto Caldelli, Italy
Alberto Campagnolo, Italy
Eric Campos, Mexico
Salvatore Cannella, Italy
Francesco Cannizzaro, Italy
Maosen Cao, China
Javier Cara, Spain
Raffaele Carli, Italy
Ana Carpio, Spain
Rodrigo Carvajal, Chile
Caterina Casavola, Italy
Sara Casciati, Italy
Federica Caselli, Italy
Carmen Castillo, Spain
Inmaculada T. Castro, Spain
Miguel Castro, Portugal
Giuseppe Catalanotti, United Kingdom
Nicola Caterino, Italy
Alberto Cavallo, Italy
Gabriele Cazzulani, Italy
Luis Cea, Spain
Fatih Vehbi Celebi, Turkey
Song Cen, China
Miguel Cerrolaza, Venezuela
M. Chadli, France
Gregory Chagnon, France
Ludovic Chamoin, France
Qing Chang, USA
Ching-Ter Chang, Taiwan
Kuei-Lun Chang, Taiwan
Xiaoheng Chang, China
Kacem Chehdi, France
Peter N. Cheimets, USA
Xinkai Chen, Japan
Shyi-Ming Chen, Taiwan
Zhiwen Chen, China
He Chen, China
Xiao Chen, China
Chih-Chiang Chen, Taiwan
Chien-Ming Chen, China
Kebing Chen, China
Xue-Bo Chen, China
Qiang Cheng, USA
Luca Chiapponi, Italy
Ryoichi Chiba, Japan
Francisco Chicano, Spain
Nicholas Chileshe, Australia
Tirivanhu Chinyoka, South Africa
Adrian Chmielewski, Poland
Seongim Choi, USA
Ioannis T. Christou, Greece
Hung-Yuan Chung, Taiwan
Simone Cinquemani, Italy
Roberto G. Citarella, Italy
Joaquim Ciurana, Spain
John D. Clayton, USA
Francesco Clementi, Italy
Piero Colajanni, Italy
Giuseppina Colicchio, Italy
Vassilios Constantoudis, Greece
Francesco Conte, Italy
Enrico Conte, Italy
Alessandro Contento, USA
Mario Cools, Belgium
Gino Cortellessa, Italy
Juan Carlos Cortés, Spain
Carlo Cosentino, Italy
Paolo Crippa, Italy
Erik Cuevas, Mexico
Guozeng Cui, China
Maria C. Cunha, Portugal

Reza Jazar, Australia
Khalide Jbilou, France
Isabel S. Jesus, Portugal
Linni Jian, China
Bin Jiang, China
Qing-Chao Jiang, China., China
Peng-fei Jiao, China
Emilio Jiménez Macías, Spain
Xiaoliang Jin, USA
Maolin Jin, Republic of Korea
Zhuo Jin, Australia
Dylan F. Jones, United Kingdom
Viacheslav Kalashnikov, Mexico
Mathiyalagan Kalidass, India
Tamas Kalmar-Nagy, Hungary
Zhao Kang, China
Tomasz Kapitaniak, Poland
Julius Kaplunov, United Kingdom
Konstantinos Karamanos, Belgium
Michal Kawulok, Poland
Irfan Kaymaz, Turkey
Vahid Kayvanfar, Iran
Krzysztof Kecik, Poland
Chaudry M. Khalique, South Africa
Mukhtaj khan, Pakistan
Abdul Qadeer Khan, Pakistan
Mostafa M. A. Khater, Egypt
Kwangki Kim, Republic of Korea
Nam-Il Kim, Republic of Korea
Philipp V. Kiryukhantsev-Korneev, Russia
P.V.V Kishore, India
Jan Koci, Czech Republic
Ioannis Kostavelis, Greece
Sotiris B. Kotsiantis, Greece
Frederic Kratz, France
Vamsi Krishna, India
Petr Krysl, USA
Edyta Kucharska, Poland
Krzysztof S. Kulpa, Poland
Kamal Kumar, India
Michal Kunicki, Poland
Cedrick A. K. Kwuimy, USA
Kyandoghere Kyamakya, Austria
Ivan Kyrchei, Ukraine
Davide La Torre, Italy
Márcio J. Lacerda, Brazil
Risto Lahdelma, Finland
Giovanni Lancioni, Italy
Jaroslaw Latalski, Poland
Antonino Laudani, Italy
Hervé Laurent, France
Aimé Lay-Ekuakille, Italy
Nicolas J. Leconte, France
Kun-Chou Lee, Taiwan
Dimitri Lefebvre, France
Eric Lefevre, France
Marek Lefik, Poland
Yaguo Lei, China
Gang Lei, Saudi Arabia
Kauko Leiviskä, Finland
Thibault Lemaire, France
afonso lemonge, Brazil
Ervin Lenzi, Brazil
Roman Lewandowski, Poland
ChenFeng Li, China
Jian Li, USA
Jun Li, China
Yueyang Li, China
Yang Li, China
Zhen Li, China
Yao-Jin Lin, China
Jian Lin, China
En-Qiang Lin, USA
Zhiyun Lin, China
Wanquan Liu, Australia
Jianxu Liu, Thailand
Yuanchang Liu, United Kingdom
Yu Liu, China
Heng Liu, China
Bo Liu, China
Bin Liu, China
Sixin Liu, China
Lei Liu, China
Bonifacio Llamazares, Spain
Alessandro Lo Schiavo, Italy
Jean Jacques Loiseau, France
Francesco Lolli, Italy
Paolo Lonetti, Italy
Sandro Longo, Italy
António M. Lopes, Portugal
Sebastian López, Spain
Pablo Lopez-Crespo, Spain
Cesar S. Lopez-Monsalvo, Mexico
Luis M. López-Ochoa, Spain

Ezequiel López-Rubio, Spain
Vassilios C. Loukopoulos, Greece
Jose A. Lozano-Galant, Spain
Gabriele Maria Lozito, Italy
Rongxing Lu, Canada
Songtao Lu, USA
haiyan Lu, Australia
Zhiguo Luo, China
Gabriel Luque, Spain
Valentin Lychagin, Norway
Dazhong Ma, China
Junhai Ma, China
Antonio Madeo, Italy
Alessandro Magnani, Belgium
Tahir Mahmood, Pakistan
Toqeer Mahmood, Pakistan
Fazal M. Mahomed, South Africa
Arunava Majumder, India
Paolo Manfredi, Italy
Adnan Maqsood, Pakistan
Giuseppe Carlo Marano, Italy
Damijan Markovic, France
Filipe J. Marques, Portugal
Luca Martinelli, Italy
Rodrigo Martinez-Bejar, Spain
Guiomar Martín-Herrán, Spain
Denizar Cruz Martins, Brazil
Francisco J. Martos, Spain
Elio Masciari, Italy
Franck Massa, France
Paolo Massioni, France
Alessandro Mauro, Italy
Jonathan Mayo-Maldonado, Mexico
Fabio Mazza, Italy
Pier Luigi Mazzeo, Italy
Laura Mazzola, Italy
Driss Mehdi, France
Zahid Mehmood, Pakistan
YUE MEI, China
Roderick Melnik, Canada
Xiangyu Meng, USA
Debiao Meng, China
Jose Merodio, Spain
Alessio Merola, Italy
Mahmoud Mesbah, Iran
Luciano Mescia, Italy
Laurent Mevel, France
Constantine Michailides, Cyprus
Mariusz Michta, Poland
Prankul Middha, Norway
Aki Mikkola, Finland
Giovanni Minafò, Italy
Hiroyuki Mino, Japan
Dimitrios Mitsotakis, New Zealand
Saleh Mobayen, Iran
Nikunja Mohan Modak, India
Sara Montagna, Italy
Roberto Montanini, Italy
Francisco J. Montáns, Spain
Gisele Mophou, France
Rafael Morales, Spain
Marco Morandini, Italy
Javier Moreno-Valenzuela, Mexico
Simone Morganti, Italy
Caroline Mota, Brazil
Aziz Moukrim, France
Shen Mouquan, China
Dimitris Mourtzis, Greece
Emiliano Mucchi, Italy
Taseer Muhammad, Saudi Arabia
Josefa Mula, Spain
Jose J. Muñoz, Spain
Giuseppe Muscolino, Italy
Dino Musmarra, Italy
Marco Mussetta, Italy
Ghulam Mustafa, Pakistan
Hariharan Muthusamy, India
Hakim Naceur, France
Alessandro Naddeo, Italy
Benedek Nagy, Turkey
Omar Naifar, Tunisia
Mariko Nakano-Miyatake, Mexico
Keivan Navaie, United Kingdom
Adrian Neagu, USA
Erivelton Geraldo Nepomuceno, Brazil
AMA Neves, Portugal
Luís C. Neves, United Kingdom
Dong Ngoduy, New Zealand
Nhon Nguyen-Thanh, Singapore
Papakostas Nikolaos, Ireland
Jelena Nikolic, Serbia
Kottakkaran Sooppy Nisar, Saudi Arabia
Tatsushi Nishi, Japan
Shanzhou Niu, USA

Xesús Nogueira, Spain
Ben T. Nohara, Japan
Mohammed Nouari, France
Mustapha Nourelfath, Canada
Kazem Nouri, Iran
Ciro Núñez-Gutiérrez, Mexico
Wlodzimierz Ogryczak, Poland
Roger Ohayon, France
Krzysztof Okarma, Poland
Mitsuhiro Okayasu, Japan
Diego Oliva, Mexico
Alberto Olivares, Spain
Enrique Onieva, Spain
Calogero Orlando, Italy
Sergio Ortobelli, Italy
Naohisa Otsuka, Japan
Taoreed Owolabi, Nigeria
Pawel Packo, Poland
Arturo Pagano, Italy
Roberto Palma, Spain
Alessandro Palmeri, United Kingdom
Pasquale Palumbo, Italy
Li Pan, China
Weifeng Pan, China
Chandan Pandey, India
K. M. Pandey, India
Jürgen Pannek, Germany
Elena Panteley, France
Achille Paolone, Italy
George A. Papakostas, Greece
Xosé M. Pardo, Spain
You-Jin Park, Taiwan
Manuel Pastor, Spain
Petr Páta, Czech Republic
Pubudu N. Pathirana, Australia
Surajit Kumar Paul, India
Sitek Paweł, Poland
Luis Payá, Spain
Alexander Paz, Australia
Igor Pažanin, Croatia
Libor Pekař, Czech Republic
Francesco Pellicano, Italy
Marcello Pellicciari, Italy
Zhi-ke Peng, China
Mingshu Peng, China
Haipeng Peng, China
Zhengbiao Peng, Australia
Jian Peng, China
Xindong Peng, China
Yuexing Peng, China
Bo Peng, China
Marzio Pennisi, Italy
Maria Patrizia Pera, Italy
Matjaz Perc, Slovenia
A. M. Bastos Pereira, Portugal
Ricardo Perera, Spain
F. Javier Pérez-Pinal, Mexico
Michele Perrella, Italy
Francesco Pesavento, Italy
Ivo Petras, Slovakia
Francesco Petrini, Italy
Hoang Vu Phan, Republic of Korea
Lukasz Pieczonka, Poland
Dario Piga, Switzerland
Antonina Pirrotta, Italy
Marco Pizzarelli, Italy
Javier Plaza, Spain
Goutam Pohit, India
Kemal Polat, Turkey
Dragan Poljak, Croatia
Jorge Pomares, Spain
Hiram Ponce, Mexico
Sébastien Poncet, Canada
Volodymyr Ponomaryov, Mexico
Jean-Christophe Ponsart, France
Mauro Pontani, Italy
Cornelio Posadas-Castillo, Mexico
Francesc Pozo, Spain
Aditya Rio Prabowo, Indonesia
Anchasa Pramuanjaroenkij, Thailand
Christopher Pretty, New Zealand
Leonardo Primavera, Italy
Luca Pugi, Italy
Krzysztof Puszynski, Poland
Goran D. Putnik, Portugal
Chuan Qin, China
Jianlong Qiu, China
Giuseppe Quaranta, Italy
Vitomir Racic, Italy
Ahmed G. Radwan, Egypt
Dr. Hamid Rahman, Pakistan
Carlo Rainieri, Italy
Kumbakonam Ramamani Rajagopal, USA
Venkatesan Rajinikanth, India

Ali Ramazani, USA
Higinio Ramos, Spain
Angel Manuel Ramos, Spain
Muhammad Afzal Rana, Pakistan
Amer Rasheed, Pakistan
Muhammad Rashid, Saudi Arabia
Manoj Rastogi, India
Alessandro Rasulo, Italy
S.S. Ravindran, USA
Abdolrahman Razani, Iran
Alessandro Reali, Italy
Oscar Reinoso, Spain
Jose A. Reinoso, Spain
X. W. Ren, China
Haijun Ren, China
Carlo Renno, Italy
Fabrizio Renno, Italy
Shahram Rezapour, Iran
Ricardo Riaza, Spain
Francesco Riganti-Fulginei, Italy
Gerasimos Rigatos, Greece
Francesco Ripamonti, Italy
Marcelo Raúl Risk, Argentina
Jorge Rivera, Mexico
Eugenio Roanes-Lozano, Spain
Bruno G. M. Robert, France
Ana Maria A. C. Rocha, Portugal
Luigi Rodino, Italy
Francisco Rodríguez, Spain
Rosana Rodríguez López, Spain
Alessandra Romolo, Italy
Abdolreza Roshani, Italy
Francisco Rossomando, Argentina
Jose de Jesus Rubio, Mexico
Weiguo Rui, China
Rubén Ruiz, Spain
Ivan D. Rukhlenko, Australia
Chaman Lal Sabharwal, USA
Kishin Sadarangani, Spain
Andrés Sáez, Spain
Bekir Sahin, Turkey
Michael Sakellariou, Greece
John S. Sakellariou, Greece
Salvatore Salamone, USA
Jose Vicente Salcedo, Spain
Alejandro Salcido, Mexico
Alejandro Salcido, Mexico
Salman saleem, Pakistan
Ahmed Salem, Saudi Arabia
Nunzio Salerno, Italy
Rohit Salgotra, India
Miguel A. Salido, Spain
Zabidin Salleh, Malaysia
Roque J. Saltarén, Spain
Alessandro Salvini, Italy
Abdus Samad, India
Nikolaos Samaras, Greece
Sylwester Samborski, Poland
Ramon Sancibrian, Spain
Giuseppe Sanfilippo, Italy
Omar-Jacobo Santos, Mexico
J Santos-Reyes, Mexico
José A. Sanz-Herrera, Spain
Nickolas S. Sapidis, Greece
Evangelos J. Sapountzakis, Greece
Musavarah Sarwar, Pakistan
Marcelo A. Savi, Brazil
Andrey V. Savkin, Australia
Tadeusz Sawik, Poland
Roberta Sburlati, Italy
Gustavo Scaglia, Argentina
Thomas Schuster, Germany
Oliver Schütze, Mexico
Lotfi Senhadji, France
Junwon Seo, USA
Michele Serpilli, Italy
Joan Serra-Sagrista, Spain
Silvestar Šesnić, Croatia
Erhan Set, Turkey
Gerardo Severino, Italy
Ruben Sevilla, United Kingdom
Stefano Sfarra, Italy
Mohamed Shaat, United Arab Emirates
Mostafa S. Shadloo, France
Kamal Shah, Pakistan
Leonid Shaikhet, Israel
Xingling Shao, China
hang shen, China
Hao Shen, China
Xin Pu Shen, China
Bo Shen, Germany
Dimitri O. Shepelsky, Ukraine
Jian Shi, China
Weichao SHI, United Kingdom

Fu-Kwun Wang, Taiwan
J.G. Wang, China
Zhibo Wang, China
Zenghui Wang, South Africa
Guoqiang Wang, China
Yung-Chung Wang, Taiwan
C. H. Wang, Taiwan
Hui Wang, China
Shuo Wang, China
Xinyu Wang, China
Qingling Wang, China
Yong Wang, China
Dagang Wang, China
Weiwei Wang, China
Zhenbo Wang, USA
Ji Wang, China
Bingchang Wang, China
Roman Wan-Wendner, Austria
P.H. Wen, United Kingdom
Fangqing Wen, China
Waldemar T. Wójcik, Poland
Wai. L. Woo, Singapore
Qiuhong Wu, China
Zhizheng Wu, China
Zhibin Wu, China
Xianyi Wu, China
Changzhi Wu, China
Yuqiang Wu, China
Michalis Xenos, Greece
hao xiao, China
Xiao Ping Xie, China
Xue-Jun Xie, China
Gen Q. Xu, China
Hang Xu, China
Zeshui Xu, China
Qingzheng Xu, China
Lei Xu, China
Qilong Xue, China
Joseph J. Yame, France
Xinggang Yan, United Kingdom
Chuanliang Yan, China
Zhiguo Yan, China
Jixiang Yang, China
Weilin Yang, China
Ray-Yeng Yang, Taiwan
Mijia Yang, USA
Min Ye, China
Jun Ye, China
Luis J. Yebra, Spain
Peng-Yeng Yin, Taiwan
Muhammad Haroon Yousaf, Pakistan
Yuan Yuan, United Kingdom
Qin Yuming, China
Abdullahi Yusuf, Nigeria
Akbar Zada, Pakistan
Elena Zaitseva, Slovakia
Arkadiusz Zak, Poland
Daniel Zaldivar, Mexico
Ernesto Zambrano-Serrano, Mexico
Francesco Zammori, Italy
Vittorio Zampoli, Italy
Rafal Zdunek, Poland
Ahmad Zeeshan, Pakistan
Ibrahim Zeid, USA
Nianyin Zeng, China
Bo Zeng, China
Junyong Zhai, China
Lingfan Zhang, China
Tianwei Zhang, China
Yong Zhang, China
Wenyu Zhang, China
Haopeng Zhang, USA
Hao Zhang, China
Jian Zhang, China
Qian Zhang, China
Xiaofei Zhang, China
Tongqian Zhang, China
Xuping Zhang, Denmark
Kai Zhang, China
Xianming Zhang, Australia
Yifan Zhao, United Kingdom
Yongmin Zhong, Australia
Zebo Zhou, China
Debao Zhou, USA
Jian G. Zhou, United Kingdom
Quanxin Zhu, China
Wu-Le Zhu, China
Gaetano Zizzo, Italy
Zhixiang Zou, China

Contents

Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 6734248, 2 pages
https://doi.org/10.1155/2021/6734248

Editorial

Structure, Dynamics, and Applications of Complex Networks in Software Engineering

Weifeng Pan,[1] Hua Ming,[2] and Chunlai Chai[1]

[1]*School of Computer Science and Information Engineering, Zhejiang Gongshang University, Hangzhou, Zhejiang 310018, China*
[2]*School of Engineering and Computer Science, Oakland University, Rochester, MI 48309, USA*

Correspondence should be addressed to Weifeng Pan; wfpan@zjgsu.edu.cn

Received 14 September 2020; Accepted 14 September 2020; Published 16 February 2021

Complex network analysis has been proved to be an effective tool to quantify the structural properties of different complex systems. Large-scale software projects are interesting examples of human-made complex systems, which can be analyzed using theories and tools in the field of complex networks [1, 2]. Generally, the complexity of these systems can be reflected both in their structure and in their development processes. Due to the wide adoption of open-source practices using online infrastructures, we can trace the software development process and the final software structure in an easy way. Thus, a large-scale dataset about software projects can be obtained, making an in-depth study of software projects possible. During the last decade, complex networks have been widely applied to analyze the topological structure and dynamics of software projects. Many shared physics-like laws of software projects have been revealed, such as *scale-free*, *small-world*, and *fractal properties*.

The objective of this special issue (SI) is to provide a comprehensive and latest collection of research works on the application of complex network theory and techniques to explore software projects. This SI receives 34 submissions in total, and after a fair and rigorous peer-review process, 13 of them are published, with the acceptance rate being roughly 38.2%. The 13 papers can be roughly categorized into three groups according to the topics that they focus on, i.e., object-oriented software systems, service-oriented software systems, and others.

1. Object-Oriented Software Systems

Five papers focus on the research topics in traditional object-oriented software systems, i.e., software metrics, bug report classification, software defect prediction, and bug triage. Li et al. [3] reviewed the interdisciplinary research work between the fields of complex networks and software engineering. These papers are published in the last seven years (2013 to 2019) and mainly focus on three different research directions, i.e., modeling, analysis, and applications of software networks. Gu et al. [4] analyzed the coupling between classes at different levels and used a set of bipartite software networks to represent them. Finally, they proposed metrics to characterize the coupling between classes. Guo et al. [5] proposed a novel approach to solve the bug report classification problem, which combines several imbalanced learning strategies and multiclass classification methods together. Shi et al. [6] proposed a novel software defect prediction model, which leverages a convolutional neural network to learn semantic features from the source code and applies network embedding to learn structural features from software networks at the class level. Ge et al. [7] proposed an improved bug triage approach for newly reported bugs, which removes the low-quality bug reports and considers the influence of the engagement of developers on their final ranking.

2. Service-Oriented Software Systems

Five papers focus on the research topics in service-oriented software systems, i.e., service clustering, service recommendation, service discovery, service selection, and service quality measurement. Zhou and Wang [8] proposed an approach to organize API services into different clusters. Their approach applies structural metrics built from service networks where APIs and Mashups are nodes, and their couplings are edges. Xiong et al. [9] applied NLP and graph embedding techniques to recommend APIs for Mashup developers. They extracted structural semantics from a two-mode graph of Mashups, APIs, and their relations. Sun et al. [10] proposed an improved web service discovery approach, which integrates labels of web services using a neural topic model as external semantics for these web services. Jiang et al. [11] proposed a novel API selection approach for Mashup development. Their approach extracted similarities from the profile of APIs and Mashups. Yang and Wang [12] proposed a hierarchical aggregation model to accurately aggregate the ratings of services.

3. Others

This SI also contains three papers which are not related to the topic of this SI, i.e., [13, 14], and [15]. These papers are handled by the editors from the editorial board of *Mathematical Problems in Engineering*.

Conflicts of Interest

The editors declare that there are no conflicts of interest regarding the publication of this SI.

Acknowledgments

We would like to thank all authors for their contributions to the SI and acknowledge the reviewers for their time and effort devoted to the peer-review process of the manuscripts. We highly appreciate the support from Rhea Lumbres in the editorial office of Hindawi Publishing Corporation.

Weifeng Pan
Hua Ming
Chunlai Chai

References

[1] W. Pan, B. Li, J. Liu, Y. Ma, and B. Hu, "Analyzing the structure of Java software systems by weightedK-core decomposition," *Future Generation Computer Systems*, vol. 83, pp. 431–444, 2018.

[2] W. Pan, H. Ming, C. Chang, Z. Yang, and D.-K. Kim, "ElementRank: ranking java software classes and packages using a multilayer complex network-based approach," *IEEE Transactions on Software Engineering*, p. 1, 2019.

[3] H. Li, T. Wang, X. X. Xu, B. Jiang, J. L. Wei, and J. L. Wang, "Modeling software systems as complex networks: analysis and their applications," *Mathematical Problems in Engineering*, vol. 2020, Article ID 5346498, 7 pages, 2020.

[4] A. H. Gu, L. Li, S. Li, Q. Xun, J. Dong, and J. Lin, "Method of coupling metrics for object-oriented software system based on CSBG approach," *Mathematical Problems in Engineering*, vol. 2020, Article ID 3428604, 20 pages, 2020.

[5] S. K. Guo, S. W. Wang, M. M. Wei, R. Chen, C. Guo, and H. Li, "Combining im-balance learning strategy and multiclassifier estimator for bug report classification," *Mathematical Problems in Engineering*, vol. 2020, Article ID 5712461, 2020.

[6] M. L. Shi, P. He, H. T. Xiao, H. X. Li, and C. Zeng, "An approach to semantic and structural features learning for software defect prediction," *Mathematical Problems in Engineering*, vol. 2020, Article ID 6038619, 13 pages, 2020.

[7] X. Ge, S. J. Zheng, J. H. Wang, and H. Li, "High-dimensional hybrid data reduction for effective bug triage," *Mathematical Problems in Engineering*, vol. 2020, Article ID 5102897, 20 pages, 2020.

[8] S. Y. Zhou and Y. L. Wang, "Clustering services based on community detection in service networks," *Mathematical Problems in Engineering*, vol. 2019, Article ID 1495676, 2019.

[9] W. Xiong, Z. Wu, B. Li, and B. Hang, "Automating Mashup service recommendation via semantic and structural features," *Mathematical Problems in Engineering*, vol. 2020, Article ID 4960439, 10 pages, 2020.

[10] C. Sun, L. Lv, G. Tian, Q. Wang, X. Zhang, and L. Guo, "Leverage label and word embedding for semantic sparse Web service discovery," *Mathematical Problems in En-Gineering*, vol. 2020, Article ID 5670215, 8 pages, 2020.

[11] B. Jiang, P. X. Liu, Y. Wang, and Y. Z. Chen, "HyOASAM: A hybrid open API selection approach for mashup development," *Mathematical Problems in Engineering*, vol. 2020, Article ID 4984375, 16 pages, 2020.

[12] R. Yang and D. H. Wang, "Hierarchical aggregation for reputation feedback of services networks," *Mathematical Problems in Engineering*, vol. 2020, Article ID 3748383, 12 pages, 2020.

[13] D. Chen, X. Wu, S. Xie et al., "Study on the thin plate model with elastic foundation boundary of overlying strata for backfill mining," *Mathematical Problems in Engineering*, vol. 2020, Article ID 8906091, 15 pages, 2020.

[14] Y. Deng, K. Yao, T. Jin, Z. Feng, and X. Liu, "PTS-FNN-based health prediction method for flexible photoelectric film material processing equipment," *Mathematical Problems in Engineering*, vol. 2020, Article ID 9232561, 10 pages, 2020.

[15] T. T. Tran, Q.-H. Pham, and T. Nguyen-Thoi, "An edge-based smoothed finite element for free vibration analysis of functionally graded porous (FGP) plates on elastic foundation taking into mass (EFTIM)," *Mathematical Problems in Engineering*, vol. 2020, Article ID 8278743, 17 pages, 2020.

Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 4960439, 10 pages
https://doi.org/10.1155/2020/4960439

Research Article

Automating Mashup Service Recommendation via Semantic and Structural Features

Wei Xiong,[1,2] Zhao Wu,[1] Bing Li,[2] and Bo Hang[1]

[1]*Hubei University of Arts and Science, Xiangyang 441000, China*
[2]*International School of Software, Wuhan University, Wuhan 430072, China*

Correspondence should be addressed to Zhao Wu; wuzhao73@163.com

Received 4 February 2020; Revised 9 June 2020; Accepted 6 July 2020; Published 12 August 2020

Guest Editor: Chunlai Chai

Increasing physical objects connected to the Internet make it possible for smart things to access all kinds of cloud services. Mashup has been an effective way to the rapid IoT (Internet of Things) application development. It remains a big challenge to bridge the semantic gap between user expectations and application functionality with the development of mashup services. This paper proposes a mashup service recommendation approach via merging semantic features from API descriptions and structural features from the mashup-API network. To validate our approach, large-scale experiments are conducted based on a real-world accessible service repository, ProgrammableWeb. The results show the effectiveness of our proposed approach.

1. Introduction

Internet of Things (IoT) was firstly introduced to the community in 1999 for supply chain management. Now, we will arrive to the post-cloud era, where there will be large amounts of smart things to access all kinds of cloud services, and the capabilities of smart things can be enhanced by interacting with other functional entities through the interfaces of cloud services. Since the functionality of an individual service is too simple to satisfy the complex requirements of users, in an IoT application, people prefer to combine several cloud services together. In recent years, mashup technology has gained great attention since it can support the development of IoT applications by composing existing cloud services in the form of web APIs.

Several platforms, including Seekda1, Google, and ProgrammableWeb [1], enabled vibrant software ecosystems between service providers and developers, where developers utilize one or more query items such as keywords and category. However, keyword-based service recommendation mechanisms usually have low accuracy [2]. NLP techniques are increasingly applied in the software engineering domain, which have been shown to be useful in requirements engineering [3], usability of API documents [4], and other areas [5]. Thus, semantic queries may get more accurate results.

Most of API descriptions and the tags are all text. Then, these approaches use similar semantic metrics that capture the service similarity, such as the similarity of service descriptions and tags. Indeed, most of similarity metrics, which are proposed to quantify the similarity of service descriptions and the similarity of tags, are based on the semantic information in the text. Some works utilized structural metrics to quantify the structural similarity where the metrics reached the topological information extracted from methods, attributes, classes and their couplings, etc [6–8]. However, there are very few works in which structural similarity has been introduced to guide the service ranking.

We present AMSRSSF (automating mashup service recommendation via semantic and structural features), a framework that utilizes NLP and graph-embedding techniques to recommend services for developers in this study. AMSRSSF takes as input the developers' personalized requirements and structural semantic and determines which services can be recommended for developers. Furthermore, we show that the structural semantics can be generated from a two-mode graph, which can describe mashups, web APIs, and their relations. We evaluate AMSRSSF against a dataset of description documents including 10050 web-based services and 7155 mashups. Our experiments demonstrate that our approach can rank mashup services efficiently, and its performance is better than semantic-based mashup service ranking approaches alone.

This paper makes the following main contributions:

(1) We propose a mashup service recommendation approach via merging semantic features from API descriptions and structural features from the mashup-API network

(2) We conduct comprehensive experiments on a real-world dataset, demonstrating the effectiveness of our approach

The remainder of this paper is organized as follows: Section 2 gives a motivating scenario and presents our approach. Section 3 describes the experiments in detail. Section 4 discusses the related works, and Section 5 concludes the paper with future work directions.

2. Automating Service Recommendation via Semantic Features and Structural Features

In this section, we first present a scenario to illustrate the motivation of our work in Section 2.1. Then, we discuss the issues of mashup service recommendation in Section 2.2; next, we propose a mashup service recommendation approach via semantic and structural features in Sections 2.3–2.8.

2.1. Motivation. Under Internet scenario, more structures are presented between data objects. A typical scenario is a knowledge graph that consists of a global relationship diagram of user behavior data and an item with more attributes. Structural and semantic similarities characterize different aspects of the service similarity. Indeed, two similarities are orthogonal, which motivate our work. We expected that better service recommendation approaches might be proposed by using structural features. In particular, we propose a simple but effective approach AMSRSSF to rank mashup services by using semantic and structural features. First, it applies a two-mode graph to describe mashups, web APIs, and their relations formally. Second, we quantify the structural similarity between every pair of mashup services based on the two-mode graph. The structural similarity results from the structural context in which web APIs are used by mashup services. Finally, we introduce a merging embedding vectors algorithm that only considers the pairwise similarities between mashup services to rank them effectively.

According to the above, it is certainly valuable to introduce structural features and then improve the accuracy of recommendation based on structural semantics.

Finally, there are some other issues which need to be addressed:

(1) How to extract semantic from natural language API descriptions?

(2) How to generate structural semantics of the mashup-mashup network?

(3) How to merge multiple embedding vectors for better accuracy?

(4) How do we design experiments for performance evaluation?

2.2. Problem Description. Our work considers the task of recommendation via representation learning as follows: given a requirement q described in multidimensional information, which includes semantic and structural features, the corresponding recommended services are given via similarity calculation, which exists in terms of the representation learning model.

Our model consists in learning a function $S(\cdot)$, which can score requirement-service pairs (q, t). Hence, finding the top-ranked answer $t(q)$ to a requirement q is directly carried out by

$$\hat{t}(q) = \arg\max_{t' \in K} S(q, t'), \tag{1}$$

and to handle multiple recommended services, we present the results as a ranked list rather than taking the top service.

2.3. Overview of Our Framework. In this section, we will present an overview of our approach. As shown in Figure 1, our approach mainly consists of four components: semantic extraction based on NLP pipeline, structural embedding generation of the mashup-mashup network, merging of multiple embeddings, and recommendation based on fusion embedding. More specifically, the descriptions of web-based services will be crawled from ProgrammableWeb.

We firstly preprocess natural language API descriptions using NLP pipeline and extract text semantic embedding. We then generate structural embedding of the mashup-mashup network; next, we merge semantic embedding and structural embedding. Finally, the developers give multi-dimensional information of requirements, which will be parsed into semantic and structural features and passed to similarity calculation for ranking (Figure 2).

2.4. Semantic Extraction Based on the NLP Pipeline. This section preprocesses the sentences in API descriptions using NLP pipeline named as NLTK [9], which resembles the following high-level flow in Figure 3.

For purposes of illustration, we introduce a sample as follows:

> "The Twitter microblogging service includes two RESTful APIs. The Twitter REST API methods allow developers to access core Twitter data."

We have the following steps.

2.4.1. EOS (End of Sentence) Detection. Breaking the texts into paragraphs might be important. However, it may be unlikely to help EOS detection, which marks the boundary of a sentence from texts and breaks them into sentences. We

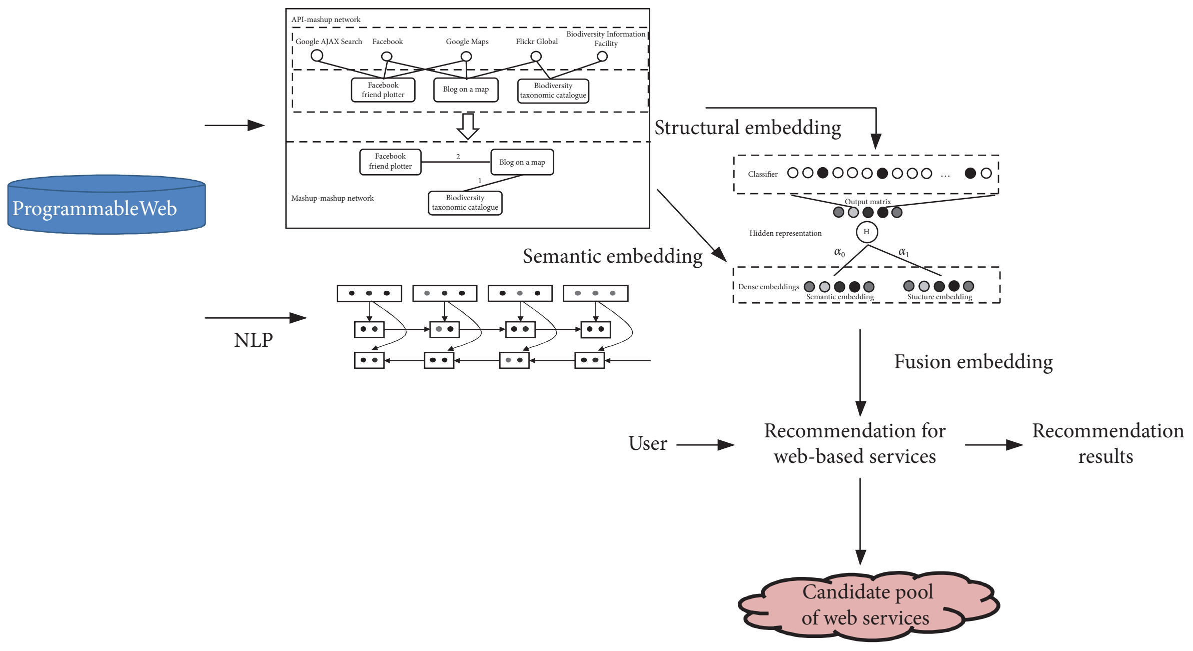


Figure 1: Overview of our approach.

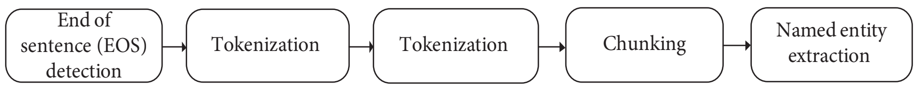


Figure 2: Overview of the NLP pipeline.

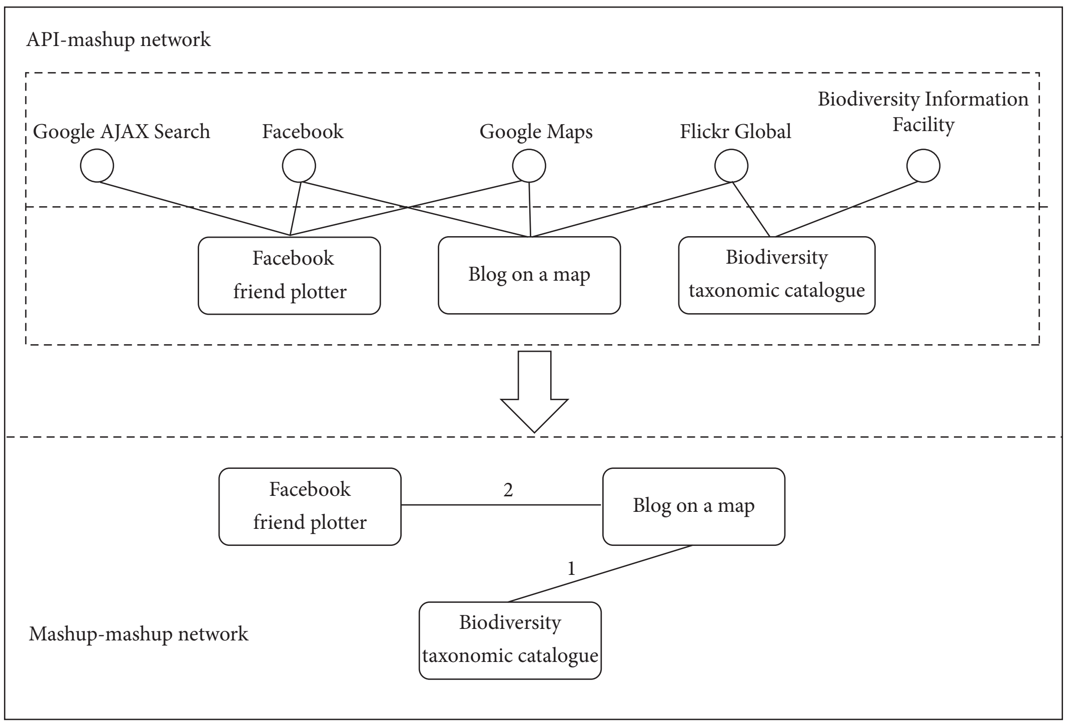


Figure 3: Mashup-API network and mashup-mashup network.

utilize them to represent logical units of thought and tend to give a predictable syntax for further analysis.

We will obtain the following results by parsing a sentence with NLTK:

> [“The Twitter micro-blogging service includes two RESTful APIs. The Twitter REST API methods allow developers to access core Twitter data.”]

A period “.” is utilized to mark the end of a sentence.

However, there are other legal usages of the period such as decimal (periods between numbers), ellipsis “...,” and shorthand notations just like “Mr.” and “Dr.”. We can overcome this difficulty by looking up abbreviations from WordNet and detecting decimals including period character by regular expressions.

Moreover, there are instances where an enumeration list is used to describe data type, such as the following:

“This service where you can obtain the following data: (a) *abc* . . ., (b) *xyz* . . ..”

It is prone to understand the implication for a human, but it is arduous to seek appropriate boundaries as far as a machine is concerned. An improvement over breaking on characters “.” is leveraging the Syntax information:

(1) Placements of tabs

(2) Bullet points (numbers, characters, roman numerals, and symbols)

(3) Delimiters such as “:” to detect appropriate boundaries

We further improve the EOS detection using the following patterns:

(1) We remove the leading and trailing “ $*$ ” and “-” characters in a sentence.

(2) We consider the following characters as sentence separators: “–”, “-”, “∲”, “§”, “ ¥ ”, “◇”, “}”, “|”, “~”, and “★” . . ..

(3) For an enumeration sentence, we split the sentence into short ones for each enumerated item.

Currently, sentence detection has occurred for arbitrary text.

2.4.2. Tokenization. We then split individual sentences into tokens, leading to the following results:

> [[“The”, “Twitter”, “microblogging”, “service”, “includes”, “two”, “RESTful”, “APIs”, “.”], [“The”, “Twitter”, “REST”, “API”, “methods”, “allow”, “developers”, “to”, “access”, “core”, “Twitter”, “data”,“.”]]

Here, tokenization appeared to split on whitespace, and it tokenized out EOS correctly as well. We further improve the discrimination by determining whether a period is the end of sentence marker or part of an abbreviation.

2.4.3. POS (Parts of Speech) Tagging. Part of speech of each token in a sentence means “word tagging” or “grammatical tagging,” where the state-of-the-art approaches have been shown to achieve 97% accuracy in classifying POS tags for well-written news articles. We can obtain the following results:

> [[(“The”, “DT”), (“Twitter”, “NNP”), (“micro-blogging”, “NNP”), (“service”, “NNP”), (“includes”, “VBZ”), (“two”, “NUM”), (“RESTful”,“’JJ”), (“APIs”,“’NNP”)], [(“The”, “DT”), (“Twitter”, “NNP”), (“REST”, “NNP”), (“API”, “NNP”), (“methods”, “NNP”), (“allow”, “VB”), (“developers”, “NNP”), (“to”, ’IN”), (“access”, “VB”), (“core”, “JJ”), (“Twitter”, “NNP”), (“data”, “NNP”), (“.”, “.”)]]

“NNP” manifests a noun as the part of a noun phrase, “VBD” manifests a verb in simple past tense, “DT” manifests an article, “IN” manifests a preposition, “NUM” manifests a numeral, and “JJ” manifests an adjective.

After finishing POS tagging, we can chunk nouns as part of noun phrases and then try to deduce the types of entities (e.g., people, places, organizations, etc.).

2.4.4. Chunking. Chunking means to analyze tagged tokens from sentence and identify logical concepts, such as “Pandora Internet radio” and “Google Maps.” We annotate such phrases as single lexical units such as a noun phrase and a verb phrase. The state-of-the-art approaches achieve around 97% accuracy in classifying phrases and clauses over well-written news articles[4]. We develop chunking with more complex grammars to substitute default one in NLTK.

2.4.5. Named Entity Extraction. Named Entity Extraction means analyzing chunks and further tagging the chunks as named entities, such as people, organizations, and objects. We will obtain the following results:

> [Tree(“S”, [Tree(“OBJ”, [(“The”, “DT”), (“Twitter”, “NNP”), (“microblogging”, “NNP”), (“service”, “NNP”)]), (“includes”, “VBZ”), Tree(“OBJ”, [(“two”, “NUM”), (“RESTful”, “JJ”), (“APIs”, “NNP”), (“.”, “.”)]), Tree(“S”, [Tree(“OBJ”, [(“The”, “DT”), (“Twitter”, “NNP”), (“REST”” “NNP”), (“PI”, “NNP”), (“methods”, “NNP”)]), (“allow”, “VB”), Tree(“-PERSON”, [(“developers”, “NNP”)]), (“to”, “IN”), (“access”, “VB”), (“core”, “JJ”), (“Twitter”, “NNP”), (“data”, “NNP”), (“.”, “.”)]

We can identify that “The Twitter microblogging service,” “two RESTful APIs,” and “The Twitter REST API methods” have been tagged as an object and “developers” has been tagged as a person.

In addition, abbreviations usually lead to incorrect parsing for sentence; for example, “Application Programming Interfaces (APIs)” is treated as single lexical unit. We can work it out looking up abbreviations from WordNet [10].

2.4.6. Semantic Embedding Generation. In order to convert named entity into feature vectors and utilize them in latter models, we utilize word2vector to encode named entities from API descriptions.

Word2vector is a powerful method to extract meaningful grammatical and semantic features, which can transform a word into a fixed-length vector. Each named entity from API description can be reduced to embedding vector V_{text} , and semantic embedding of API description can then be represented as a fixed-length vector as follows:

$$V_m = \text{average}\left(V_{\text{text}i}\right), \quad I = \{1, 2, \ldots, N_m\}, \tag{2}$$

where N_m is the number of named entities of API description.

2.5. Two-Mode Graph of Mashup Services and Web APIs. The first step of structural embedding generation is to generate the mashup-mashup network. The network can be generated through mashup behavior sequence. The information, such as the same APIs between mashups, can be utilized to establish the edges between mashups as well, so as to generate structural graph. The embedding vectors based on this graph can be called structural embedding vectors.

In this article, two networks are utilized to represent all relationships of mashup services and APIs: one is the network between API and mashup services, and the other is the network between mashup services.

Definition 1. MAN (mashup-API network) is a network of 2 modes, where the nodes represent the API and the mashup services, and edges represent the relationship between them.

Definition 2. MMN (mashup-mashup network) is a weighted network between mashups, where each node represents a mashup service and edges represent the relationship between them; the weight of edge represents the relevancy between them. MMN can be represented with a triple:

$$\text{MMN} = \{N, E, W\}, \tag{3}$$

where N is a node set of all mashup services; E is a set of all edges in MMN network; W is an edge weights matrix. For example, if Node$i$ and Node$j$ are linked, then Wij is defined as the number of API intersections of two mashup services.

2.6. Structural Embedding Generation of the Mashup-Mashup Network. Structural embedding is designed to represent a network in a low-dimensional space and retain as much information as possible. By using structural embedding, each node in the network can be represented as a vector.

The traditional sequence embedding is inadequate for network structure. We can conduct a random walk on the graph structure to generate a large number of node sequences, which were input into word2vec as training samples and obtained structural embedding. Figure 4 illustrates the random walk process graphically.

As shown in Figure 4, random walk process firstly selects random mashup as starting point and produces the

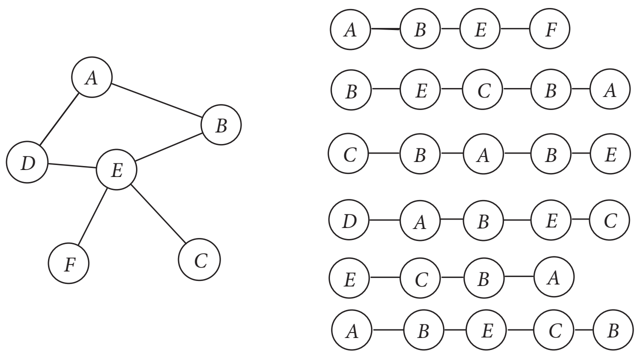


FIGURE 4: Random walk on network.

sequence, where the hop probability of the random walk needs to be formally defined, that is, the probability of traversing vj after reaching node vi. If the graph is an undirected graph, then the hop probability from node vi to node vj is defined as follows:

$$P\left(v_i \middle| v_j\right) = \begin{cases} (1-d) + d\left(\dfrac{w_{ij}}{\sum_{j\in N(v_i)} w_{ij}}\right), \\ 0, \quad e_{ij} \notin \varepsilon, \end{cases} \tag{4}$$

where $N(v_i)$ is the set of all outgoing edges of node v_i and w_{ij} is the weight of edge from nodevi to node v_j.

We further tradeoff the homogeneity and structural equivalence of network embedding by adjusting the weight of random walk.

Specifically, the homogeneity of the network means that the embeddings of nodes that are close to each other should be as similar as possible. As shown in Figure 5, the embedding expression of node u should be close to the nodes which are connected to s1, s2, s3, and s4. That is the embodiment of homogeneity.

Structural equivalence means that the embeddings of nodes with similar structures should be as close as possible. In Figure 4, both node u and node s6 are the central node of their local networks, respectively, which are similar in structure. Their embedding expressions should be similar, which is the embodiment of structural equivalence.

Random walk needs to be inclined to the process of breadth first search (BFS) in order to express the homogeneity of the network. There will be more homogeneous information in generated sequence because BFS prefers access to direct connection. On the other hand, depth-first search (DFS) can grasp the overall structure of the network through multiple hops. However, it is important to trade off the tendency of BFS and DFS. Formally, the hop probability from node v to node x is enhanced as follows:

$$P(v|x) = \lambda_{pq}(t, x) \cdot P(v|x), \tag{5}$$

where $\lambda_{pq}(t, x)$ is defined as follows:

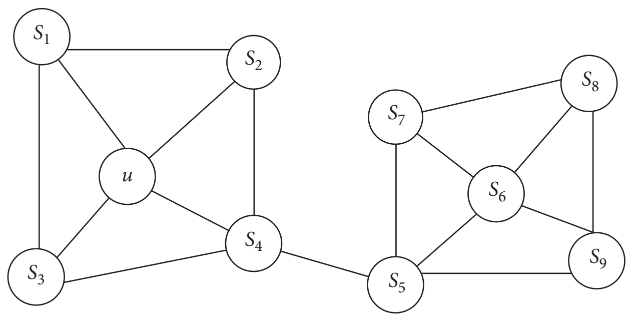


Figure 5: Homogeneity and structural equivalence of the network.

$$\lambda_{pq}(t,x) = \begin{cases} \dfrac{1}{p_r}, & \text{if } d_{tx} = 0, \\ 1, & \text{if } d_{tx} = 1, \\ \dfrac{1}{p_{io}}, & \text{if } d_{tx} = 2, \end{cases} \tag{6}$$

where d_{tx}refers to the distance from node t to node x and the parameters p_r and p_{io} jointly control the tendency of random walk. Parameter p_r is known as return parameter, which controls the probability of accessing the node visited repeatedly. Parameter p_{io} is called in-out parameter, which controls the outward or inward tendency. Random walk tends to access nodes close to node t if $p_{io} \geq 1$ and tends to access nodes far from node t if $p_{io} \leq 1$. Our approach enhances Node2vec [11] by replacing w_{ij} as $P(v|x)$.

Figure 6 shows the hop probability of our approach while jumping from node t to node v in the next step.

The homogeneity and structure equivalence of the network embodied by equation (5) can be intuitively explained in the recommendation system. Items with the same homogeneity are likely to be items of the same category and attributes, while items with the same structure are items with similar trends or structural attributes. There is no doubt that both of them are very important features in recommendation systems.

Next, we should encode above node sequences into structural embedding. Our model named sequence2-vector is derived from skip-gram, where the node sequence also utilizes central node sequence to predict the context, and its data structure can be represented by a triplet $\{s_{t-1}, s_t, s_{t+1}\}$. s_t is input, and $\{s_{t-1}, s_{t+1}\}$ is output. The neural network structure of the sequence2vector model is the common encoder-decoder architecture using the GRU model. Therefore, word vector is utilized while training sequence vector; the output of encoder is embedding vector of last word of node sequences. The formulas for the realization of encoding stage are as follows:

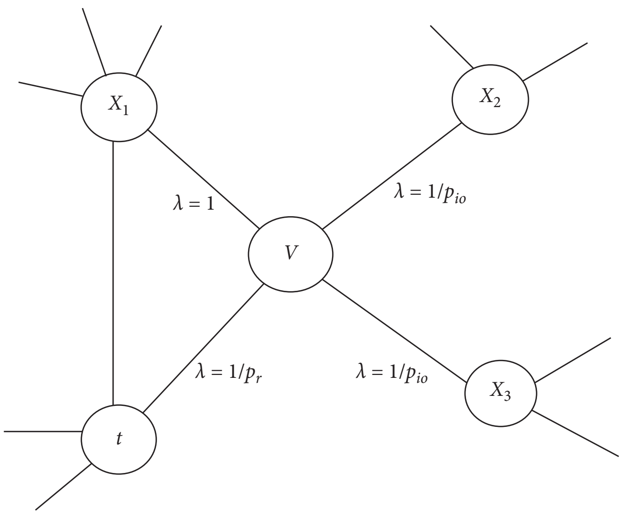


Figure 6: Hop probability of the network.

$$\begin{aligned} r^t &= \sigma\left(W_r x^t + U_r h^{t-1}\right), \\ z^t &= \sigma\left(W_z x^t + U_z h^{t-1}\right), \\ \overline{h^t} &= \tan h\left(W^d x^t + U\left(r^t \odot h^{t-1}\right)\right), \\ h^t &= \left(1 - z^t\right)\odot h^{t-1} + z^t \odot \overline{h^t}, \end{aligned} \tag{7}$$

where h^trepresents the output of the hidden layer at the time of t. While we take s_{t+1} as an example, the formula for the realization of decoding stage is as follows:

$$\begin{aligned} r^t &= \sigma\left(W_r^d x^{t-1} + U_r^d h^{t-1} + C_r h_i\right), \\ z^t &= \sigma\left(W_z^d x^{t-1} + U_z^d h^{t-1} + C_z h_i\right), \\ \overline{h^t} &= \tan h\left(W^d x^{t-1} + U^d\left(r^t \odot h^{t-1}\right) + C h_i\right), \\ h_{i+1}^t &= \left(1 - z^t\right)\odot h^{t-1} + z^t \odot \overline{h^t}, \end{aligned} \tag{8}$$

where C_r , C_z, and C are used as vector bias for reset gate, update gate, and hidden layer, respectively. Sequence2vector can even be combined into the subsequent deep learning network to retain different features of objects (Algorithm 1).

2.7. Embedding Fusion. From the above, we have got semantic embedding of API description and structural embedding through mashup behavior sequence. Then it is important to integrate multiple embedding vectors to form the final embedding of mashup. The simplest approach is to add the average pooling layer into the deep neural network to average different kinds of embedding. On this basis, we have added weights to each embedding, as shown in Figure 7. The hidden representation layer is the layer that performs the weighted average operation on different kinds of embedding. After the weighted average embedding vector is obtained, it is directly input into the softmax layer. In this way, the weighted $\alpha_i\,(i = 0, \ldots, 1)$ can be trained.

```
Input
MMN
Output
structure embedding vector h
Body
Random select a starting point for random walk using (5) and produce sequence $s$;
$h$ = sequence2vector ($s$);
return $h$
```

Algorithm 1: Structure embedding algorithm.

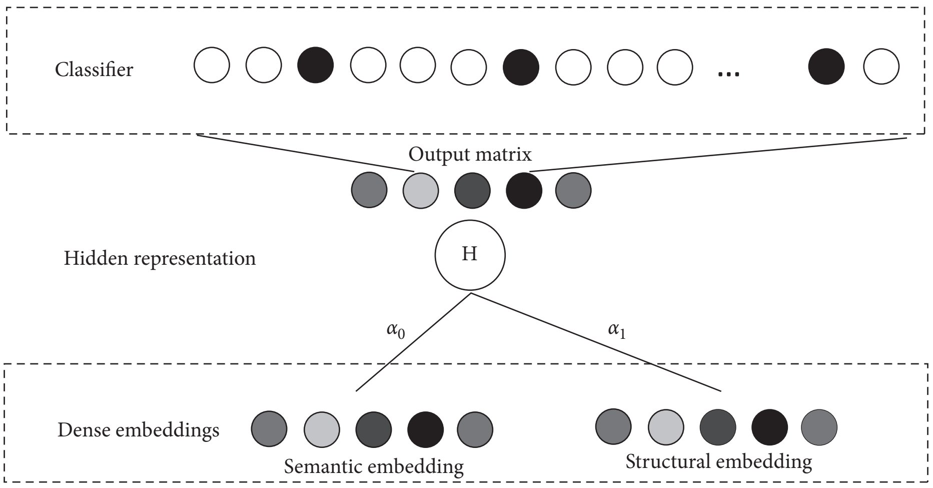


Figure 7: Embedding fusion.

In the actual model, we adopted $e^{\alpha i}$ instead of α_i as the weight of corresponding embedding, which can avoid the weight being 0 and own good mathematical properties during the process of gradient descent.

Our model finally inputs semantic and structure embedding into the model to generate the final fusion embedding.

2.8. Interfering Semantic-Related Services. In this section, we utilize fusion embedding from the above to calculate the similarity between mashups as follows:

$$\mathrm{sim}\left(h_i, h_j\right) = \exp\left(-\left\|h_i - h_j\right\|_1\right), \tag{9}$$

where $\| \|_1$ is the Manhattan distance. Finally, we recommend the top-k mashups to developers.

3. Experiments

In this section, we conduct experiments to compare our approach with state-of-the-art method. Our experiments are intended to address the following questions:

(1) Does structural embedding improve the effectiveness with respect to semantic similarities?

(2) What is the performance of top-k rankings of our embedding models?

3.1. Data Set Description. We adopt a real-world dataset, ProgrammableWeb, which was crawled from all registered services in publically available repository until Nov. 25, 2015. The dataset contains description documents including 12250 web-based services and 7875 mashups, which have attributes such as name, tags, summary, and description.

3.2. Metrics. In statistical classification [12], Precision is defined as the ratio of the number of true positives to the total number of items reported to be true, and Recall is defined as the ratio of the number of true positives to the total number of items that are true. *F*-score is defined as the weighted harmonic mean of Precision and Recall. Accuracy is defined as the ratio of sum of true positives and true negatives to the total number of items. Higher values of precision, recall, *F*-score, and accuracy indicate higher quality of identifying the semantic relationship. Based on the total number of TPs, FPs, TNs, and FNs, we computed the precision, recall, and *F*-score as follows:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$
$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad (10)$$
$$F - \text{score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}},$$

where TP denotes true positives, FP denotes false positives, FN denotes false negatives, and TN denotes true negatives.

MAP (mean average precision) is the average of precisions computed at the point of each of the relevant documents in the ranked sequence, whose equation is as follows:

$$\text{MAP} = \frac{1}{|Q_R|} \sum_{q \in QR} \text{AP}(q), \quad (11)$$

where AP is average precision and Q_R denotes relevant documents.

We utilize the normalized discounted cumulative gain (NDCG) [13] metric as well, which is a popular metric for evaluating ranking results. Given ideal rankings (used as ground truth) and the derived rankings, the NDCG value of top-k ranked services can be calculated by the following formula:

$$\text{NDCG}_k = \frac{\text{DCG}_k}{\text{IDCG}_k}, \quad (12)$$

where DCG_k and IDCG_k denote the discounted cumulative gain (DCG) values of derived rankings and ideal rankings of top-k services.

The value of DCG_k can be calculated by the following formula:

$$\text{DCG}_k = \text{rel}_1 + \sum_{i=2}^{k} \frac{\text{rel}_i}{\log_2 i}, \quad (13)$$

where rel_i is the score of the service at position i of rankings. The premise of DCG is that a high-quality service appearing lower in a ranking list should be penalized as the score is reduced logarithmically proportional to the position of the result divided by $\log_2 i$.

3.3. Baseline Approaches. In this section, to show the effectiveness of our approach, we compare our approach with the following approaches:

(1) SimpleGraph [14]: this approach extracts features of subgraphs of question-answer pairs and utilizes a logistic regression classifier to predict the probability that a candidate answer is correct

(2) AvgPara: the semantics for a given question in this model are directly averaged to predict the answers

(3) DataAugment: this approach employs semantics for data augmentation during training which utilize the question-answer pairs and semantics to automatically generate new training samples

(4) SepPara [15]: this approach is the semantic scoring model, which is trained on classification data, without considering question-answer pairs

(5) Para4QA [15]: this approach results in learning semantics, where semantic scoring and QA models are trained end-to-end on question-answer pairs

3.4. Evaluation Protocols. In this section, we have designed the experiments to evaluate our approach. We then created a set of (q, t, l) triples, where q is a API description, t is a mashup-mashup network, and l is a label (classes of mashups). We manually tagged each pair with a label l. This resulted in a set of approximately 20 human judgments. The union of them composes set T1, which is considered as the baseline to evaluate the precision of mashup recommendation.

Five authors pick out 30 API descriptions of mashups as well; top-20 rankings are artificially extracted by two authors at least after analyzing and augmenting them, which is considered as the baseline to evaluate the rankings of mashup recommendation. The union of them composes set T2.

3.5. Training Details. We utilized encoder-decoder architecture using the GRU model with 2 layers, with 800 cells at each layer and 800 dimensional word embeddings. The complete training details are given below:

(1) We used word2vector to initialize the word embedding matrix. Out-of-vocabulary words were replaced with a special unknown symbol. We also augmented API descriptions with start-of-sequence and end-of-sequence symbols. Word vectors for these special symbols were updated during training. Model hyperparameters were validated on set T1. Each of the structural and semantic features is represented by a vector of 64 dimensions. The dropout rate was selected from {0.2, 0.3, 0.4}.

(2) Parameters were randomly initialized from a uniform distribution U(−0.08,0.08). The learning rate was 0.7. After 5 epochs, we began halving the learning rate every half epoch. We trained our models for a total of 20 epochs. The batch size was set to 150. To alleviate the exploding gradient problem, the gradient norm was clipped to 5.

(3) We used the DNN model to train weighted $\alpha_i\,(i = 0, \ldots, 1)$. The DNN model takes the combination of semantic and structural features as input, and each feature is represented by a vector of 16 dimensions. The total input vector dimension is 32 $(16 + 16)$, and output vector dimension vector is the tag. To convert the input into a neural node, the hidden layer utilizes a linear transformation and then compresses the nonlinearity. The DNN model utilizes back-propagation and adjusts the weight α_i according to the training set. Finally, there is a softmax layer that translates the issue into classification problems.

Table 1: Performance of variants from the T1 test set.

Method	Precision (%)	Recall (%)	*F*-score (%)	Map (%)
SimpleGraph	86.32	26.55	40.60	16.54
AvgPara	77.99	32.13	45.51	22.62
DataAugment	71.66	35.55	47.52	32.97
Our approach	66.43	42.58	51.89	73.65

Table 2: Performance of top-*k* from the T2 test set.

Method	NDCG1 (%)	NDCG10 (%)	NDCG20 (%)
SimpleGraph	43.33	57.94	66.72
AvgPara	57.56	63.44	78.54
DataAugment	68.93	76.68	79.34
SepPara	72.30	80.84	82.49
Para4QA	81.73	82.70	87.84
Our approach	83.27	85.83	88.76

3.6. Results

3.6.1. Answer to RQ1. Table 1 presents the results of the accuracy experiments. AvgPara achieves better accuracy compared with SimpleGraph, since semantic embeddings contain more information than the features of subgraphs. DataAugment achieves better accuracy compared with AvgPara, since data augmentation works. Our approach achieves better accuracy compared with DataAugment, since structural embedding and semantic embedding are fused together.

3.6.2. Answer to RQ2. Table 2 presents the results of the ranking experiments. Our approach obtains the best prediction accuracy (largest NDCG values) under all the experimental settings consistently, since our approach considers fusion embedding of semantic and structural features, which improved the accuracy of similarity calculation.

3.7. Threats to Validity. The threats to external validity are related to the representativeness of the dataset. To evaluate the quality of semantic extraction and service recommendation, we applied our approach on real-world dataset, ProgrammableWeb, which is suitable for our study due to public availability and activeness. This threat has been partially mitigated by the fact that we collected description documents including 12250 web-based APIs and 7875 mashups. Furthermore, we applied our approach on top-10 subjects of them as well.

The threats to internal validity of our study are related to the results of the pairwise comparison on the criteria (see Section 3.5). There is a possibility that the results are subjective, as people may be biased. It may affect the weighted α_i which is assigned to the semantic and structural embedding. This threat has been partially mitigated by the fact that five authors elicit 10 service description documents from each of top-10 subjects independently to evaluate the quality of semantic relationship extraction and query, where we manually achieve top-10 rankings as the baseline in our evaluations.

4. Related Works and Discussion

Our approach touches some areas such as recommendation, DL artifacts, and requirements engineering. We next discuss relevant works pertinent to our proposed approach in these domains.

Most of recommendations focused on keyword-based searching, where the keywords are usually tokenized from the interface of web-based services, such as input, output, precondition, and postcondition. For example, Junghans et al. give an oriented functionalities and requests formalization method [16]. OWLS-MX is proposed to calculate the similarity of concept in OWL-S for recommendation [17]. Chen et al. give a clustering approach named as WTCluster using WSDL documents and service tags [18]. Mueller et al. give an approach based on nature-inspired swarm intelligence [19].

Semantic parsing offers NLP-based techniques, such as syntax directed [20], parse trees [21], combinatory categorical grammars [22, 23], and dependency-based semantics [24, 25]. These approaches typically have high precision but lower recall, which are sensitive to grammatically incorrect or ill-formed descriptions. Several approaches utilize either NLP or a merging of NLP and machine learning algorithms (SVM driven, etc.) to infer specifications, such as API descriptions [26–29].

Generally, most ranking approaches ranked services by utilizing the text in service profiles (mashup descriptions, API descriptions, WSDL documents, tags, etc.) to calculate the semantic similarity between every pair of services. In contrast, our proposed approach utilized structural similarities to guide the ranking activities, which can reduce false positives and reach higher performance.

5. Conclusion and Future Work

This paper introduces a mashup service recommendation approach via merging semantic features from API descriptions and structural features from the mashup-API network. Our approach can significantly outperform previous works after encoding semantic and structural features.

In spite of these promising results, some exciting challenges remain, especially in order to scale up this model to more dimensions. Furthermore, many more modes have to be carried out to encode the complex semantics into the embedding space.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The work described in this study was fully supported by the National Key Research and Development Program of China (no. 2017YFB1400602), the National Natural Science Foundation of China (nos. 61572371, 61702377, 61728202, 61572137, and 61873309), Science and Technology Development Program of Central Guide to Local Government of China (no. 2019ZYYD043), International Science and Technology Cooperation Program of Hubei Province (no. 2019AHB059), the Natural Science Foundation of Hubei Province in China (no. 2019CFC870), and Hubei Superior and Distinctive Discipline Group of "Mechatronics and Automobiles" (XKQ2020016).

References

[1] https://www.programmableweb.com/.

[2] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments: a survey," *Decision Support Systems*, vol. 74, pp. 12–32, 2015.

[3] V. Gervasi and D. Zowghi, "Reasoning about inconsistencies in natural language requirements," *ACM Transactions on Software Engineering and Methodology*, vol. 14, no. 3, pp. 277–330, 2005.

[4] U. Dekel and J. D. Herbsleb, "Improving API documentation usability with knowledge pushing," in *Proceedings of the 2009 IEEE 31st International Conference on Software Engineering*, Vancouver, BC, Canada, May 2009.

[5] G. Little and R. C. Miller, Keyword programming in Java, vol. 16, Springer US, Boston, MA, USA, 2007.

[6] W. Pan, B. Li, J. Liu, Y. Ma, and B. Hu, "Analyzing the structure of java software systems by weighted *k*-core decomposition," *Future Generation Computer Systems*, vol. 83, pp. 431–444, 2018.

[7] W. Pan, B. Song, K. Li, and K. Zhang, "Identifying key classes in object-oriented software using generalized *k*-core decomposition," *Future Generation Computer Systems*, vol. 81, pp. 188–202, 2018.

[8] W. Pan, H. Ming, C. K. Chang, Z. Yang, and D. K. Kim, "ElementRank: ranking java software classes and packages using multilayer complex network-based approach," *IEEE Transactions on Software Engineering*, 2019.

[9] http://www.nltk.org.

[10] https://wordnet.princeton.edu/.

[11] A. Grover and J. Leskovec, "node2vec: scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864, New York, NY, USA, August 2016.

[12] D. Olson, *Advanced Data Mining Techniques*, Springer-Verlag, Berlin, Germany, 2008.

[13] K. Kekäläinen and J. Kekalainen, "Cumulated gain-based evaluation of IR techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.

[14] S. Reddy, O. Täckström, M. Collins et al., "Transforming dependency structures to logical forms for semantic parsing," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 127–140, 2016.

[15] L. Dong, J. Mallinson, S. Reddy et al., "Learning to paraphrase for question answering," 2017, http://arxiv.org/abs/1708.06022.

[16] M. Junghans, S. Agarwal, and R. Studer, "Towards practical semantic web service discovery," in *Proceedings of the Extended Semantic Web Conference*, pp. 15–29, Heraklion, Greece, June 2010.

[17] M. Klusch, B. Fries, and K. Sycara, "OWLS-MX: a hybrid semantic web service matchmaker for OWL-S services," *Journal of Web Semantics*, vol. 7, no. 2, pp. 121–133, 2009.

[18] L. Chen, L. Hu, Z. Zheng et al., "Utilizing tags for web services clustering," in *Proceedings 2011 International Conference on Service-Oriented Computing*, pp. 204–218, Paphos, Cyprus, 2011.

[19] C. Mueller, N. Kiehne, and N. Kiehne, "Keyword-based service matching in a cloud environment using nature-inspired swarm intelligence," *Lecture Notes on Software Engineering*, vol. 3, no. 4, pp. 299–302, 2015.

[20] R. J. Mooney, "Learning for semantic parsing," *Computational Linguistics and Intelligent Text Processing*, Springer, Berlin, Germany, pp. 311–324, 2007.

[21] R. Ge and R. J. Mooney, "A statistical semantic parser that integrates syntax and semantics," in *Proceedings of the 9th Conference on Computational Natural Language Learning*, pp. 9–16, Stroudsburg, PA, USA, 2005.

[22] L. Zettlemoyer and M. Collins, "Learning to map sentences to logical form: structured classification with probabilistic categorial grammars," in *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pp. 658–666, Edinburgh, UK, 2005.

[23] L. S. Zettlemoyer and M. Collins, "Online learning of relaxed CCG grammars for parsing to logical form," in *Proceedings of the EMNLP-CoNLL*, pp. 678–687, Prague, Czech Republic, June 2007.

[24] P. Liang, M. I. Jordan, and D. Klein, "Learning dependency-based compositional semantics," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 590–599, Portland, OR, USA, June 2011.

[25] H. Poon, "Grounded unsupervised semantic parsing," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pp. 933–943, Sofia, Bulgaria, August 2013.

[26] L. Tan, D. Yuan, G. Krishna, and Y. Zhou, "/∗icomment," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, pp. 145–158, 2007.

[27] R. J. Kate and R. J. Mooney, "Using string-Kernels for learning semantic parsers," in *Proceedings of the 21st International Conference on Computational Linguistics*, pp. 913–920, Sydney, Australia, July 2006.

[28] W. Xiong, Z. Lu, B. Li, B. Hang, and Z. Wu, "Automating smart recommendation from natural language API descriptions via representation learning," *Future Generation Computer Systems*, vol. 87, pp. 382–391, 2018.

[29] W. Xiong, Z. Wu, B. Li, Q. Gu, L. Yuan, and B. Hang, "Inferring service recommendation from natural language API descriptions," in *Proceedings of the IEEE International Conference on Web Services (ICWS)*, pp. 316–323, San Francisco, CA, USA, June 2016.

Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 5102897, 20 pages
https://doi.org/10.1155/2020/5102897

Research Article

High-Dimensional Hybrid Data Reduction for Effective Bug Triage

Xin Ge, Shengjie Zheng, Jiahui Wang, and Hui Li

The College of Information Science and Technology, Dalian Maritime University, Dalian 116026, China

Correspondence should be addressed to Xin Ge; ge_xin@dlmu.edu.cn

Received 28 November 2019; Revised 22 January 2020; Accepted 11 February 2020; Published 11 May 2020

Guest Editor: Chunlai Chai

Owing to the ever-expanding scale of software, solving the problem of bug triage efficiently and reasonably has become one of the most important issues in software project maintenance. However, there are two challenges in bug triage: low quality of bug reports and engagement of developers. Most of the existing bug triage solutions are based on the text information and have no consideration of developer engagement, which leads to the loss of bug triage accuracy. To overcome these two challenges, we propose a high-dimensional hybrid data reduction method that combines feature selection with instance selection to build a small-scale and high-quality dataset of bug reports by removing redundant or noninformative bug reports and words. In addition, we also study the recent engagement of developers, which can effectively distinguish similar bug reports and provide a more suitable list of the recommended developers. Finally, we experiment with four bug repositories: GCC, OpenOffice, Mozilla, and NetBeans. We experimentally verify that our method can effectively improve the efficiency of bug triage.

1. Introduction

A large amount of data is generated during software development and maintenance. Bug reports are generated continuously during this process. A bug report contains a basic description of the bug, error messages, current status of the bug (whether it has been solved or assigned to a developer), etc. [1]. According to statistics, bug fixing increases the costs of software companies by 45% [2] and increases the time of software development. Therefore, solving the problem of bug triage efficiently and accurately, that is, assigning the bug to the most suitable developer [3–9], becomes important for software projects. Bug tracking systems were developed to track all aspects of bug messages and help software developers fix bugs in time. These systems play an important role in dispatching work between developers. Typical bug tracking systems are Mantis [10], Bugzilla [11], and JIRA [12].

The earliest method of bug triage was to manually assign a bug to a corresponding developer. The senior manager would read a bug report in the bug tracking system and would select a suitable developer for this bug report based on his/her own experience and knowledge of the developer's ability. However, this method of assignment not only wastes time and human resources, but also has limited accuracy. First, in large-scale software development, the number of bugs dramatically increases, and the quality of bug reports may vary. In 2001–2010, 333,371 bugs were submitted to Eclipse from more than 34,917 developers. When developers submit many duplicated or invalid bugs, a lot of time is wasted [13–15]. Moreover, owing to a large number of developers, senior managers cannot remember the bug-fixing skills of each developer and the types of bugs that this developer is good at. Thus, senior managers may assign bugs improperly, which may reduce the accuracy of bug fixes. Recently, a novel approach based on software networks is proposed to address software quality-related problems, e.g., in the perspective of bug network [16] and social networks [17]. These methods need two essential premises, constructed network extracted from software and extra information such as the social relationship of developers. Due to these drawbacks, it is difficult to widely adopt to improve software quality and bug triage.

To solve the above problems of manual bug triage and improve classification accuracy, Murphy et al. [5] proposed to apply text categorization technology to bug triage; it automatically generated a list of recommended developers after training on a bug dataset with developer labels. In their research, a bug report is an instance. The words in the bug report are the corresponding attributes. The developer is the label of an instance. Then, they apply the classification algorithm to predict the best developer sequence for bug fixes. Subsequently, some researchers used the vector space model to represent the bug report [2, 7, 18–24]. In addition, the theme model was used to identify different documents, and the words within the document have a specific relationship to each theme [25]. Researchers [26–31] not only used the theme model but also studied metadata (such as products, components, and operating system types in the bug report) to improve the accuracy.

Two challenges of automatic bug triage technology remain to be solved. First, the original bug repositories are large scale with low-quality bug reports. Many problems may arise due to large scale and low quality, such as extensive computations and a decline in predictive performance. Second, most of the existing work ignores the influence of developer engagement. Developers who have joined recently may be relatively more active because other developers may change positions or leave.

In this study, we propose a high-dimensional hybrid data reduction method to combine feature selection with instance selection method to build a small-scale and high-quality set of bug reports by removing redundant or noninformative bug reports and words. Our method combines feature selection (FS) with instance selection (IS) using the differential evolution (DE) method with updated rules of crossover and variation. In addition, we consider the developer engagement in each project for more reasonable bug triage. If the developer has dealt with bugs with this kind of product information before, we consider he/she is more active, and we are more likely to assign this bug report to him/her. We conduct experiments on four public bug repositories: GCC, OpenOffice, Mozilla, and NetBeans. The main contributions of this paper are summarized as follows:

(1) We present a high-dimensional hybrid data reduction method based on DE to obtain a small-scale and high-quality dataset for bug triage. Specifically, we aim to address two aspects: (a) to simultaneously reduce the bug reports dimension and the words dimension and (b) to improve the performance of bug triage.

(2) We consider the developer engagement level further and reorder the optimal list of recommended developers, combining it with product information that is related to bug reports and developers.

(3) We verify the effectiveness of our proposed method on four bug repositories (GCC, OpenOffice, Mozilla, and NetBeans). The experimental results show that the results obtained by using our proposed method are superior to the existing methods.

The rest of the paper is structured as follows. Section 2 presents closely related work on bug triage. Section 3 details the proposed DE method, which is improved in our paper. Section 4 describes our experiment with GCC, OpenOffice, Mozilla, and NetBeans and explains the results. Finally, Section 5 concludes our paper and describes future work.

2. Related Work

2.1. Methods of Bug Triage. The main purpose of bug triage is to find the right developer to fix a newly submitted bug [2, 7, 11, 19, 20, 32–34]. At present, machine learning has been used to accomplish automatic bug dispatching. When we use machine learning, the developer is considered as the category tag of the bug, and the text information of the bug is regarded as a feature. First, the algorithm learns on historical bug data. Then, the algorithm can predict which developer should be assigned to a new bug report. Murphy and Cubranic [7] propose a text classification approach to bug triage. Based on the Naïve Bayes classification algorithm, a list of recommended developers was predicted for Eclipse. Anvik et al. [2] used a variety of machine learning algorithms (such as Naïve Bayes, SVM, and C4.8) to learn historical data for bug triage. Tamrawi et al. [23] proposed "Bugzie," a method in which a fuzzy set was used to simulate the developer's expertise to determine whether a new bug report is suitable for this developer. Naguib et al. [35] used a latent Dirichlet allocation language model to learn the similarity between new bug reports and developers by learning from previously available data on bugs already fixed by developers. Yang et al. [27] divided bug reports into different topics. Zhang et al. [36] considered both developer relationships and topic models to recommend the bug report to a developer who would fix it better. Jeong et al. [21] used a Markov chain-based graph model to improve bug triage accuracy and tested it on Eclipse and Mozilla. Xia et al. [37] proposed a composite method named DevRec and analyzed bug reports and developers simultaneously.

Except for the bug report features, other information is also used to classify bugs. Linares-Vásquez et al. [38] analyzed the title comments of related documents and found relevant documents to recommend developers. Bhattacharya and Neamtiu [22] studied the tossing graph with various attributes, which included the developers who could not fix this bug and others who could fix it, to improve the triage accuracy. Kevic et al. [39] assigned the current bug to a developer by finding a developer whose fixed bugs were similar to the current bug report. Wang and Lo [40] used a new approach named FixerCache, which assigned new bug reports to developers based on the developer's enthusiasm for different product components. In the study by Shokripour et al. [41], four information resources were considered to obtain a list of recommended developers. Xia et al. [37] proposed the TopicMinerMTM model, which used training data to model the topic distribution of bug reports. However, Xia's approach had difficulties with distinguishing similar developers.

2.2. Bug Triage and Software Networks. Besides the bug reports, researchers adopt more information to address the bug-related problems. Because the quality of a software system is partially determined by its structure (topological structure), software systems can be modeled as complex networks in which software components are abstract nodes and their interactions are abstract edges. Therefore, researchers have proposed many approaches and measures based on general software networks. Zhang and Lee [42] proposed an automated developer recommendation approach for bug triage via building the concept profile (CP) for extracting the bug concepts with topic terms from the documents produced by related bug reports. They identify and rank the important developers by using social network (SN) for bug fixing. Because of the functional form of the incoming link distribution of software dependence network, software is fragile with respect to the failure of a random single component. Locating a faulty component is easy if the failure only affects its nearest neighbors, while it is hard if it propagates further. Challet and Lombardoni [43] addressed the issue of how software components are affected by the failure, and the inverse problem of locating the faulty component through adopting bug propagation and debugging in asymmetric software structures. Pan et al. [44] also analyzed the bug propagation process based on the weighted software networks (WSNs). It considered the process of a bug in one class propagating to the other and is effect to locate the source of component with bug. The aforementioned approaches take advantage of complex networks, especially the relationship among developers, users, software components, and bug reports. In the perspective of software networks, bug triage involves developers and bug reports, and thus, the improvement of the accuracy of bug triage is able to promote and enhance the effectiveness of software network-related methods. For example, bug triage can help us gain a deeper insight of bug propagation on the software networks. Correspondingly, the novel measurements based on software networks may provide more useful information for bug triage. In other words, they are mutually beneficial.

In summary, we find that these traditional feature selection approaches do not have sufficient search range and depth and do not consider chronological order. Therefore, the denoising ability is limited, and the effect on data reduction is not obvious. This study addresses these two aspects. First, we fully consider the chronological order of bug reports. Second, we propose to improve the differential evolution method based on this to expand the search range and search depth. Meanwhile, we also can ensure the convergence of the method, and the accuracy of bug triage increases effectively. Unlike the above research, our paper focuses more on developer engagement. Except for the text provided in the bug report itself, the bug report and the developer's developers' bug-fixing experience are also used to obtain the best list of recommended developers. When the similarity of different bug reports is very high, our approach helps the classifier to strictly divide different products to complete the bug triage task successfully and substantially improve the efficiency of bug triage.

3. The Proposed Algorithm

3.1. Overview. In this section, we propose a high-dimensional hybrid data reduction method for bug triage, as shown in Figure 1. The method includes three main phases: data preprocessing, data reduction, and developer engagement detection. In the data preprocessing phase, we preprocess each bug report using word segmentation, stop words, stemming, and vector space representation to obtain the word vectors. In the data reduction phase, we propose a high-dimensional hybrid data reduction method to combine feature selection with the instance selection method to build a small-scale and high-quality dataset of bug reports by removing redundant or noninformative bug reports and words. In the developer engagement detection phase, we consider the influence caused by the product information of the current test case, which can include the level of developer engagement in the final order of recommended developers. These three models are described in the following sections.

3.2. Data Preprocessing. After extracting text and developer information from the bug report tracker, our method preprocessed the data to obtain word vectors for each bug report. The text preprocessing includes tokenization, stop-word removal, stemming, and keyword vector representation [11]. Specifically, tokenization converts text from the original bug report into a set of words. Stop-word removal refers to removing insignificant words that appear frequently in bug reports (such as "the" or "in"). Stemming transforms words that may appear in different forms into their basic form (for example, "computerized" can be changed into "computer"). Keyword vector representation produces a word vector to model a bug report after previous steps, and we delete words with the word frequency less than 10. Finally, our method uses a multidimensional vector space, where each word represents a dimension to describe the processed bug report collection. Every bug report is a vector based on the word dimensions, as shown in the following equation:

$$R = w_1, w_2, w_3, \ldots, w_i, \ldots, w_n, \quad i \in [1, n], \qquad (1)$$

where R is a bug report and w_i refers to i word in the word dimension.

3.3. Data Reduction. After using the bug report preprocessing model, our method obtains word vectors for bug reports. Considering that some bug reports will become outdated over time, the data will be noisy and redundant. To reduce the impact of chronological order, our method divides the bug reports into three parts according to the ratio of 7 : 1 : 2 from small ID to large ID (the first 70% is the training set, the middle 10% is the verification set, and the last 20% is the test set). In the search process, our method first uses the training set for training and the verification set for examining to find the optimal solution results with FS, IS, and FS + IS. Next, the combination of the training set and validation set is regarded as a new training set, and our method obtains a final list of Top-10 developers for the test set. Subsequently, we will explain the rules of the DE method and the data reduction method of FS, IS, and FS and IS simultaneously (FS + IS).

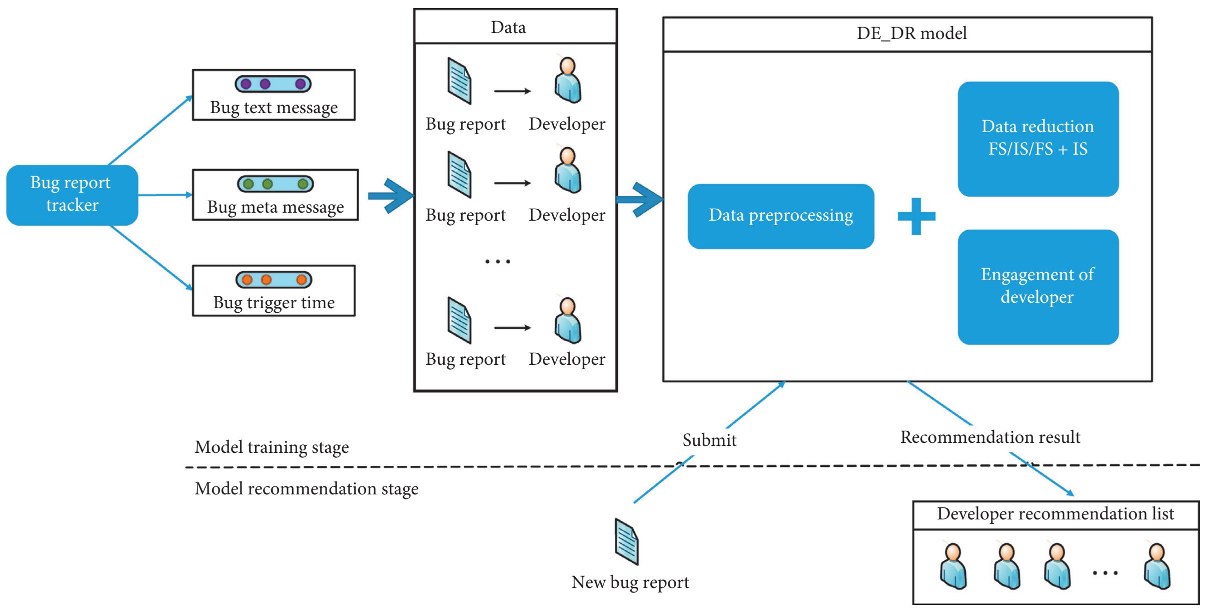


Figure 1: Flow diagram of bug triage process.

3.3.1. Population Coding. To represent feature combination and instance combination clearly and intuitively, we adopt binary coding: our method uses a feature vector with n feature dimensions to replace the feature sequence in the population, and it transforms the selected feature combinations into one 0, 1 binary string by the following equation:

$$F_i = f_1, f_2, f_3, \ldots, f_i, \ldots, f_N, \quad i \in [1, N], \tag{2}$$

where F_i is a feature vector corresponding to a feature sequence, $f_i = 0$ indicates that feature i is not selected and $f_i = 1$ indicates that feature i is selected, and N is the total number of features. Thus, a binary string represents a feature selection method.

Similarly, the sequence of instances in the dataset can be represented as a feature vector of $1 \times M$, and the selected instance combination is also a 0, 1 binary string, which is described by the following equation:

$$S_j = s_1, s_2, s_3, \ldots, s_j, \ldots, s_M, \quad j \in [1, M], \tag{3}$$

where S_j is a feature vector corresponding to an instance sequence, $s_j = 0$ indicates that instance j is not selected and $s_j = 1$ indicates that instance j is selected, and M is the total number of instances. A binary string represents an instance selection method.

3.3.2. Population Initialization. The population initialization in DE_IS (initial feature selection scheme): we sort bug reports by their IDs, extract all features to obtain a feature set, and generate 10 initial selection options that select 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, and 100%, respectively.

The population initialization in the initial feature selection scheme (DE_IS): we generate a random number between [0, 1] for each position in the binary string of each scheme in the instance selection scheme. If the number is less than 0.5, this position is set to 0. Otherwise, it is set to 1. In this case, our method can obtain ten initial instance selection schemes.

The initial extraction scheme in DE_(FS + IS) (combined scheme of feature selection and instance selection): first, our method generates 10 initial schemes according to the initial population generation methods in DE_FS and DE_IS; second, the generated features and instances are combined with the corresponding number of binary strings. Then, our method obtains 10 extraction schemes in the initial population.

For each individual in the initial population, after calculating their values of fitness, our method records the maximum value and its corresponding code value.

3.3.3. Genetic Manipulation

(1) Variation (Differential Variation). First, our method generates a variation rate P_{rv} randomly between [0, 1] and defines the differential variation rate as P_d (P_d in this method is always dynamic; it is judged according to the number of iterations that our method is inclined to random variation or differential variation, which takes place between the individual and the currently optimal individual. This is described by the following equation:

$$P_d = \mathrm{de}\left(\frac{1 - \mathrm{index}}{\mathrm{total\,Generation}}\right), \tag{4}$$

where de is the differential coefficient of variation, index is the current iteration number, and total Generation is the number of iterations. If $P_{rv} < P_d$, we use this extraction scheme as a parent to carry out differential variation. Otherwise, we do not perform variation. In differential variation, Variation num (the number of mutated genes) positions of variant genes (L_v) are randomly selected in the scheme, and differential variation rule is defined by the following equation:

$$L_n = \begin{cases} 1, & L_v = L_b \text{ or } (L_v \neq L_b \text{ and } D_r \geq 0.5), \\ 0, & L_v \neq L_b \text{ and } D_r < 0.5, \end{cases} \quad (5)$$

where L_v is the current variant gene position, L_b is the optimal individual corresponding gene position, L_n is the corresponding gene position of the new individual, and D_r is a random number. If L_v is different from L_b, L_n is 1; if not, a random number D_r between [0, 1] is generated. If $D_r \geq 0.5$, then L_n is 1; otherwise, it is set to 0. The other genetic positions of the new individual are identical to the currently selected parent.

(2) Crossover. Similarly, a crossover probability P_{rc} is randomly generated at start. The crossover variation rate is defined as P_c (also dynamically changed), and the formula is shown in the following equation:

$$P_c = \text{aberrance Rate}\left(\frac{\text{index}}{\text{total Generation}}\right), \quad (6)$$

where aberrance Rate is the cross coefficient of variation, index is the current iteration number, and totalGeneration is the number of iterations. If $P_{rc} < P_c$, cross-variation is performed to create a child generation. When cross-variation works, two positive integers P_1 and P_2 are randomly generated ($P_1 < P_2$ and $P_1, P_2 \in [1, \text{length}]$), then the code between P_1 and P_2 is cross-operated according to the midpoint MID, where MID = $(P_1 + P_2)/2$. The rest of the new individual is the same as the parent.

(3) Variation (Random Variation). For each extraction scheme of population, our method generates a variation rate P_{rv} and defines the random variation rate as P_r (also dynamically changed); its formula is defined as follows:

$$P_r = \text{heredity Date}\left(\frac{1 - \text{index}}{\text{total Generation}}\right), \quad (7)$$

where heredityDate is the random variation coefficient, index is the current iteration number, and totalGeneration is the number of iterations. When P_{rv} is less than P_r, the extraction scheme mutates; otherwise, it is not mutated. During the random variation, Variation num positions of variant genes (L_v) are randomly selected in the binary string, and the random variation rules are described as follows:

$$L_n = \begin{cases} 1, & L_v = 0, \\ 0, & L_v = 1, \end{cases} \quad (8)$$

where L_v is the current variant gene position and L_n is the corresponding gene position of the new individual. If its value is 0, it will be changed to 1, and the other condition is the opposite.

(4) Selection. We calculate the fitness values of all individuals in the population. Our method not only conserves the new individuals that are generated by the crossover, but also the parent individuals. Subsequently, a new population with twice the size is aggregated. Next, all individuals are sorted according to their fitness value in descending order. The former half is selected, and the latter half is eliminated. Meanwhile, the binary code that has optimal fitness is updated.

Table 1: Parameters.

Parameter	Meaning
Train	The training set
NP	Population number
N	The number of features
M	The number of bug reports
Variation num	The number of mutated genes
totalGeneration	The number of iterations
P_d	The differential variation rate
P_c	The crossover variation rate
P_r	The random variation rate

3.3.4. Data Reduction Algorithm. After our analysis, both FS and IS can achieve the goal of extracting useful attributes. FS can be reduced at the attribute level, and IS can be reduced at the instance level. The effect of FS + IS might surpass the result obtained by using FS and IS separately. Thus, we choose three feature extraction methods: FS, IS, and FS + IS.

The parameters used in this section are defined in Table 1.

$$\begin{aligned} P_d &= \text{de}\frac{1 - \text{index}}{\text{totalGeneration}}, \\ \text{de} &= \alpha, \end{aligned} \quad (9)$$

$$\begin{gathered} P_c = \text{aberranceRate}\left(\frac{\text{index}}{\text{totalGeneration}}\right), \\ \text{aberranceRate} = \beta, \end{gathered} \quad (10)$$

$$\begin{aligned} P_r &= \text{heredityDate}\frac{1 - \text{index}}{\text{totalGeneration}}, \\ \text{heredityDate} &= \gamma. \end{aligned} \quad (11)$$

Algorithm 1 indicates that we carry out feature selection (DE_FS) or instance selection (DE_IS) on the dataset. In the first line, we empty the initial extraction scheme and the best best_individual scheme. In lines 2–6, we judge whether it is a feature selection or an instance selection, and we call Algorithm 2 for the former and Algorithm 3 for the latter. In lines 7-8, we calculate the fitness value of each extraction scheme in population through the fitness function (the accuracy of Naïve Bayesian Mode (NBM)) and record the extraction scheme with the largest fitness value. For each of these three variants, parent can only choose one method to mutate and generate a child. Lines 13–26 are the choice of differential variation. And lines 28–36 present the choice of cross-variation. The choice of random variation is in lines 38–44. In lines 49–54, we unite the parents and child generated, sorting the fitness in descending order, select the best Top 10 to enter the next generation, and update the optimal fitness and the corresponding binary code. In line 57, after executing T iterations, we decode the saved optimal extraction scheme into a corresponding dataset

```
Input: $Train$, $NP$, $T$, $N$, $M$, $hereditvDate$, $aberranceRate$,de
Output: Reduced_$Train$
Population ⟵ ∅best_individual ⟵ ∅
if feature select then
    Population ⟵ Initialize_FS(Train, NP, N)
else
    Population ⟵ Initialize_IS(Train, NP, M)
end if
Calculate the fitness value of each individual by NBM
best_individual ⟵ The individual with the largest fitness value
$T = 1$
while $T < totalGeneration$ do
    for all I from 1 to N do
        //Difference variation
        if randomf$(0, 1) < de(1 - index/totalGeneration)$ then
            Randomly generate Variation_Num mutation location
            Generates a new individual exactly the same as the current selected parent
            for all j from 1toVariation_Num do
                if $father[j] \neq best[i]$ then
                    $kid[i] = 1$
                else
                    if randomf$(0, 1) < 0.5$ then
                        $kid[i] = 0$
                    else
                        $kid[j] = father[j]$
                    end if
                end if
            end for
            //Crossover
        else
            if randomf$(0, 1) < aberranceRate(index/totalGeneration)$ then
                //Generatesanewindividualexactlythesameastheselectedparent
                if featureselect then
                    P1andP2arerandomlygeneratedfrom0toN $(P1 < P2)$
                else
                    P1andP2arerandomlygeneratedfrom0toM $(P1 < P2)$
                    P1andP2areswappedinsectionsaccordingtothemidpoint
                end if
                //Randomvariation
            else
                if randomf$(0, 1) < hereditvDate(1 - index/tnoqtha_{l}xG7eCn; eration)$ then
                    Generatesanewindividualexactlythesameastheselectedparent
                    RandomlygenerateVariation_Nummutationlociation
                    for alljfrom1toVariation_Num do
                        Ifthelocusis0, setitto1, andifitis1thensetitto0
                    end for
                end if
            end if
        end if
    end for
    Calculate fitness values for all individuals
    Select theTop10 from all individuals sorted by fitness value from large to small
    if the current optimal value > the original optimal value then
        update the current optimal value and the corresponding binary code
    end if
    Updating Population
    $T++$
end while
The binary string corresponding to best_individual is decoded as Reduced_Train
return Reduced_Train
```

Algorithm 1:Reduction (DE).

```
     Input: $Train$, $NP$, $N$
     Output: $Initialize\_Population\_FS$
(1)  $Initialize\_Population\_FS \leftarrow \varnothing$
(2)  $metrics \leftarrow [IG]$
(3)  for $each\ metric\ in\ metrics$ do
(4)    $Order features by metric$
(5)    for $all i from 1\ to 10$ do
(6)      $Take\ the first\ i \times 10\ of N\ as\ fs$
(7)      $Initialize\_Population\_FS.add(fs)$
(8)    end for
(9)  end for
(10) return $Initialize\_Population\_FS$
```

Algorithm 2: Initialize_FS.

```
     Input: $Train$, $NP$, $M$
     Output: $Initialize\_Population\_IS$
(1)  $Initialize\_Population\_IS \leftarrow \varnothing$
(2)  for $all i from\ 1\ to\ NP$ do
(3)    $Treat\ individual\ i\ as\ ins$
(4)    for $ins\ gene\ location\ j\ from\ 1\ to\ M$ do
(5)      $rate = Random(0, 1)$
(6)      if $rate \geq 0.5$ then
(7)        $Change\ the\ j\ locus\ of\ the\ ith\ individual\ to\ 1$
(8)      else
(9)        $Change\ the\ j\ locus\ of\ the\ ith\ individual\ to\ 0$
(10)     end if
(11)   end for
(12) end for
(13) $Initialize\_Population\_IS.add(ins)$
(14) return $Initialize\_Population\_IS$
```

Algorithm 3: Initialize_IS.

Reduced_Train. In line 58, we return Reduced_Train, which is reduced (if this gene position is different from the corresponding position of an optimal individual, the same position of the new individual is 1. Otherwise, a random number between [0, 1] is generated. If this number is equal or greater than 0.5, the position of the new individual is 1; otherwise, it is set to 0. The other genetic positions of the new individual are identical to the currently selected parent).

Algorithm 2 represents the population initialization process when we select features using the DE method. In the first line, we empty the initial extraction scheme. In line 2, the IG is used to select features and is recorded as metrics. In lines 3–9, we sort the training set from high to low according to the importance of features and select top 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, and 100% to add them into the feature selection scheme (Initialize_Population_FS). We set the extracted feature column to 1 and the opposite feature column to 0.

Algorithm 3 represents the population initialization process when we select instances using the DE method. In the first line, we empty the initial extraction scheme. In lines 2–11, each instance selection scheme is represented as a $1 \times M$ binary string. Then, we generate a random decimal between 0 and 1 for each gene position of each extraction scheme. If the number is greater than or equal to 0.5, the corresponding gene position of the individual is 1. Otherwise, it is set to 0. We add generated instance selection schemes to Initialize_Population_IS. In line 14, we return all generated extraction schemes.

DE_(FS + IS) indicates that we select features and instances simultaneously for the dataset using the DE method. In line 1, we empty the initial extraction scheme and the best best_individual scheme. In lines 2–6, we combine the initialized feature selection scheme with the initialized instance selection scheme. In lines 7-8, we calculate the fitness value of each extraction scheme to save the best extraction scheme and its fitness value. Differential variation is described in lines 11–26. Cross-variation is detailed in lines 28–37. Lines 39–47 present the random variation. In lines 51–56, parents and children are mixed and sorted by the fitness values to select the best Top 10, which can enter the next generation. The optimal fitness and binary code are updated. In line 60, after T iterations, we obtain Reduced_Train by decoding the best extraction scheme. In line 61, we return the reduced Reduced_Train.

3.4. Engagement Detection. After analyzing the actual situation, we find that some developers may change jobs or leave the company over time. Meanwhile, a bug report that has the same problem with different product information may be very similar. Thus, we reorder developers depending on their level of engagement and the product information. First, we find the corresponding product information in the latter N bug reports of each developer's training set (the original training set and the validation set, which analyze developer engagement using recent bug reports). Subsequently, we use a linked list to encapsulate the data to store more product information. Next, the NBM classifier is used to select the Top 30 developers. We record the product information of current bug reports and check whether each developer has dealt with the same product information or not. If not, we choose to discard this developer, and the subsequent developers will fill up the empty positions one by one. Finally, we can select the optimal Top 10 in this way (Algorithm 4).

4. Experiment

We conduct several experiments to verify our method. In Section 4.1, we describe the experimental design, including data preparation and experimental setup. In Section 4.2, we specify the evaluation metrics. In Section 4.3, we discuss experimental results, which answer our research questions.

4.1. Experimental Design

4.1.1. Data Preparation. To demonstrate the effectiveness of our approach, we carry out a series of experiments on four large-scale open-source bug repositories: GCC, OpenOffice, NetBeans, and Mozilla. We only collected the fixed bug reports which were denoted as "resolved" or "closed" before December 31, 2014, due to their stability and reliability. We have uploaded our experimental datasets to URL (https://github.com/gexinxinge/complexnetwork). Table 2 shows the dataset statistics, including the total number of products, total number of reports, total number of words, total number of developers, and time range.

We use the processing method from the previous research [23, 26]. The label of the developer who was assigned to the bug report is regarded as the developer who fixed this bug. To obtain a list of recommended developers, we exclude the bugs that were never fixed [2]. Moreover, data on developers who worked on a small number of bug reports would not be helpful for obtaining accurate results; therefore, we exclude the developers who worked on less than 10 bug reports from our study.

4.1.2. Experimental Setup. In this paper, we segment data by a chronological method. The bug reports are divided into three parts in chronological order. Seventy percent is used as the training set, ten percent is used as the validation set, and the remaining twenty percent is used as the test set. We make the following adjustments to the parameters (the experimental parameters of the three differential methods are consistent): $\alpha = 0.7, \beta = 0.08, \gamma = 0.8$. It is worth pointing out that our approach is sensitive to these parameters. In equation (9), α makes an influence on differential variation rate (Pd). In equation (10), β decides the value of crossover variation rate (Pc). And in equation (11), γ affects the random variation rate (Pr). For example, if the values of α and γ are too large, it is hard to remove unimportant features. However, small values may lead to the local optimum. Therefore, we use reasonable values published by others' experience using difference algorithm. In this way, our method can not only have sufficient search scope, but also converge in the end.

We use two types of benchmarks for comparison: supervised methods and unsupervised methods. Supervised benchmark methods mostly include classifiers that are effective in text classification: Naïve Bayes (NB), polynomial Naïve Bayes (NBM), support vector machine (SVM), k-nearest neighbor (KNN), random tree (RT), and decision tree (J48).

4.2. Evaluation Metrics. In this paper, we use the accuracy of Top-k developers to evaluate the quality of bug triage, that is, the ratio of the predicted exact quantity to all test data. The accuracy of Top-1 to Top-10 is examined in the experiment altogether. The accuracy of Top-k is calculated by the following equation:

$$\text{Accuracy}_k = \frac{\text{Bug\_}k_{\text{correct}}}{\text{Bug}_{\text{total}}}, \tag{12}$$

where Accuracy_k is the accuracy of Top-k, $\text{Bug\_}k_{\text{correct}}$ refers to the number of bug reports that are assigned correctly to Top-k, and $\text{Bug}_{\text{total}}$ is the total number of bug reports in Top-k. A higher Accuracy_k describes a better list of recommended developers.

4.3. Experimental Results

4.3.1. RQ1: Which Supervised Learning Algorithms Are the Most Suitable? In this experiment, six supervised learning algorithms (NB, NBM, SVM, KNN, RT, and J48) are applied to classify four open-source bug repositories (Mozilla, NetBeans, OpenOffice, and GCC). We use the classification accuracy as an evaluation metric to choose the best classification algorithm.

The results are shown in Tables 3–6. According to the results, the classification effect of NBM (which is presented in bold) is the best among the six methods. The highest classification accuracies of NBM on four bug repositories are 28.89%, 57.59%, 48.39%, and 68.84%, respectively. Moreover, NB ranks second; its highest classification accuracies are 23.69%, 54%, 37.94%, and 50.27%, respectively. Both methods have much higher classification accuracy than other classifiers.

For OpenOffice, the accuracies of NBM from Top-1 to Top-10 are much higher than those of other algorithms (NB, NBM, SVM, KNN, RT, J48). Additionally, the Top-10 accuracy of NBM is 37.33% higher than the Top-10 accuracy of SVM on OpenOffice. Similarly, Table 6 shows that NBM performs the best among six approaches (NB, NBM, SVM,

```
     Input: $Train$, $NP$, $T$, $N$, $M$, $hereditytDate$, $aberranceRate$, de
     Output: $Reduced\_Train(FS + IS)$
(1)  $Population \leftarrow \varnothing$ $best\_individual \leftarrow \varnothing$
(2)  for $all\ i from 1 to\ NP$ do
(3)     $Initialize\_Population\_FS = Initialize\_FS(Train, NP, N)$
(4)     $Initialize\_Population\_IS = Initialize\_IS(Train, NP, M)$
(5)     $Population \leftarrow Combine(Initialize\_Population\_FS, Initialize\_Population\_IS)$
(6)  end for
(7)  $Calculate\ the\ fitness\ value\ of\ each\ individual\ by\ NBM$
(8)  $best\_individual \leftarrow The\ individual\ with\ the\ largest\ fitness\ value$
(9)  while $T < totalGeneration$ do
(10)    for $all\ i from\ 1\ to\ NP$ do
(11)       $//Difference\ variation$
(12)       if $randomf(0, 1) < de(1 - index/tnoqtha_l xG7eCn; eration)$ then
(13)          $Randomly\ generate\ Variation\_Num\ mutation\ location$
(14)          $Generates\ a\ new\ FS\ individual\ exactly\ the\ same\ as\ the\ selected parent$
(15)          $Generates\ a\ new\ IS\ individual\ exactly\ the\ same\ as\ the\ selected parent$
(16)          for $all j from\ 1\ to\ Variation\_Num$ do
(17)             if $father[j] \neq best[i]$ then
(18)                $kid[i] = 1$
(19)             else
(20)                if $randomf(0, 1) < 0.5$ then
(21)                   $kid[i] = 0$
(22)                else
(23)                   $kid[j] = father[j]$
(24)                end if
(25)             end if
(26)          end for
(27)          $//Crossover$
(28)       else
(29)          if $randomf(0, 1) < aberranceRate(index/tnoqtha_l xG7eCn; eration)$ then
(30)             $Generates\ a\ new\ FS\ individual\ exactly\ the\ same\ as\ the\ selected parent$
(31)             $Generates\ a\ new\ IS\ individual\ exactly\ the\ same\ as\ the\ selected\ parent$
(32)             $//Cross\_feature\ select$
(33)             $P1\ and\ P2\ are\ randomly\ generated\ from\ 0\ to\ N(P1 < P2)$
(34)             $P1\ and\ P2\ are\ swapped\ in\ sections\ according\ to\ the\ midpoint$
(35)             $//Cross\_instance\ select$
(36)             $P1\ and\ P2\ are\ randomly\ generated\ from\ 0\ to\ M(P1 < P2)$
(37)             $P1\ and\ P2\ are\ swapped\ in\ sections\ according\ to\ the\ midpoint$
(38)             $//Random\ variation$
(39)          else
(40)             if $randomf(0, 1) < hereditytDate(1 - index/tnoqtha_l xG7eCn; eration)$ then
(41)                $Generates\ a\ new\ IS\ individual\ exactly\ the\ same\ as\ the\ selected parent$
(42)                $Generates\ a\ new\ IS\ individual\ exactly\ the\ same\ as\ the\ selected parent$
(43)                $Randomly\ generate\ Variation\_Num\ mutation\ lociation$
(44)                for $all j from1\ to Variation\_Num$ do
(45)                   $If\ the\ locus\ is\ 0, set\ it\ to\ 1, and\ if\ it\ is\ 1\ then\ set\ it\ to\ 0$
(46)                end for
(47)             end if
(48)          end if
(49)       end if
(50)    end for
(51)    $Calculate\ fitness\ values\ for\ all\ individuals$
(52)    $Select\ the\ Top10\ individuals\ sorted\ by\ fitness\ value$
(53)    $Enter\ the\ next\ generation\ and\ eliminate\ the\ rest.$
(54)    if $the\ current\ optimal\ value\ > the\ original\ optimal\ value$ then
(55)       $update\ the\ current\ optimal\ value\ and\ the\ corresponding\ binary\ code$
(56)    end if
(57)    $Updating\ Population$
(58)    $T++$
(59) end while
(60) $The\ binary\ string\ corresponding\ to\ best\_individual\ is\ decoded\ as$ Reduced $\_Train(FS + IS)$
(61) return $Reduced\_Train(FS + IS)$
```

Algorithm 4: DE_(FS + IS).

Table 2: Statistics of datasets.

	Product	Bug reports	Words	Developers	Period
GCC	2	13961	51005	263	1999/08/03–2014/12/01
OpenOffice	37	23475	50166	553	2000/10/21–2014/12/31
Mozilla	12	18793	33186	1022	1999/03/17–2014/12/31
NetBeans	12	21538	41256	265	1999/02/11–2014/12/31

Table 3: The accuracy of Mozilla using different classifiers.

Mozilla	NB	NBM	SVM	KNN	RT	J48
Top-1	0.0815	**0.0906**	0.0226	0.0509	0.046	0.0997
Top-2	0.1143	**0.1463**	0.0226	0.0509	0.046	0.1334
Top-3	0.1369	**0.1833**	0.0352	0.0627	0.0582	0.1477
Top-4	0.1578	**0.2045**	0.0352	0.0634	0.0582	0.1547
Top-5	0.1749	**0.2233**	0.0387	0.0662	0.0613	0.1599
Top-6	0.1868	**0.2404**	0.0397	0.0672	0.0624	0.1624
Top-7	0.1976	**0.2551**	0.0397	0.0672	0.0624	0.1652
Top-8	0.2129	**0.2669**	0.0397	0.0672	0.0624	0.1693
Top-9	0.2265	**0.278**	0.0397	0.0672	0.0624	0.1704
Top-10	0.2369	**0.2889**	0.0397	0.0672	0.0624	0.1704

Table 4: The accuracy of NetBeans using different classifiers.

NetBeans	NB	NBM	SVM	KNN	RT	J48
Top-1	0.2157	**0.2917**	0.1478	0.1184	0.1184	0.2163
Top-2	0.3041	**0.3972**	0.1478	0.1184	0.1184	0.2783
Top-3	0.3649	**0.4311**	0.1478	0.1184	0.1184	0.2979
Top-4	0.4039	**0.4612**	0.1548	0.1251	0.1251	0.3069
Top-5	0.4359	**0.4847**	0.1548	0.1254	0.1257	0.317
Top-6	0.4572	**0.5091**	0.1585	0.1302	0.1307	0.3212
Top-7	0.4825	**0.5273**	0.2508	0.2202	0.2205	0.3621
Top-8	0.5018	**0.5447**	0.2508	0.2202	0.2205	0.3933
Top-9	0.5195	**0.5627**	0.2508	0.2202	0.2205	0.4084
Top-10	0.54	**0.5759**	0.2508	0.2202	0.2205	0.4087

Table 5: The accuracy of OpenOffice using different classifiers.

OpenOffice	NB	NBM	SVM	KNN	RT	J48
Top-1	0.0969	**0.182**	0.0044	0.076	0.0716	0.132
Top-2	0.1607	**0.275**	0.0059	0.0775	0.0731	0.1984
Top-3	0.219	**0.3211**	0.0228	0.0935	0.091	0.2247
Top-4	0.249	**0.3507**	0.0434	0.1118	0.1087	0.2401
Top-5	0.2801	**0.3775**	0.0434	0.1128	0.1089	0.2583
Top-6	0.3054	**0.4008**	0.0491	0.117	0.1143	0.2666
Top-7	0.3274	**0.4209**	0.0505	0.1185	0.1155	0.2715
Top-8	0.3434	**0.4437**	0.0515	0.1221	0.1192	0.2745
Top-9	0.3603	**0.4636**	0.1018	0.1575	0.1577	0.2897
Top-10	0.3794	**0.4839**	0.1106	0.1594	0.1602	0.3066

Table 6: The accuracy of GCC using different classifiers.

GCC	NB	NBM	SVM	KNN	RT	J48
Top-1	0.1598	**0.3464**	0.3295	0.1286	0.1188	0.1821
Top-2	0.2464	**0.525**	0.3313	0.1304	0.1205	0.2759
Top-3	0.3125	**0.5786**	0.3313	0.1304	0.1205	0.2955
Top-4	0.3589	**0.6036**	0.3313	0.1304	0.1205	0.3009
Top-5	0.4	**0.6241**	0.3313	0.1304	0.1205	0.3027
Top-6	0.4259	**0.6464**	0.3321	0.1313	0.1214	0.3045
Top-7	0.4473	**0.6607**	0.3321	0.1313	0.1214	0.3045
Top-8	0.4679	**0.6705**	0.3402	0.1393	0.1295	0.3089
Top-9	0.4884	**0.6804**	0.3402	0.1393	0.1295	0.3107
Top-10	0.5027	**0.6884**	0.342	0.1411	0.1304	0.3116

KNN, RT, and J48). The Top-1 accuracy of NBM is 22.76% higher than that of RT and the Top-10 accuracy is 55.8% higher than RT on GCC.

Therefore, we conclude that NBM is the most effective algorithm among all methods. Thus, we use NBM as a reliable classification algorithm in subsequent experiments.

4.3.2. RQ2: Can Our Proposed FS and IS Methods Perform Better Than Traditional FS and IS Methods in terms of Data Reduction? In this part, we conduct four experiments on four open-source bug repositories, respectively. For each bug repository, we consider the data reduction of the original bug repository to specific data size, using feature selection or instance selection methods as benchmarks. We choose three traditional feature selection methods (IG, CHI, and SU) and three traditional instance selection methods (CNN, ENN, and ICF) as benchmarks to calculate Top-*k* accuracies, which reduce the original bug repository to the same data size as benchmarks. The accuracies of our proposed FS and IS heuristic search method with the same size of data reduction are also obtained. The column with the best effect is shown in bold in Tables 7–10.

Table 7 shows that our proposed FS method has 26.01% accuracy of Top-10 and the IS method has 30.44% accuracy of Top-10. Compared with the original data reduction and the traditional FS and IS methods, our proposed methods perform better for Mozilla. Meanwhile, the accuracy of IS is 4.43% higher than that of FS. The result for OpenOffice (Table 9) is quite similar to Mozilla. Our proposed FS method has the highest accuracy of 46.26% among feature selection methods, and the accuracy of our proposed IS method is 53.76%, which is also the best in IS approaches. The accuracy of IS is 7.5% higher than FS. Table 10 shows similar experimental results for GCC: the highest accuracy is 84.51% for our proposed FS method (in FS methods) and the highest accuracy is 83.69% for our proposed IS approach (in IS approaches). The only difference from the former is that FS is 0.82% higher than IS, which indicates that FS is more suitable for data reduction on GCC. However, there are many differences in NetBeans results (Table 8): our FS method is still the best with 54.14% accuracy among FS methods. However, a traditional IS approach called ENN has the highest accuracy of 59.88% in IS methods. Our IS method appears to overfit. The accuracy is rather high on the validation set (61%) but is reduced on the test set. A possible reason is that the developer flow of NetBeans may have been frequent recently. Moreover, NetBeans is not greatly affected by text information. In the RQ5, we verify this reason further.

We consider that when reducing to the same data size, the heuristic search method is generally better than the traditional IS and FS search method. The main reasons are as follows:

(1) The traditional FS methods and traditional IS methods are based on the overall training set. However, the effect of the bug report is time-sensitive. Outdated bug reports will add redundancy and noise, which may affect the classification accuracy. Traditional IS and FS methods cannot effectively eliminate noisy and redundant data.

(2) In the 7 : 1 : 2 search strategy, our proposed approach focuses more on extracting features related to recent bug reports, which is helpful for removing noisy and redundant data.

4.3.3. RQ3: Are the Accuracies of Our Proposed FS + IS, Best FS, and Best IS Methods Improved Comparing with the FS + IS Results of Xuan? If Our Results Are Improved, Which One Is the Most Effective among Our Three Methods? Based on the experimental results in RQ2, we reproduce the FS -> IS (IG + ICF) and IS -> FS (ICF + IG) experiments of Xuan and gain the Top-*k* accuracy on four bug repositories (Mozilla, NetBeans, OpenOffice, and GCC). In addition, we also combine our proposed FS method with the IS method (FS + IS) to reduce data of four bug repositories. The comparative results are shown in Tables 11–14. Compared with FS -> IS and IS -> FS, our proposed heuristic search method is better when reducing to the same data size.

Our FS + IS method has the best effect for Mozilla in Table 11 and NetBeans in Table 12. The accuracies are 29.44% and 56.07%, respectively. The Top-10 accuracy of FS + IS is 6.78% higher than that of IS -> FS on Mozilla and 16.42% higher than that of IS -> FS on NetBeans. We think another reason is that the heuristic search FS + IS focuses on the extraction of features related to recent bug reports. Thus, the amount of information loss is relatively small, and the ability to denoise and exclude redundancy is stronger. For OpenOffice in Table 12, the best accuracy 53.76% belongs to our IS method, which is 7.73% higher than IS -> FS and 45.57% higher than FS -> IS. On the contrary, we can find that our FS approach, which has 84.51% accuracy, is the most effective on GCC in Table 14. The Top-10 accuracy of FS is 9.23% higher than that of IS -> FS method on GCC.

We conclude that our proposed approaches are overall more effective than the results of Xuan. For four different bug repositories, FS + IS performs well on Mozilla and NetBeans. However, OpenOffice has the optimal effect with IS and the best of GCC is FS.

4.3.4. RQ4: How about the Data Reduction Effect of Our Proposed FS, IS, and FS + IS Methods? In this experiment, we use our proposed FS, IS, and FS + IS methods based on particle swarm optimization to obtain data reduction rates and study the effect of our approaches on four open bug repositories (Mozilla, NetBeans, OpenOffice, and GCC). The results are detailed in Tables 15–18 (data reduction degree of FS or IS and overall reduction accuracy refer to the ratio of the reserved instances or features to the original quantity). Here, the "original information" represents the dataset which is divided into 70% training dataset, 10% validation dataset, and 20% test dataset. Our approach finds primary features and instances using differential evolution algorithm with training set. Then, they are merged to get a new training set "original information (7 + 1)."

Table 7: Comparison of different feature selection and instance selection methods on Mozilla.

Mozilla	Original	IG	CHI	SU	FS
Top-1	0.0906	0.0723	0.0723	0.0723	0.0868
Top-2	0.1463	0.1155	0.1155	0.1155	0.13
Top-3	0.1833	0.1792	0.1792	0.1792	0.1578
Top-4	0.2045	0.1944	0.1944	0.1944	0.178
Top-5	0.2233	0.2072	0.2072	0.2072	0.1956
Top-6	0.2404	0.2196	0.2196	0.2196	0.2104
Top-7	0.2551	0.23	0.23	0.23	0.2287
Top-8	0.2669	0.2397	0.2397	0.2397	0.2424
Top-9	0.278	0.2487	0.2487	0.2487	0.2518
Top-10	0.2889	0.2563	0.2563	0.2563	0.2601
Mozilla	*Original*	*CNN*	*ENN*	*ICF*	*IS*
Top-1	0.0906	0.0152	0.0218	0.0225	**0.1208**
Top-2	0.1463	0.0218	0.0402	0.0506	**0.1884**
Top-3	0.1833	0.0277	0.0568	0.0654	**0.2105**
Top-4	0.2045	0.0377	0.0724	0.0841	**0.2351**
Top-5	0.2233	0.044	0.0845	0.1049	**0.2528**
Top-6	0.2404	0.0485	0.097	0.1195	**0.2639**
Top-7	0.2551	0.054	0.1118	0.1281	**0.276**
Top-8	0.2669	0.0578	0.124	0.1423	**0.2843**
Top-9	0.278	0.063	0.133	0.1569	**0.2933**
Top-10	0.2889	0.0661	0.1461	0.1676	**0.3044**

Table 8: Comparison of different feature selection and instance selection methods on NetBeans.

NetBeans	Original	IG	CHI	SU	FS
Top-1	0.2917	0.1778	0.1778	0.1778	0.2532
Top-2	0.3972	0.2571	0.2571	0.2571	0.3524
Top-3	0.4311	0.3188	0.3188	0.3188	0.4012
Top-4	0.4612	0.3551	0.3551	0.3551	0.4329
Top-5	0.4847	0.3877	0.3877	0.3877	0.4576
Top-6	0.5091	0.417	0.417	0.417	0.4732
Top-7	0.5273	0.4383	0.4383	0.4383	0.4911
Top-8	0.5447	0.4574	0.4574	0.4574	0.5072
Top-9	0.5627	0.4732	0.4732	0.4732	0.5251
Top-10	0.5759	0.4881	0.4881	0.4881	0.5414
NetBeans	*Original*	*CNN*	*ENN*	*ICF*	*IS*
Top-1	0.2917	0.0378	**0.225**	0.1863	0.2993
Top-2	0.3972	0.0614	**0.3384**	0.2706	0.3965
Top-3	0.4311	0.0867	**0.4038**	0.3353	0.4418
Top-4	0.4612	0.1084	**0.4512**	0.3828	0.4661
Top-5	0.4847	0.1268	**0.4928**	0.4203	0.4812
Top-6	0.5091	0.1405	**0.5216**	0.4468	0.4923
Top-7	0.5273	0.1554	**0.5462**	0.4689	0.5058
Top-8	0.5447	0.1707	**0.5662**	0.4883	0.5207
Top-9	0.5627	0.183	**0.5821**	0.5053	0.5299
Top-10	0.5759	0.1974	**0.5988**	0.5171	0.5409

According to our results, the reduction degrees of FS + IS are 96.05% on Mozilla, 94.43% on GCC, and 97.08% on OpenOffice, which can substantially reduce the data dimensions. However, FS + IS does not work best on any bug repository. For Mozilla, FS has the highest reduction degree of 97.21%, which is 1.16% higher than FS + IS. However, the reduction degrees of FS are smaller than for the FS + IS method on OpenOffice by 2.16% and GCC by 16.19%. The IS method also has a low degree of reduction on Mozilla, OpenOffice, and GCC. Nevertheless, it can reach 50% on OpenOffice. Moreover, our three methods do not perform well on NetBeans. If we compare the performance of three methods according to the degree of reduction, the result is FS + IS > FS > IS.

The degree of data reduction by FS, IS, and FS + IS based on the DE method is quite large. Compared with the traditional methods of removing invalid bug reports by name, the heuristic search based on the DE method, which can automatically delete invalid bug reports, is more effective in denoising and excluding redundancy. However, according to the previous experiment, FS + IS cannot achieve a further improvement in the accuracy of the NBM classifier compared to separate FS and IS methods. We consider the reason may be the excessive reduction, which can result in serious information loss. Therefore, we conclude that the data reduction of FS, IS, and FS + IS based on the DE method is very effective.

Table 9: Comparison of different feature selection and instance selection methods on OpenOffice.

OpenOffice	Original	IG	CHI	SU	FS
Top-1	0.182	0.0919	0.0919	0.0919	0.1458
Top-2	0.275	0.1563	0.1563	0.1563	0.2155
Top-3	0.3211	0.2123	0.2123	0.2123	0.266
Top-4	0.3507	0.2462	0.2462	0.2462	0.3005
Top-5	0.3775	0.2618	0.2618	0.2618	0.3295
Top-6	0.4008	0.2904	0.2904	0.2904	0.3569
Top-7	0.4209	0.3079	0.3079	0.3079	0.3802
Top-8	0.4437	0.3257	0.3257	0.3257	0.4078
Top-9	0.4636	0.3437	0.3437	0.3437	0.4368
Top-10	0.4839	0.353	0.353	0.353	0.4624
OpenOffice	*Original*	*CNN*	*ENN*	*ICF*	*IS*
Top-1	0.182	0.0289	0.0289	0.0289	**0.1477**
Top-2	0.275	0.0491	0.0491	0.0491	**0.2092**
Top-3	0.3211	0.0735	0.0735	0.0735	**0.2447**
Top-4	0.3507	0.1017	0.1017	0.1017	**0.2723**
Top-5	0.3775	0.1266	0.1266	0.1266	**0.2971**
Top-6	0.4008	0.1479	0.1479	0.1479	**0.3202**
Top-7	0.4209	0.1673	0.1673	0.1673	**0.3844**
Top-8	0.4437	0.1815	0.1815	0.1815	**0.4599**
Top-9	0.4636	0.1992	0.1992	0.1992	**0.5088**
Top-10	0.4839	0.2197	0.2197	0.2197	**0.5376**

Table 10: Comparison of different feature selection and instance selection methods on GCC.

GCC	Original	IG	CHI	SU	FS
Top-1	0.3464	0.1832	0.1832	0.1832	**0.5115**
Top-2	0.525	0.3109	0.3109	0.3109	**0.6724**
Top-3	0.5786	0.4151	0.4151	0.4151	**0.7677**
Top-4	0.6036	0.4899	0.4899	0.4899	**0.7863**
Top-5	0.6241	0.548	0.548	0.548	**0.799**
Top-6	0.6464	0.5912	0.5912	0.5912	**0.8098**
Top-7	0.6607	0.6281	0.6281	0.6281	**0.8224**
Top-8	0.6705	0.6608	0.6608	0.6608	**0.8302**
Top-9	0.6804	0.6817	0.6817	0.6817	**0.8403**
Top-10	0.6884	0.7085	0.7085	0.7085	**0.8451**
GCC	*Original*	*CNN*	*ENN*	*ICF*	*IS*
Top-1	0.3464	0.1832	0.0923	0.1906	0.519
Top-2	0.525	0.3109	0.1869	0.3433	0.6798
Top-3	0.5786	0.4151	0.2695	0.4564	0.7677
Top-4	0.6036	0.4899	0.3325	0.5398	0.7822
Top-5	0.6241	0.548	0.3924	0.5961	0.7949
Top-6	0.6464	0.5912	0.4464	0.6448	0.8016
Top-7	0.6607	0.6281	0.4989	0.6839	0.8179
Top-8	0.6705	0.6608	0.5391	0.7118	0.8232
Top-9	0.6804	0.6817	0.5786	0.7375	0.8306
Top-10	0.6884	0.7085	0.6158	0.7587	0.8369

4.3.5. RQ5: How Can the Developer's Activities Affect the Optimal List of Recommended Developers? In this part, we add the developer engagement level into our experiment to calculate the accuracy on four bug repositories (Mozilla, NetBeans, OpenOffice, and GCC) from Top-1 to Top-10. The results are shown in Tables 19–22 (in Tables 19–22, we use "Without developer engagement" to describe not considering the developer engagement, and "With developer engagement" refers to the consideration of developer engagement).

According to the results, NBM's accuracy improves higher on all feature extraction schemes except for Mozilla. The highest classification accuracies on the four bug repositories have changed compared with accuracies without considering developer engagement. The improved percent of FS + IS can reach up to 2.63% on Mozilla. For NetBeans, the overall accuracies of FS, IS, and FS + IS have substantially improved. The Top-10 accuracies of FS, IS, and FS + IS have gone up 5.64%, 2.47%, and 2.91%, respectively, which verifies our thinking in RQ2. The results indicate that the accuracy of NetBeans with consideration of developer engagement has substantial improvement, which explains that the developer flow is frequent in NetBeans. Table 21 shows that the accuracies of the three proposed methods improve on OpenOffice, and FS has a good improved range of 9.33%. Similarly, considering developer engagement, the

Table 11: Comparison experiment using FS, IS, and FS + IS on Mozilla.

Mozilla	Original	FS -> IS (IG + ICF)	IS -> FS (ICF + IG)	FS	IS	FS + IS
Top-1	0.0906	0.0059	0.0654	0.0868	**0.1208**	0.1001
Top-2	0.1463	0.01	0.0879	0.13	**0.1884**	0.1488
Top-3	0.1833	0.0135	0.1083	0.1578	**0.2105**	0.1756
Top-4	0.2045	0.0197	0.1602	0.178	**0.2351**	0.2072
Top-5	0.2233	0.0214	0.1736	0.1956	**0.2528**	0.2388
Top-6	0.2404	0.0218	0.1916	0.2104	**0.2639**	0.2505
Top-7	0.2551	0.0256	0.1985	0.2287	**0.276**	0.2645
Top-8	0.2669	0.0284	0.21	0.2424	**0.2843**	0.2778
Top-9	0.278	0.0301	0.2269	0.2518	**0.2933**	0.2896
Top-10	0.2889	0.0308	0.2383	0.2601	**0.3044**	0.2944

Table 12: Comparison experiment using FS, IS, and FS + IS on NetBeans.

NetBeans	Original	FS -> IS (IG + ICF)	IS -> FS (ICF + IG)	FS	IS	FS + IS
Top-1	0.2917	0.128	0.1148	0.2532	0.2945	**0.261**
Top-2	0.3972	0.1847	0.1827	0.3524	0.3951	**0.3402**
Top-3	0.4311	0.2279	0.2458	0.4012	0.4529	**0.3974**
Top-4	0.4612	0.2491	0.2786	0.4329	0.4708	**0.4429**
Top-5	0.4847	0.2687	0.3066	0.4576	0.4842	**0.475**
Top-6	0.5091	0.2864	0.3348	0.4732	0.4923	**0.4946**
Top-7	0.5273	0.3015	0.3532	0.4911	0.5019	**0.5133**
Top-8	0.5447	0.314	0.3683	0.5072	0.5176	**0.5325**
Top-9	0.5627	0.3292	0.3819	0.5251	0.5297	**0.5426**
Top-10	0.5759	0.3421	0.3965	0.5414	0.5442	**0.5607**

Table 13: Comparison experiment using FS, IS, and FS + IS on OpenOffice.

OpenOffice	Original	FS -> IS (IG + ICF)	IS -> FS (ICF + IG)	FS	IS	FS + IS
Top-1	0.182	0.0084	0.0313	0.1458	**0.1477**	0.1396
Top-2	0.275	0.0244	0.075	0.2155	**0.2092**	0.2022
Top-3	0.3211	0.0311	0.1061	0.266	**0.2447**	0.2516
Top-4	0.3507	0.0362	0.1554	0.3005	**0.2723**	0.2843
Top-5	0.3775	0.0411	0.1796	0.3295	**0.2971**	0.3122
Top-6	0.4008	0.0642	0.2025	0.3569	**0.3202**	0.3431
Top-7	0.4209	0.0708	0.4196	0.3802	**0.3844**	0.3681
Top-8	0.4437	0.075	0.4427	0.4078	**0.4599**	0.3955
Top-9	0.4636	0.0806	0.4547	0.4368	**0.5088**	0.424
Top-10	0.4839	0.0819	0.4603	0.4624	**0.5376**	0.454

Table 14: Comparison experiment using FS, IS, and FS + IS on GCC.

GCC	Original	FS -> IS (IG + ICF)	IS -> FS (ICF + IG)	FS	IS	FS + IS
Top-1	0.3464	0.2878	0.3414	**0.5115**	0.519	0.4749
Top-2	0.525	0.4494	0.4758	**0.6724**	0.6798	0.6356
Top-3	0.5786	0.5551	0.5566	**0.7677**	0.7677	0.7648
Top-4	0.6036	0.6288	0.6095	**0.7863**	0.7822	0.7884
Top-5	0.6241	0.6742	0.6504	**0.799**	0.7949	0.7989
Top-6	0.6464	0.7096	0.6768	**0.8098**	0.8016	0.8112
Top-7	0.6607	0.736	0.6992	**0.8224**	0.8179	0.8213
Top-8	0.6705	0.7543	0.7148	**0.8302**	0.8232	0.8281
Top-9	0.6804	0.7755	0.736	**0.8403**	0.8306	0.8345
Top-10	0.6884	0.7893	0.7528	**0.8451**	0.8369	0.8393

accuracy of GCC is going up slightly. FS + IS has the biggest improvement of 1.05% among the three proposed approaches.

Meanwhile, the study of developer engagement alleviates the overfitting problem effectively for feature extraction on NetBeans. Moreover, the accuracy of NBM is significantly

Table 15: Reduced data size comparison of FS, IS and FS + IS on Mozilla.

Mozilla	FS + IS	FS	IS	Original information (7 + 1)
IS number	5272	11291	4679	15035
FS number	3528	1163	14686	31317
Data reduction degree of IS	0.3506	0.7510	0.3112	
Data reduction degree of FS	0.1127	0.0371	0.4689	
Overall reduction accuracy (row ∗ column)	0.0395	0.0279	0.1459	

Table 16: Reduced data size comparison of FS, IS, and FS + IS on NetBeans.

NetBeans	FS + IS	FS	IS	Original information (7 + 1)
IS number	16919	16941	5944	17231
FS number	9041	9964	24324	34413
Data reduction degree of IS	0.9819	0.9832	0.3450	
Data reduction degree of FS	0.2627	0.2895	0.7068	
Overall reduction degree (row ∗ column)	0.2580	0.2847	0.2438	

Table 17: Reduced data size comparison of FS, IS, and FS + IS on OpenOffice.

OpenOffice	FS + IS	FS	IS	Original information (7 + 1)
IS number	12527	17941	18001	18781
FS number	1934	2349	24175	44248
Data reduction degree of IS	0.6670	0.9553	0.9585	
Data reduction degree of FS	0.0437	0.0531	0.5464	
Overall reduction degree (row ∗ column)	0.0292	0.0507	0.5237	

Table 18: Reduced data size comparison of FS, IS, and FS + IS on GCC.

GCC	FS + IS	FS	IS	Original information (7 + 1)
IS number	I7493	10742	6394	11169
FS number	3842	10468	26170	46251
Data reduction degree of IS	0.6709	0.9618	0.5725	
Data reduction degree of FS	0.0831	0.2263	0.5658	
Overall reduction degree (row ∗ column)	0.0557	0.2177	0.3239	

Table 19: Accuracy of Top-*k* considering developer engagement and without considering engagement on Mozilla.

Mozilla	Without developer engagement	With developer engagement
FS		
Top-1	0.0868	0.0868
Top-2	0.13	0.13
Top-3	0.1578	0.1578
Top-4	0.178	0.178
Top-5	0.1956	0.1956
Top-6	0.2104	0.2104
Top-7	0.2287	0.2287
Top-8	0.2424	0.2424
Top-9	0.2518	0.2518
Top-10	0.2601	0.2601
IS		
Top-1	0.1208	**0.1236**
Top-2	0.1884	**0.1911**
Top-3	0.2105	**0.2123**
Top-4	0.2351	**0.2382**
Top-5	0.2528	**0.2548**
Top-6	0.2639	**0.2715**
Top-7	0.276	**0.2777**
Top-8	0.2843	**0.2881**
Top-9	0.2933	**0.3012**

Table 19: Continued.

Mozilla	Without developer engagement	With developer engagement
Top-10	0.3044	**0.3096**
FS + IS		
Top-1	0.1001	**0.1071**
Top-2	0.1488	**0.159**
Top-3	0.1756	**0.204**
Top-4	0.2072	**0.2398**
Top-5	0.2388	**0.2607**
Top-6	0.2505	**0.2778**
Top-7	0.2645	**0.2853**
Top-8	0.2778	**0.3003**
Top-9	0.2896	**0.3116**
Top-10	0.2944	**0.3207**

Table 20: Accuracy of Top-*k* considering engagement and without considering engagement on NetBeans.

NetBeans	Without developer engagement	With developer engagement
FS		
Top-1	0.2532	**0.2521**
Top-2	0.3524	**0.3578**
Top-3	0.4012	**0.4173**
Top-4	0.4329	**0.4496**
Top-5	0.4576	**0.4734**
Top-6	0.4732	**0.4919**
Top-7	0.4911	**0.523**
Top-8	0.5072	**0.5518**
Top-9	0.5251	**0.5815**
Top-10	0.5414	**0.6342**
IS		
Top-1	0.2945	**0.2995**
Top-2	0.3951	**0.3972**
Top-3	0.4529	**0.4423**
Top-4	0.4708	**0.4671**
Top-5	0.4842	**0.4824**
Top-6	0.4923	**0.4937**
Top-7	0.5019	**0.51**
Top-8	0.5176	**0.5277**
Top-9	0.5297	**0.5544**
Top-10	0.5442	**0.628**
FS + IS		
Top-1	0.261	**0.2427**
Top-2	0.3402	**0.3551**
Top-3	0.3974	**0.4073**
Top-4	0.4429	**0.4406**
Top-5	0.475	**0.4649**
Top-6	0.4946	**0.4876**
Top-7	0.5133	**0.5197**
Top-8	0.5325	**0.5452**
Top-9	0.5426	**0.5717**
Top-10	0.5607	**0.6246**

higher than the original accuracy. Because the accuracy of the feature extraction scheme of FS + IS has been obviously improved, we conclude that the introduction of developer engagement can successfully compensate for the problem of information loss caused by FS + IS.

We find that the accuracy on the test set using the NBM classifier generally presents a trend of increasing first, then lowering, and finally, flattening with increasing *N*. After a careful analysis, we find that the two kinds of information compete with the growth of *N*. One is the effective information related to the test set, and the other is the noise. This confrontation relationship leads to the accuracy increasing first and then decreasing with the growth of *N*. For the data reduction of Mozilla_total's DE_FS, the accuracy is always at a lower degree with *N* changing. It is because the flow of Mozilla_total staff changes frequently with time, which causes the noisy and redundant data to grow and fragment. However, we also found that the dataset generated by

Table 21: Accuracy of Top-*k* considering engagement and without considering engagement on OpenOffice.

OpenOffice	Without developer engagement	With developer engagement
FS		
Top-1	0.1458	**0.1703**
Top-2	0.2155	**0.2409**
Top-3	0.266	**0.2861**
Top-4	0.3005	**0.3208**
Top-5	0.3295	**0.3562**
Top-6	0.3569	**0.3951**
Top-7	0.3802	**0.4357**
Top-8	0.4078	**0.4889**
Top-9	0.4368	**0.5256**
Top-10	0.4624	**0.5557**
IS		
Top-1	0.1477	**0.1646**
Top-2	0.2092	**0.2247**
Top-3	0.2447	**0.2636**
Top-4	0.2723	**0.3002**
Top-5	0.2971	**0.3724**
Top-6	0.3202	**0.455**
Top-7	0.3844	**0.5059**
Top-8	0.4599	**0.5392**
Top-9	0.5088	**0.5574**
Top-10	0.5376	**0.5741**
FS + IS		
Top-1	0.1396	**0.1579**
Top-2	0.2022	**0.2305**
Top-3	0.2516	**0.2717**
Top-4	0.2843	**0.3106**
Top-5	0.3122	**0.3434**
Top-6	0.3431	**0.381**
Top-7	0.3681	**0.4197**
Top-8	0.3955	**0.4656**
Top-9	0.424	**0.5023**
Top-10	0.454	**0.5315**

Table 22: Accuracy of Top-*k* considering engagement and without considering engagement on GCC.

GCC	Without developer engagement	With developer engagement
FS		
Top-1	0.5115	**0.5115**
Top-2	0.6724	**0.6724**
Top-3	0.7677	**0.7673**
Top-4	0.7863	**0.7863**
Top-5	0.799	**0.7986**
Top-6	0.8098	**0.8094**
Top-7	0.8224	**0.8235**
Top-8	0.8302	**0.8347**
Top-9	0.8403	**0.8436**
Top-10	0.8451	**0.8507**
IS		
Top-1	0.519	**0.519**
Top-2	0.6798	**0.6798**
Top-3	0.7677	**0.7681**
Top-4	0.7822	**0.7807**
Top-5	0.7949	**0.7937**
Top-6	0.8016	**0.8116**
Top-7	0.8179	**0.8213**
Top-8	0.8232	**0.8284**
Top-9	0.8306	**0.838**
Top-10	0.8369	**0.847**

Table 22: Continued.

GCC	Without developer engagement	With developer engagement
FS + IS		
Top-1	0.4749	**0.4753**
Top-2	0.6356	**0.6352**
Top-3	0.7648	**0.7652**
Top-4	0.7884	**0.788**
Top-5	0.7989	**0.7996**
Top-6	0.8112	**0.8131**
Top-7	0.8213	**0.8213**
Top-8	0.8281	**0.8303**
Top-9	0.8345	**0.8382**
Top-10	0.8393	**0.8498**

Table 23: Distance corresponding to the peak of the engagement accuracy.

Bug repositories	FS + IS	Total number	FS	Total number	IS	Total number
GCC	231	7493	1095	10742	195	6394
OpenOffice	162	12527	176	17941	288	18001
NetBeans	502	5272	11291	11291	596	4679
Mozilla	464	16941	464	16941	460	5944

Mozilla_total's DE_IS scheme performs well after adding developer engagement, which indicates that the denoising ability of IS on the Mozilla_total is better than that of the FS approach.

We conclude that the introduction of developer engagement can effectively improve the classification accuracy of NBM. Moreover, it significantly alleviates the overfitting phenomenon which happens when a model learns the details and noise in the training data to the extent that it negatively impacts the performance of the model on unseen data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact NBM's ability to generalize. In our improved NBM classification, removal of redundant features from bug trial dataset can prevent overfitting. In addition, the developer engagement effectively compensates for information loss caused by the FS + IS method and substantially increases the accuracy. Meanwhile, compared with the overall dataset, the optimal reference range is smaller and easier to implement if considering developer engagement.

In Table 23, we analyze the peak of engagement, which can explain the frequency of developers' flow in different bug repositories. A greater distance between the peak and the average means a higher frequency of recent personnel movements. We can learn that the developers of Mozilla and OpenOffice flow more frequently than GCC and NetBeans.

5. Conclusion and Future Work

In this paper, we propose a new bug triage method for recommending suitable developers to fix newly reported bugs. To solve the problem of small search range and neglecting the chronological order in the traditional bug triage method, we improve the existing heuristic search method and expand the search scope further based on the chronological order of bug reports. We find that developer engagement has an impact on bug triage; therefore, in addition to the text information provided in the bug report, we consider the developer's product information to recommend the best developer for the new bug report. We use FS, IS, and FS + IS to verify our approach on four bug repositories: GCC, OpenOffice, Mozilla, and NetBeans. The results show that the method proposed in this paper is more effective than the previous methods. In future work, we plan to verify our approach using more bug repositories. Moreover, we plan to apply our method to additional software projects.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (nos. 61672122, 61602077, 61771087, 51879027, 51579024, 71831002, and 61902050), Program for Innovative Research Team in University of Ministry of Education of China (no. IRT 17R13), the Fundamental Research Funds for the Central Universities (nos. 3132019501 and 3132019502), and the Next-Generation Internet Innovation Project of CERNET (nos. NGII20181203, NGII20181205, and NGII201810128).

References

[1] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss, "What makes a good bug report?"

IEEE Transactions on Software Engineering, vol. 36, no. 5, pp. 618–643, 2010.

[2] J. Anvik, L. Hiew, and G. C. Murphy, “Who should fix this bug?” in *Proceedings of the 28th International Conference on Software Engineering*, pp. 361–370, Shanghai, China, May 2006.

[3] W. Deng, H. Zhao, L. Zou, G. Li, X. Yang, and D. Wu, “A novel collaborative optimization algorithm in solving complex optimization problems,” *Soft Computing*, vol. 21, no. 15, pp. 4387–4398, 2017.

[4] S. Guo, R. Chen, H. Li, T. Zhang, and Y. Liu, “Identify severity bug report with distribution imbalance by CR-SMOTE and ELM,” *International Journal of Software Engineering and Knowledge Engineering*, vol. 29, no. 2, pp. 139–175, 2019.

[5] J. Anvik and G. C. Murphy, “Reducing the effort of bug report triage,” *ACM Transactions on Software Engineering and Methodology*, vol. 20, no. 3, pp. 1–35, 2011.

[6] J. Xuan, H. Jiang, Z. Ren, and W. Zou, “Developer prioritization in bug repositories,” in *Proceedings of the 2012 34th International Conference on Software Engineering (ICSE)*, IEEE, Zurich, Switzerland, pp. 25–35, June 2012.

[7] H. Li, G. Gao, R. Chen, X. Ge, S. Guo, and L.-Y. Hao, “The influence ranking for testers in bug tracking systems,” *International Journal of Software Engineering and Knowledge Engineering*, vol. 29, no. 1, pp. 93–113, 2019.

[8] W. Pan and C. Chai, “Structure-aware mashup service clustering for cloud-based internet of things using genetic algorithm based clustering algorithm,” *Future Generation Computer Systems*, vol. 87, pp. 267–277, 2018.

[9] W. Pan, J. Dong, K. Liu, and J. Wang, “Topology and topic-aware service clustering,” *International Journal of Web Services Research*, vol. 15, no. 3, pp. 18–37, 2018.

[10] https://www.mantisbt.org/.

[11] https://www.bugzilla.org/.

[12] https://www.atlassian.com/software/jira.

[13] S. Guo, Y. Liu, R. Chen, X. Sun, and X. Wang, “Improved SMOTE algorithm to deal with imbalanced activity classes in smart homes,” *Neural Processing Letters*, vol. 50, no. 2, pp. 1503–1526, 2019.

[14] S. Guo, R. Chen, M. Wei, H. Li, and Y. Liu, “Ensemble data reduction techniques and multi-RSMOTE via fuzzy integral for bug report classification,” *IEEE Access*, vol. 6, pp. 45934–45950, 2018.

[15] W. Deng, J. Xu, and H. Zhao, “An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem,” *IEEE Access*, vol. 7, pp. 20281–20292, 2019.

[16] M. S. Zanetti, I. Scholtes, C. J. Tessone, and F. Schweitzer, “Categorizing bugs with social networks: a case study on four open source software communities,” in *Proceedings of the 2013 35th International Conference on Software Engineering (ICSE)*, IEEE, San Francisco, CA, USA, pp. 1032–1041, 2013.

[17] H. Hu, H. Zhang, J. Xuan, and W. Sun, “Effective bug triage based on historical bug-fix information,” in *Proceedings of the 2014 IEEE 25th International Symposium on Software Reliability Engineering*, IEEE, Naples, Italy, pp. 122–132, November 2014.

[18] W. Pan, B. Song, K. Li, and K. Zhang, “Identifying key classes in object-oriented software using generalizedk-core decomposition,” *Future Generation Computer Systems*, vol. 81, pp. 188–202, 2018.

[19] H. Jiang, L. Nie, Z. Sun et al., “ROSF: leveraging information retrieval and supervised learning for recommending code snippets,” *IEEE Transactions on Services Computing*, vol. 12, no. 1, pp. 34–46, 2019.

[20] R. Chen, S.-K. Guo, X.-Z. Wang, and T.-L. Zhang, “Fusion of multi-RSMOTE with fuzzy integral to classify bug reports with an imbalanced distribution,” *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 12, pp. 2406–2420, 2019.

[21] G. Jeong, S. Kim, and T. Zimmermann, “Improving bug triage with bug tossing graphs,” in *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pp. 111–120, New York, NY, USA, August 2009.

[22] P. Bhattacharya and I. Neamtiu, “Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging,” in *Proceedings of the 2010 IEEE International Conference on Software Maintenance*, IEEE, Washington, DC, USA, pp. 1–10, 2010.

[23] A. Tamrawi, T. T. Nguyen, J. M. Al-Kofahi, and T. N. Nguyen, “Fuzzy set and cache-based approach for bug triaging,” in *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, pp. 365–375, New York, NY, USA, 2011.

[24] W. Pan, B. Li, J. Liu, Y. Ma, and B. Hu, “Analyzing the structure of Java software systems by weightedK-core decomposition,” *Future Generation Computer Systems*, vol. 83, pp. 431–444, 2018.

[25] K. Somasundaram and G. C. Murphy, “Automatic categorization of bug reports using latent dirichlet allocation,” in *Proceedings of the 5th India Software Engineering Conference*, pp. 125–130, Kanpur, India, February 2012.

[26] X. Xia, D. Lo, Y. Ding, J. M. Al-Kofahi, T. N. Nguyen, and X. Wang, “Improving automated bug triaging with specialized topic model,” *IEEE Transactions on Software Engineering*, vol. 43, no. 3, pp. 272–297, 2016.

[27] G. Yang, T. Zhang, and B. Lee, “Towards semi-automatic bug triage and severity prediction based on topic model and multi-feature of bug reports,” in *Proceedings of the 2014 IEEE 38th Annual Computer Software and Applications Conference*, IEEE, Vasteras, Sweden, pp. 97–106, July 2014.

[28] W. Deng, S. Zhang, H. Zhao, and X. Yang, “A novel fault diagnosis method based on integrating empirical wavelet transform and fuzzy entropy for motor bearing,” *IEEE Access*, vol. 6, pp. 35042–35056, 2018.

[29] W. Deng, H. Zhao, X. Yang, J. Xiong, M. Sun, and B. Li, “Study on an improved adaptive PSO algorithm for solving multi-objective gate assignment,” *Applied Soft Computing*, vol. 59, pp. 288–302, 2017.

[30] H. Jiang, X. Li, Z. Ren, J. Xuan, and Z. Jin, “Toward better summarizing bug reports with crowdsourcing elicited attributes,” *IEEE Transactions on Reliability*, vol. 68, no. 1, pp. 2–22, 2019.

[31] H. Zhao, M. Sun, W. Deng, and X. Yang, “A new feature extraction method based on EEMD and multi-scale fuzzy entropy for motor bearing,” *Entropy*, vol. 19, no. 1, p. 14, 2016.

[32] J. Xuan, H. Jiang, Y. Hu et al., “Towards effective bug triage with software data reduction techniques,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 1, pp. 264–280, 2015.

[33] H. Zhao, R. Yao, L. Xu, Y. Yuan, G. Li, and W. Deng, “Study on a novel fault damage degree identification method using high-order differential mathematical morphology gradient spectrum entropy,” *Entropy*, vol. 20, no. 9, p. 682, 2018.

[34] Y. Xiang, W. Pan, H. Jiang, Y. Zhu, and H. Li, “Measuring software modularity based on software networks,” *Entropy*, vol. 21, no. 4, p. 344, 2019.

[35] H. Naguib, N. Narayan, B. Brügge, and D. Helal, "Bug report assignee recommendation using activity profiles," in *Proceedings of the 2013 10th Working Conference on Mining Software Repositories (MSR)*, IEEE, San Francisco, CA, USA, pp. 22–30, May 2013.

[36] T. Zhang, G. Yang, B. Lee, and E. K. Lua, "A novel developer ranking algorithm for automatic bug triage using topic model and developer relations," in *Proceedings of the 2014 21st Asia-Pacific Software Engineering Conference*, pp. 223–230, Jeju, Republic of Korea, December 2014.

[37] X. Xia, D. Lo, X. Wang, and B. Zhou, "Accurate developer recommendation for bug resolution," in *Proceedings of the 2013 20th Working Conference on Reverse Engineering (WCRE)*, IEEE, Koblenz, Germany, pp. 72–81, October 2013.

[38] M. Linares-Vásquez, K. Hossen, H. Dang, H. Kagdi, M. Gethers, and D. Poshyvanyk, "Triaging incoming change requests: bug or commit history, or code authorship?" in *Proceedings of the 2012 28th IEEE International Conference on Software Maintenance (ICSM)*, IEEE, Trento, Italy, pp. 451–460, September 2012.

[39] K. Kevic, S. C. Müller, T. Fritz, and H. C. Gall, "Collaborative bug triaging using textual similarities and change set analysis," in *Proceedings of the 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, IEEE, San Francisco, CA, USA, pp. 17–24, May 2013.

[40] S. Wang and D. Lo, "Version history, similar report, and structure: putting them together for improved bug localization," in *Proceedings of the 22nd International Conference on Program Comprehension*, pp. 53–63, Hyderabad, India, June 2014.

[41] R. Shokripour, J. Anvik, Z. M. Kasirun, and S. Zamani, "Why so complicated? simple term filtering and weighting for location-based bug report assignment recommendation," in *Proceedings of the 2013 10th Working Conference on Mining Software Repositories (MSR)*, IEEE, San Francisco, CA, USA, pp. 2–11, May 2013.

[42] T. Zhang and B. Lee, "An automated bug triage approach: a concept profile and social network based developer recommendation," in *Proceedings of the International Conference on Intelligent Computing*, pp. 505–512, Huangshan, China, July 2012.

[43] D. Challet and A. Lombardoni, "Bug propagation and debugging in asymmetric software structures," *Physical Review E*, vol. 70, no. 4, Article ID 046109, 2004.

[44] W.-F. Pan, B. Li, Y.-T. Ma, Y.-Y. Qin, and X.-Y. Zhou, "Measuring structural quality of object-oriented softwares via bug propagation analysis on weighted software networks," *Journal of Computer Science and Technology*, vol. 25, no. 6, pp. 1202–1213, 2010.

Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 3748383, 12 pages
https://doi.org/10.1155/2020/3748383

Research Article

Hierarchical Aggregation for Reputation Feedback of Services Networks

Rong Yang and Dianhua Wang

College of Computer Science and Technology, Hubei University of Science and Technology, Xianning 437100, China

Correspondence should be addressed to Rong Yang; harry804@163.com and Dianhua Wang; wdhtj@126.com

Received 21 November 2019; Revised 8 February 2020; Accepted 2 March 2020; Published 7 May 2020

Guest Editor: Chunlai Chai

Product ratings are popular tools to support buying decisions of consumers, which are also valuable for online retailers. In online marketplaces, vendors can use rating systems to build trust and reputation. To build trust, it is really important to evaluate the aggregate score for an item or a service. An accurate aggregation of ratings can embody the true quality of offerings, which is not only beneficial for providers in adjusting operation and sales tactics, but also helpful for consumers in discovery and purchase decisions. In this paper, we propose a hierarchical aggregation model for reputation feedback, where the state-of-the-art feature-based matrix factorization models are used. We first present our motivation. Then, we propose feature-based matrix factorization models. Finally, we address how to utilize the above modes to formulate the hierarchical aggregation model. Through a set of experiments, we can get that the aggregate score calculated by our model is greater than the corresponding value obtained by the state-of-the-art IRURe; i.e., the outputs of our models can better match the true rank orders.

1. Introduction

With the advances and rapid proliferation of Web 2.0 innovations, many sites on the World Wide Web offer consumers the possibility of sharing their experiences with products and services through reviews and ratings. Consumer feedback can not only rank a wide variety of online offerings, but also enable ease of discovery of more useful products and build trust in marketplaces. Moreover, positive consumer feedback contributes to increase in visibility and sale of offerings [1, 2]. Therefore, an accurate model of consumer feedback aggregation is absolutely critical for decision-making and marketing strategies of marketplaces, which can help users avoid bad choices and drive them toward more useful items.

Our goal in this paper is to study the problem of modeling consumer feedback from large-scale sale data in order to support personalized and scalable recommendation and demand-forecasting systems. We focus on modeling hierarchical aggregation method for reputation feedback of services networks.

1.1. Motivation. As shown in Figure 1, shopping is an individual or household's day-to-day activity, which can be simply divided into three stages, i.e., category purchase, product choice, and purchase quantity. For example, Amy would like to buy a carton of milk. When she wanders around fat-free milk and whole milk, she must do a choice. If fat-free milk, she should select a brand, finally deciding the quantity. Actually, the above purchase process indicates Amy's preferences.

Product preferences are generally reflected by purchase incidence or purchase quantity in a consumer's shopping history. In the field of recommender systems, consumer preference matching is well done in item-based collaborative filtering [3] and matrix factorization technique [4]. Moreover, user preferences are also taken into account in service selection [5, 6] and service composition [7–11]. To satisfy increasingly complex user requirements, PaaS (API-driven platform as a service cloud) allows quick composition of existing services to deliver packaged solutions. It is very important, for solution developers, to quickly assess those composite services and regular feedback on performance of

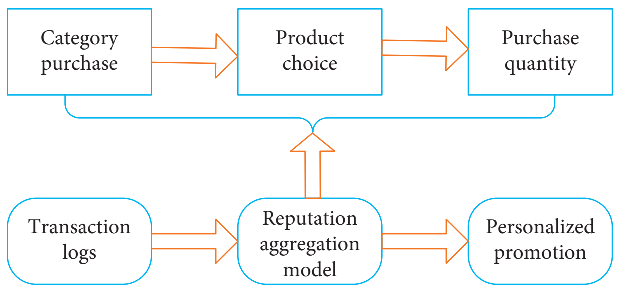


Figure 1: General shopping process.

component services. Only in this way can they dynamically update their compositions to ensure quality. However, during the process of assessment, consumer feedback plays a decisive role, which is dynamic and ephemeral. So, it is very crucial to efficiently aggregate consumer feedback.

1.2. Hierarchical Aggregation. To address the challenging problem about aggregations of consumer feedback, in this paper we present a hierarchical aggregation model for reputation feedback.

As shown in Figure 1, we model user shopping as a three-stage decision-making process (so does service composition, i.e., service provider selection, service categories choice, and quantity decision for each category of atomic service). In a real-world supermarket, we usually display products either based on an existing commodity hierarchy or by clustering their associated characteristics (e.g., text descriptions). For each category, it may consist of some kinds of products where consumers' purchase decisions share similar patterns. In womenswear department about sports style, for example, maybe you can find Adidas or Nike jackets. However, because of different user preferences [12–15], in a concrete purchase decision-making process, stages are heterogeneous.

In our model, we regard user category purchase as a binary prediction problem, where a multinomial distribution is explored to model the category purchasing process. Then, user will choose one product. However, user determines what quantity of a product, which is up to a numeric prediction problem. Our reputation feedback produce procedure where binary, categorical, and numeric prediction are combined, is quite different from that used by traditional ways of aggregating feedback. So, new approaches must be developed.

In this paper, we develop a hierarchical aggregation model and extend state-of-the-art feature-based matrix factorization models to include feedback as a factor. To summarize, in this paper, we make the following contributions:

(a) A generalized feature-based matrix factorization approach was adjusted and applied in our hierarchical feedback aggregation model.

(b) To evaluate the contribution of a node's own ratings to the aggregate score, we present a model which consists of two parts, i.e., the mean rating of the node and the mean rating of the node's universe. Moreover, the preceding models (detailed in Section 4) are used for relevance or weight estimation.

(c) To effectively evaluate the contribution of a node's child nodes to its aggregate score, a model in (14) is presented, where we do not only take sons into account, but also consider siblings and cousins (siblings and cousins are almost not concerned in existing models for reputational feedback). It is a weighted mean of the aggregate score AS(a_i) of the d child nodes. For each child, its contribution is controlled by two factors, i.e., the trust value of its ratings and the importance of its contribution.

(d) To illustrate the feasibility and efficiency of the proposed framework, we conduct comprehensive experiments. The experimental results show that the proposed framework is effective and efficient in the hierarchical aggregation of consumer feedback using consumer ratings.

The rest of this paper is organized as follows: Section 2 surveys related work on user preference, trust, and reputation management. Section 3 extends GLMix and consider a generalized feature-based matrix factorization (FBMF) model. Section 4 details the hierarchical aggregation model for reputation feedback. Section 5 discusses the experimental settings and results. Finally, Section 6 concludes this paper and outlines future work.

2. Related Work

The theme of user preference has been richly studied for recommender systems in various application scenarios such as content-based approaches [16, 17] and collaborative filtering approaches [3, 4, 18, 19]. To improve performance, [20, 21] both combine multiple techniques to achieve more complex tasks in hybrid recommender systems. Matrix factorization techniques are the most widely used methods in predicting the missing ratings of a user-item rating matrix due to their accuracy and scalability in prediction [18, 22–29]. In particular, feature-based matrix factorization techniques have been well done in [30–34]. Moreover, some researchers have developed efficient tools such as SVDFeature and libFM [35, 36]. Zhang et al. [37] presented a generalized linear mixed model (GLMix) for the LinkedIn job recommender system, where a scalable parallel block-wise coordinate descent algorithm was used. In this paper, we also concern user preference, but we focus on aggregating user preference by a hierarchical aggregation model. We build our model upon GLMix to fit different prediction settings.

It is also common to influence consumer behavior in making purchases based on aggregate consumer feedback [2, 38]. Floyd et al. [39] reached a conclusion that the volume of reviews, review valence, and influence of reviewers have a strong influence on purchasing decisions. For measuring the aggregate consumer preferences, researchers navigated many solutions to analyze the online product reviews. For instance, Ghose and Ipeirotis did reviews ranking by a consumer-oriented mechanism or a manufacturer-oriented mechanism, which were based on review helpfulness and review's expected effect on sales, respectively [40]. Xiao et al. [41] addressed an econometric preference measurement model, where a modified ordered choice model (MOCM) was also presented to extract aggregate consumer preferences from online product reviews. Banic et al. [42] focused on opinion mining by means of sentiment analysis, where a system was presented for collecting, evaluating, and aggregating user opinions. Zhang et al. [43] proposed a feedback aggregation approach to rank products based on the quality of reviews, which was calculated using a review's credibility as measured by helpfulness votes, relevance to the product, and the posting date of the reviews. However, all above approaches only consider product reviews rather than user ratings.

There are also several studies on trust and reputation management systems development, which aim to evaluate the reputation of services based on consumer feedback [44]. To monitor the execution of composite services, Bianculli et al. [45] presented a generic and customizable reputation infrastructure, where notifications upon changes in service reputation were allowable. In [46], Malik and Bouguettaya proposed a framework for establishing trust in service-oriented environments, where different ratings were aggregated to derive a service provider's reputation. Similarly, Wang et al. [47] proposed a reputation measure method for web services, which could ensure the reputation measure accuracy through two phases, i.e., malicious rating detection and rating adjustment [48]. Employed subjective probability theory to do trust evaluation for composite services. Different from our work, these work focuses on reputation system construction.

Many methods have been addressed to measure aggregate consumer preferences, which can be reduced to three major approaches: survey-, behavior-, and online review-based. Due to the advantages of conjoint analysis which depends strongly on survey data, it was explored to do preference measuring by Netzer et al. [49]. By means of collecting users' preference data from surveys or experiments, the survey-based approach could determine how people value the different features that constitute an individual product or service [50, 51]. However, they are time consuming and costly. To deal with these challenges, some work takes consumers' behavioral data into account to infer aggregate consumer preferences. For example, Fader and Hardie [52] presented a discrete choice model to measure consumer preferences for selected product features. But in [53], based on transaction data and path data, aggregate consumer preferences could be well estimated. Now, since online product reviews are available and accessible, several studies employed online product reviews to measure aggregate consumer preferences. For instance, Decker and Trusov proposed an econometric framework, which consisted of three models (i.e., Poisson's regression, negative binominal regression, and latent-class Poisson's regression models), to measure aggregate consumer preferences from online product reviews [54]. By means of analyzing the reviewers' knowledge and their opinion sentiment toward the target products, Li et al. [55] exploited a social intelligence mechanism for extracting and consolidating the reviews which could provide insights into enterprises to make decisions on product portfolio design. Different from previous work, this work focuses on the hierarchical aggregation of consumer reputation feedback.

Complex network refers to such network, which could have properties of self-organization, self-similarity, attractor, small world, or no scale. There are abundant examples of systems composed by a large number of highly interconnected dynamical units, such as neural networks, biological and chemical systems, the Internet, and the World Wide Web. To capture the global properties of such systems, we usually model them as graphs whose nodes represent the dynamical units and whose links stand for the interactions between them [56]. In [57], the authors addressed a survey of the use of measurements capable of expressing the most relevant topological features which characterize its connectivity and highly influence the dynamics of processes executed on the complex network. In [58], the authors explored the toolkit used for studying complex systems, i.e., nonlinear dynamics, statistical physics, and network theory.

At the same time, software networks have attracted more and more attention from various fields of science and engineering [59]. In [60], the optimal software-defined network planning was investigated with multicontrollers, where an adaptive feedback control mechanism was proposed. In [61], the authors explored the community structure of a real complex software network and correlated this modularity information with the internal dynamical processes, which the network is designed to support. Pan et al. [62] presented a systematic approach to investigate the complex software systems by using the k-core theories of complex networks. Wood et al. [63] addressed communication networks through the use of software-defined networking and the use of virtualization, where a comprehensive SDN control plane was needed. In [64, 65], the software key classes identification was addressed through the use of algorithms in complex networks.

Finally, service network is a typical complex adaptive system, and we can reveal the mechanism of its formation, evolution, and self-organization by the related theories and methods of complex network. For instance, in [66], the authors took advantage of the theory of complex network and existing networked software research works to explore the basic characteristics of services and service networks, such as the service network's "small world," "scale-free" characteristics and service network topology. Zhou and Wang [67] proposed a SCAS (service clustering approach using structural metrics) to group services into different clusters, where a metric A2S (atomic service similarity) was utilized to characterize the atomic service similarity. To explore the needs of support tools and service provisioning environments, [68] introduced the architecture of the open-source SONATA system, a service programming, orchestration, and management framework, where a development toolchain for virtualized network services could be fully integrated with a service platform and orchestration system. Correia et al. [69] proposed a hierarchical SDN-based vehicular architecture, which aimed to improve performance in the situation of loss of connection with the central SDN controller. Similarly, for services networks, we model user shopping or service purchasing as a three-stage decision-making process (i.e., provider selection, service or item categories choice, and quantity decision for each category), where a generalized feature-based matrix factorization (FBMF) model is used. We also address a hierarchical aggregation model for consumer ratings, so that the true quality of offerings can be embodied. Finally, we present how to combine the above models to raise the aggregation precision. Since the work in [70] is most similar to our approach, in the experiments, we will mainly detail the work of [70].

3. Preliminaries

In this section, we present a generalized feature-based matrix factorization approach, which can be adjusted and applied in our hierarchical feedback aggregation model. The basic notations used in this paper are shown in Table 1.

Generalized linear model (GLM) is widely used for statistical inference and response prediction problems. For example, in order to recommend relevant content to a user, a large number of web companies utilize logistic regression models to predict the probability of the user's clicking on an item (e.g., ad, news article, and job). In scenarios where the data is abundant, constructing a more fine-grained model focusing on user or item level would mostly contribute to more accurate prediction, since both the user's preferences on items and the item's specific attraction for users can be better captured. Some work combines ID-level regression coefficients with the global regression coefficients in a GLM setting [71], and such models are called generalized linear mixed models (GLMix) in the statistical literature.

Table 1: Notations.

Symbol	Description
i, u, t	Item, user, timestamp
C, gi, u	Global coefficient, global feature
$\widetilde{\Phi}_i^{(c)}$, $\widetilde{\Psi}_u^{(c)}$	Explicit item features, explicit user features
$\Phi_i^{(c)}$, $\Psi_u^{(c)}$	Item random coefficient, user random coefficient
$\Phi_i^{(lf)}$, $\Psi_u^{(lf)}$	Item latent factors, user latent factors
$\varsigma_u(t)$	Probability of user u selecting a category
$\xi_{s',u}(t)$	Conditional probability of user u purchasing $s\prime$
$\mathrm{OR}(a)$	Contribution of a's own ratings
$\mathrm{CR}(a)$	Contribution of a's child nodes
$\mathrm{MR}(a)$	Weighted mean rating of node a
$\mathrm{UR}(a)$	Weighted mean rating of universe of node a
Rai	ith consumer rating of node a
C_u^{ai}	Consumer credibility for Rai of node a
$\mathrm{TV}(a)$	Trust value of ratings of node a
TV_a	Trust votes of node a

In this paper, we extend GLMix and consider a generalized feature-based matrix factorization (FBMF) model:

$$\mathrm{link}(L(t)) = K(t) \approx \Phi(t)^T \Psi(t). \tag{1}$$

Here, $L(t)$ is the time-aware label matrix, where each element $li, u(t)$ indicates the label for an item i and a user u at timestamp t. Depending on the application, $li, u(t)$ can be either a real label or a binary label. When users explicitly express their opinions on products, $li, u(t)$ is a real label, often in the range [1, 5], and $li, u(t)$ is a binary label when predicting category purchase or product choice. The original label matrix can be transformed into a numeric matrix $K(t)$ by means of the logit function or logarithm function. And we decompose $K(t)$ as a product of $\Phi(t)$ and $\Psi(t)$, where $\Phi(t)$ and $\Psi(t)$ embody both explicit features and latent factors from items and users. For each element $k_{i,u}(t)$ in $K(t)$, it can be formulated as follows:

$$\begin{aligned} k_{i,u}(t) &\approx \langle \Phi i(t), \Psi u(t) =\rangle \\ &\underbrace{< \mathbb{C}, \overbrace{gi, u(t)}^{\text{global features}} >}_{\text{global effect}} + \underbrace{< \overbrace{\widetilde{\Phi}_i^{(c)}(t)}^{\text{item features}}, \Psi_u^{(c)} > + < \Phi_i^{(c)}, \overbrace{\widetilde{\Psi}_u^{(c)}(t)}^{\text{user features}} >}_{\text{observed item/user–specific effect}} \\ &+ \underbrace{< \Phi_i^{(lf)}, \Psi_u^{(lf)} >}_{\text{latent item–user interaction}}, \end{aligned} \tag{2}$$

where <, > denotes the inner product. In our model, we simply decompose each prediction into three components, i.e., global effects, observed item/user-specific effects, and latent item-user interactions.

Specifically, for global effects, $gi, u(t)$ includes a set of features for (i, u, t) and $\mathbb{C}$ denotes a set of global coefficients, which can be estimated but should be consistent for all (i, u, t) triples. For example, the weighted mean rating of universe of a node x and universal relevance are all such features. In fact, the second term (i.e., item/user-specific effects) is similar to the random coefficient model [72, 73], which includes explicit features with item- or user-dependent coefficients. Generally speaking, in our model, contribution of node x from its own ratings and consumer credibility are explicit item- and user-related features. Finally, latent item-user interaction is designed to capture the remaining latent effects in terms of low-rank user and item factors.

4. Methodology

To achieve more complex tasks or to mash up data from different data resources by using business process description languages, web services usually need to be composed as workflows (i.e., service processes). As shown in Figure 2, the process of constructing a service process can be simply divided into three stages, i.e., service provider selection, atomic service categories choice, and quantity decision for each category of atomic service. In this section, we present an integrated model to produce the aggregation of feedback.

Users interact with services from a marketplace where both atomic and composite services are available, refer to existing feedback, and provide feedback based on their own perception. According to the different contexts, a service can independently receive direct feedback. Therefore, we aggregate the feedback of a composite service based on not only its direct feedback, but also the aggregate feedback of its components. Below, we detail the hierarchical aggregation method that provides an accurate evaluation of feedback.

Given a service $s\prime$ in service category sc, a user u, and a timestamp t, suppose there are the following definitions:

$\mathrm{SC}_u^{sc}(t)$: user u selects the service category sc at time t;
$S_u^{s'}(t)$: user u selects the service s' at time t;
$Q_u^{s'}(t) = n$: user u's selection quantity of s' at t is n.

Thus, assuming that we focus on the service category sc, user u's preferences can be calculated by the joint probability of choosing a certain quantity of service s' in category sc; i.e.,

$$P\left(Q_u^{s'}(t) = n, S_u^{s'}(t), SC_u^{sc}(t)\right) = \underbrace{P\left(SC_u^{sc}(t)\right)}_{\substack{\text{category}\\ \text{preference}}} \times \underbrace{P\left(S_u^{s'}(t) \,\middle|\, SC_u^{sc}(t)\right)}_{\substack{\text{conditional}\\ \text{service preference}}} \times \underbrace{P\left(Q_u^{s'}(t) = n \,\middle|\, S_u^{s'}(t), SC_u^{sc}(t)\right)}_{\substack{\text{conditional}\\ \text{quantity preference}}}. \tag{3}$$

Equation (3) can be regarded as a product of three conditional probabilities which represent the preferences in previous service selection stages. By adopting different link functions in the previous FBMF formulation, these three preferences can be estimated by logistic, categorical, and quantity-based FBMF models.

Service Category Selection (C-FBMF). For a given service category sc, user u can get the following logistic probability:

$$\varsigma_u(t) := P\left(\mathrm{SC}_u^{sc}(t)\right) = \sigma\left(s_u^{(\text{cate})}(t)\right), \tag{4}$$

where $\sigma(\cdot)$ is the sigmoid function, and $s_u^{(\text{cate})}(t)$ denotes a service category preference score, factorized using (2), where there is only one general "item," i.e., the service category sc.

Atomic Service Choice (S-FBMF). Next, we formulate the probability of selecting an atomic service within a service category as a multinomial distribution via a softmax formulation:

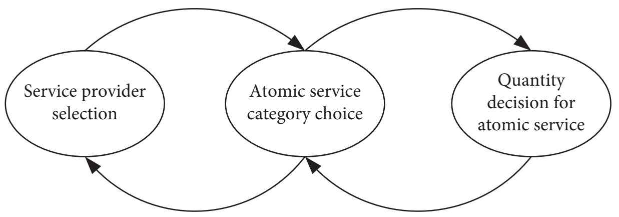

Service provider selection
Atomic service category choice
Quantity decision for atomic service

FIGURE 2: The hierarchical composition for service process.

$$\xi_{s',u}(t) := P\left(S_u^{s'}(t) \,\middle|\, \mathrm{SC}_u^{\mathrm{sc}}(t)\right) = \frac{\exp\left(s_{s',u}^{(\text{atom})}(t)\right)}{\sum_{s''}\exp\left(s_{s'',u}^{(atom)}(t)\right)}. \tag{5}$$

Similarly, the atomic service preference score $s_{s',u}^{(\text{atom})}(t)$ is factorized by (2).

Atomic Service Quantity Decision (Q-FBMF). The quantity of choosing an atomic service $s\prime$ follows a shifted Poisson distribution:

$$P\left(Q_u^{s'}(t) = n \,\middle|\, S_u^{s'}(t), SC_u^{sc}(t)\right) = \frac{\tau s', u(t)^{n-1}\exp\left(-\tau s', u(t)\right)}{(n-1)!}, \tag{6}$$

where $\tau s', u(t) = \exp\left(s_{s',u}^{(\text{quan})}(t)\right)$. Again, we apply (2) to factorize the atomic service quantity preference scores$_{s',u}^{(quan)}(t)$, and we can get the conditional expectation of atomic service quantity as

$$\hat{q}_u^{s'}(t) := E\left(Q_u^{s'}(t) \,\middle|\, S_u^{s'}(t), \mathrm{SC}_u^{\mathrm{sc}}(t)\right) = \tau s', u(t) + 1, \tag{7}$$

which can be taken as an estimate of $Q_u^{s'}(t)$.

Consider the generalized hierarchy for service composition shown in Figure 3, based on the composite service decision process in Figure 2. Feedback aggregation is performed for every node at each level of the tree, starting from the bottom with the leaves. In this work, we combine all ratings for a particular node to have a single 5-star score. In short, for a node at a higher level, the aggregation score involves not only its own ratings, but also contributions from the lower-level descendants.

For a node a, its aggregate score is calculated as follows:

$$\mathrm{AS}(a) = \beta \times \mathrm{OR}(a) + (1-\beta) \times \mathrm{CR}(a), \tag{8}$$

where $\mathrm{OR}(a)$ denotes the contribution of a's own ratings, $\mathrm{CR}(a)$ represents the contribution of its child nodes, and β is a system parameter. If a has no child nodes, then $\beta = 1$, and vice versa.

$$\mathrm{OR}(a) = \chi \times \mathrm{MR}(a) + (1-\chi) \times \mathrm{UR}(a). \tag{9}$$

We can evaluate the contribution of a node's own ratings by (9). In (9), it consists of two parts, i.e., the mean rating of the node $(\mathrm{MR}(a))$ and the mean rating of the node's universe $(\mathrm{UR}(a))$. If there are not numerous ratings for a, the existing ratings of its similar nodes (e.g., other instances of a) are used, as it is possible that $a\prime s$ ratings will be analogous to the ratings of similar nodes. So, (9) is a trade-off between $\mathrm{MR}(a)$ and $\mathrm{UR}(a)$. Generally speaking, (9) is a weighted

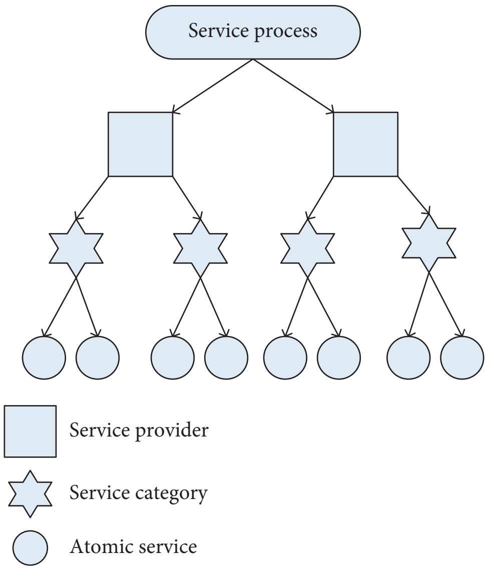


Figure 3: A generalized hierarchy for service composition.

mean such that the nodes with fewer ratings are dominated by the mean rating across similar nodes, while the nodes with more ratings are mostly dominated by its own mean rating.

$$\mathrm{MR}(a) = \frac{\sum_{i=1}^{k} Rai \times \xi_{a,u}(t) \times C_u^{ai}}{\sum_{i=1}^{k} \xi_{a,u}(t) \times C_u^{ai}}. \tag{10}$$

We use (10) to calculate a node's mean rating, which is a weighted mean of k ratings received by a node. As shown in (10), Rai denotes a rating, and its weight comes from (5). The weight can indicate the utility of a service as perceived by the user. C_u^{ai} presents the credibility of user u who makes the rating and adjusts the rating accordingly. Actually, there are users who may try to drive up or down the rating score. By means of adjusting the contribution of each rating based on the respective weight of user credibility, we can lower the influence of those fake users.

$$\mathrm{UR}(a) = \frac{\delta1\left(\sum_{i=1}^{m}\sum_{j=1}^{k1} Raij \times \xi_{a,u}^{ij}(t) \times C_u^{aij}\right) + \delta2\left(\sum_{i=1}^{n}\sum_{j=1}^{k2} Ra\prime ij \times \xi_{a',u}^{ij}(t) \times C_u^{a'ij}\right)}{\delta1\left(\sum_{i=1}^{m}\sum_{j=1}^{k1} \xi_{a,u}^{ij}(t) \times C_u^{aij}\right) + \delta2\left(\sum_{i=1}^{n}\sum_{j=1}^{k2} \xi_{a',u}^{ij}(t) \times C_u^{a'ij}\right)}, \tag{11}$$

$$\delta1 = P\left(Q_u^a(t) = m \,\middle|\, S_u^a(t), \mathrm{SC}_u^{sc}(t)\right) = \frac{\tau a, u(t)^{m-1}\exp(-\tau a, u(t))}{(m-1)!}, \tag{12}$$

$$\delta2 = P\left(Q_u^{\acute{a}}(t) = n \,\middle|\, S_u^{\acute{a}}(t), \mathrm{SC}_u^{s\acute{c}}(t)\right) = \frac{\tau a\prime, u(t)^{n-1}\exp(-\tau a\prime, u(t))}{(n-1)!}. \tag{13}$$

Equation (11) is used to evaluate the mean rating of a node's universe. Generally speaking, the universe refers to the set of nodes similar to this node. In this work, we just consider two levels of similarity—siblings and cousins. As shown in (11), for a service node a with service category sc, it may have m siblings and n cousins with k_1 and k_2 ratings, respectively. For the m siblings, they could be instances of a, which can independently receive direct feedback. However, for the n cousins, they might come from different service categories, even from different service providers. $\delta1$ and $\delta2$ are sibling similarity weight and cousin similarity weight, respectively. Obviously, sibling nodes have a higher degree of similarity than the cousin nodes; i.e., $\delta1$ may be greater than$\delta2$:

$$\mathrm{CR}(a) = \frac{\sum_{i=1}^{d} \mathrm{AS}(a_i) \times \mathrm{TV}(a_i) \times w(a, a_i)}{\sum_{i=1}^{d} \mathrm{TV}(a_i) \times w(a, a_i)}. \tag{14}$$

In (8), $\mathrm{CR}(a)$ represents the contribution of the d child nodes to $a's$ aggregate score. We use (14) to calculate it, which is a weighted mean of the aggregate score $\mathrm{AS}(a_i)$ of the d child nodes. For each child a_i, its contribution is controlled by two factors, i.e., the trust value of its ratings and the importance of its contribution, which are denoted as

$\mathrm{TV}(a_i)$ and $w(a, a_i)$, respectively. $w(a, a_i)$ can be decided by $a_i^{\prime}s$ age, functionality, frequency of usage, etc. From (14), we can conclude that all a node's descendant nodes contribute its aggregate score:

$$\mathrm{TV}(a) = \frac{1}{2}\left(\mathrm{TV}_a + \frac{1}{d}\sum_{i=1}^{d}\mathrm{TV}(a_i)\right). \quad (15)$$

We define trust value by (15), which is an arithmetic mean and consists of two parts, i.e., a node's own trust votes TV_a and the trust values of its d child nodes $\mathrm{TV}(a_i)$. The trust value of a node is a measure of consumer confidence in its ratings and can be used as a replacement of the number of ratings for a service.

$$\mathrm{TV}_a = \sum_{i=1}^{k}\xi_{a,u}(t)\times C_u^{ai}. \quad (16)$$

By means of summing the multiplication of k feedback relevance $\xi_{a,u}(t)$ and the respective consumer credibility C_u^{ai} received by the node, we can get the trust value of itself for a node.

5. Experiments and Results Analysis

5.1. Datasets. In this section, we conduct experiments to evaluate our approach. We compare our FBMF with the method detailed in [70] on multiple public real-world datasets, which are extracted from Amazon.com by McAuley et al. [74]. The datasets contain product reviews (i.e., ratings, text, and helpfulness votes) and product metadata. Specifically, the metadata includes price, title, a list of also viewed products, and a list of also bought products. We preprocess all datasets so that each user rated at least four products. Table 2 details the statistics of our datasets, which include five datasets, i.e., Baby, Office Products, Pet Supplies, Electronics, and Sports and Outdoors. In Figure 4, the number of rated products in each dataset is counted, respectively.

All experiments are implemented in Java. The hardware environment is a machine with the Intel® Core™ i5 CPU 760, 2.80 GHz, and 4 GB RAM running Windows 7 (64-bit version).

5.2. Relevance Estimation. In Section 4, we use (5) to model input relevance, i.e., the utility of a service as perceived by the consumer. In Amazon, we can find "N people found this helpful" for each review along with Yes and No buttons. Many online malls similarly allow customers to upvote or downvote those posted reviews, which can present an idea about their relevance and be formulated as follows [70] (for simplicity, we call this method IRURe):

$$\mathrm{Rel} = \frac{Us}{Ts\max} + \left(1 - \frac{Ts}{Ts\max}\right)\times\frac{\sum_{i=1}^{k}Usi}{\sum_{i=1}^{k}Tsi}. \quad (17)$$

Re l is a weighted mean of the initial relevance (*IRe*, the former part of (17)) and the universal relevance (*URe*, the final part of (17)) of a review, where Us denotes the upvotes

Table 2: Data statistics.

Dataset	Users no.	Products no.	Ratings no.
Baby	531890	64426	915446
Office Products	909314	130006	1243186
Pet Supplies	740985	103288	1235316
Electronics	4201696	476002	7825308
Sports and Outdoors	1990521	478898	3268695

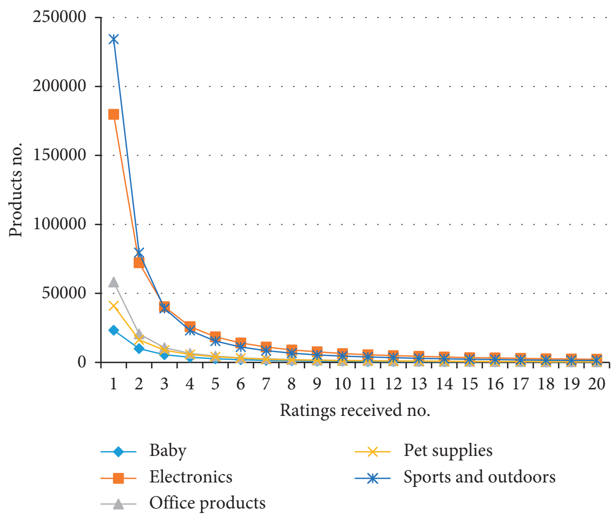


Figure 4: The number of rated products in each dataset.

on a review, Ts is the total votes on a review, and $Ts_{\max}$ is the maximum total votes across all reviews in the universe.

In the next section, we will conduct several groups of experiments to evaluate the effectiveness and robustness of our approach.

5.3. Experimental Results. Both our model FBMF (in (8)) and IRURe (detailed in [70]) can get an aggregate score for a node, respectively. A higher aggregate score means a best-selling product or a more popular service, but is that really the case?

Actually, it is really difficult to evaluate the true quality of a product due to the subjectivity in the process. To deal with this problem, many researchers try to evaluate the effectiveness of a product ranking system using the sales rank feature of products [39], where the relative rank of a product in a given category is indicated by the amount of its sales. In our experiments, for the five datasets (i.e., Baby, Office Products, Pet Supplies, Electronics, and Sports and Outdoors), we choose the top five aggregate scores, respectively. Then, under each dataset, we take pairwise comparison of true relative sales ranks of products with the ranking order generated by the mentioned models. Through experiments, we analyze how well the outputs of the models match the true rank orders; i.e., a higher aggregate score should translate into a better (smaller) sales rank.

In our experiments, we use the sales rank values in metadata, which are extracted from Amazon.com by McAuley et al. [74]. The below five tables, Tables 3–7, are the

Table 3: Correlation of aggregate rating and sales rank on Baby.

Product ID	IRURe	FBMF	Sales ranks	IRURe-A	FBMF-A
B004U47T38 vs. B0089PSCVC	4.4235 vs. 4.312	4.7694 vs. 4.4248	154314 vs. 302889	√	√
B004U47T38 vs. B006PZ3WWC	4.4235 vs. 4.2725	4.7694 vs. 4.7702	154314 vs. 59909	√	√
B004U47T38 vs. B00HSFF9WY	4.4235 vs. 3.76	4.7694 vs. 4.4405	154314 vs. 11262	×	×
B004U47T38 vs. B005NV518M	4.4235 vs. 2	4.7694 vs. 3.2	154314 vs. 450577	√	√
B0089PSCVC vs. B005PWE6US	4.312 vs. 4.2725	4.4248 vs. 4.7702	302889 vs. 59909	√	√
B0089PSCVC vs. B00HSFF9WY	4.312 vs. 3.76	4.4248 vs. 4.4405	302889 vs. 11262	√	√
B0089PSCVC vs. B005NV518M	4.312 vs. 2	4.4248 vs. 3.2	302889 vs. 450577	√	√
B005PWE6US vs. B00HSFF9WY	4.2725 vs. 3.76	4.7702 vs. 4.4405	59909 vs. 11262	×	×
B005PWE6US vs. B005NV518M	4.2725 vs. 2	4.7702 vs. 3.2	59909 vs. 450577	√	√
B00HSFF9WY vs. B005NV518M	3.76 vs. 2	4.4405 vs. 3.2	11262 vs. 450577	√	√

Table 4: Correlation of aggregate rating and sales rank on Electronics.

Product ID	IRURe	FBMF	Sales ranks	IRURe-A	FBMF-A
B00000J49E vs. B000UVWLUQ	5 vs. 4.9335	5 vs. 4.9734	73397 vs. 136262	√	√
B00000J49E vs. B000N3SR8Q	5 vs. 4.923	5 vs. 4.9692	73397 vs. 140901	√	√
B00000J49E vs. B000MWFDF8	5 vs. 4.9165	5 vs. 4.9666	73397 vs. 174604	√	√
B00000J49E vs. B000X18Y9U	5 vs. 4.9121	5 vs. 4.8594	73397 vs. 180386	√	√
B000UVWLUQ vs. 000N3SR8Q	4.9335 vs. 4.923	4.9734 vs. 4.9692	136262 vs. 140901	√	√
B000UVWLUQ vs. 000MWFDF8	4.9335 vs. 4.9165	4.9734 vs. 4.9666	136262 vs. 174604	√	√
B000UVWLUQ vs. 000X18Y9U	4.9335 vs. 4.9121	4.9734 vs. 4.8594	136262 vs. 180386	√	√
B000N3SR8Q vs. B000MWFDF8	4.923 vs. 4.9165	4.9692 vs. 4.9666	140901 vs. 174604	√	√
B000N3SR8Q vs. B000X18Y9U	4.923 vs. 4.9121	4.9692 vs. 4.8594	140901 vs. 180386	√	√
B000MWFDF8 vs. B000X18Y9U	4.9165 vs. 4.9121	4.9666 vs. 4.8594	174604 vs. 180386	√	√

Table 5: Correlation of aggregate rating and sales rank on Office Products.

Product ID	IRURe	FBMF	Sales ranks	IRURe-A	FBMF-A
1842104837 vs. B004I40BNK	5 vs. 4.9735	5 vs. 4.9894	950114 vs. 135142	×	×
1842104837 vs. B005NSB69I	5 vs. 4.9565	5 vs. 4.9826	950114 vs. 640434	×	×
1842104837 vs. B00FO81MCS	5 vs. 4.9375	5 vs. 4.975	950114 vs. 2058369	√	√
1842104837 vs. B001XE79S8	5 vs. 4.896	5 vs. 4.9816	950114 vs. 682879	×	×
B004I40BNK vs. B005NSB69I	4.9735 vs. 4.9565	4.9894 vs. 4.9826	135142 vs. 640434	√	√
B004I40BNK vs. B00FO81MCS	4.9735 vs. 4.9375	4.9894 vs. 4.975	135142 vs. 2058369	√	√
B004I40BNK vs. B001XE79S8	4.9735 vs. 4.896	4.9894 vs. 4.9816	135142 vs. 682879	√	√
B005NSB69I vs. B00FO81MCS	4.9565 vs. 4.9375	4.9826 vs. 4.975	640434 vs. 2058369	√	√
B005NSB69I vs. B001XE79S8	4.9565 vs. 4.896	4.9826 vs. 4.9816	640434 vs. 682879	√	√
B00FO81MCS vs. B001XE79S8	4.9375 vs. 4.896	4.975 vs. 4.9816	2058369 vs. 682879	×	√

Table 6: Correlation of aggregate rating and sales rank on Pet Supplies.

Product ID	IRURe	FBMF	Sales ranks	IRURe-A	FBMF-A
B002JBDF6E vs. B0051BWC1S	4.9775 vs. 4.9615	4.991 vs. 4.9846	57251 vs. 81784	√	√
B002JBDF6E vs. B009V18PJM	4.9775 vs. 4.9765	4.991 vs. 4.9834	57251 vs. 103444	√	√
B002JBDF6E vs. B001UH5EZI	4.9775 vs. 4.9530	4.991 vs. 4.9812	57251 vs. 106414	√	√
B002JBDF6E vs. B00448HS36	4.9775 vs. 4.9500	4.991 vs. 4.9800	57251 vs. 146735	√	√
B0051BWC1S vs. B009V18PJM	4.9615 vs. 4.9765	4.9846 vs. 4.9834	81784 vs. 103444	×	√
B0051BWC1S vs. B001UH5EZI	4.9615 vs. 4.9530	4.9846 vs. 4.9812	81784 vs. 106414	√	√
B0051BWC1S vs. B00448HS36	4.9615 vs. 4.9500	4.9846 vs. 4.9800	81784 vs. 146735	√	√
B009V18PJM vs. B001UH5EZI	4.9765 vs. 4.9530	4.9834 vs. 4.9812	103444 vs. 106414	√	√
B009V18PJM vs. B00448HS36	4.9765 vs. 4.9500	4.9834 vs. 4.9800	103444 vs. 146735	√	√
B001UH5EZI vs. B00448HS36	4.9530 vs. 4.9500	4.9812 vs. 4.9800	106414 vs. 146735	√	√

experimental results for the five datasets, respectively. Among those tables, the first column is the IDs of two compared products. The second and the third columns correspond to aggregate scores obtained by IRURe and FBMF, respectively. For simplicity, all aggregate scores are normalized into the range of zero to five. The corresponding sales ranks for pairwise products are presented in column 4. The two rightmost columns show the accuracy of the models

Table 7: Correlation of aggregate rating and sales rank on Sports and Outdoors.

Product ID	IRURe	FBMF	Sales ranks	IRURe-A	FBMF-A
B0016NPB54 vs. B001P9M76A	4.9665 vs. 4.963	4.9866 vs. 4.9852	66779 vs. 334514	√	√
B0016NPB54 vs. B00005USQZ	4.9665 vs. 4.9585	4.9866 vs. 4.9864	66779 vs. 224671	√	√
B0016NPB54 vs. B000RLLW8G	4.9665 vs. 4.9583	4.9866 vs. 4.9334	66779 vs. 330559	√	√
B0016NPB54 vs. B000PO018W	4.9665 vs. 4.9588	4.9866 vs. 4.9810	66779 vs. 406073	√	√
B001P9M76A vs. B00005USQZ	4.963 vs. 4.9585	4.9852 vs. 4.9864	334514 vs. 224671	×	√
B001P9M76A vs. B000RLLW8G	4.963 vs. 4.9583	4.9852 vs. 4.9334	334514 vs. 330559	×	×
B001P9M76A vs. B000PO018W	4.963 vs. 4.9588	4.9852 vs. 4.9810	334514 vs. 406073	√	√
B00005USQZ vs. B000RLLW8G	4.9585 vs. 4.9583	4.9864 vs. 4.9334	224671 vs. 330559	√	√
B00005USQZ vs. B000PO018W	4.9585 vs. 4.9588	4.9864 vs. 4.9810	224671 vs. 406073	×	√
B000RLLW8G vs. B000PO018W	4.9583 vs. 4.9588	4.9334 vs. 4.9810	330559 vs. 406073	×	×

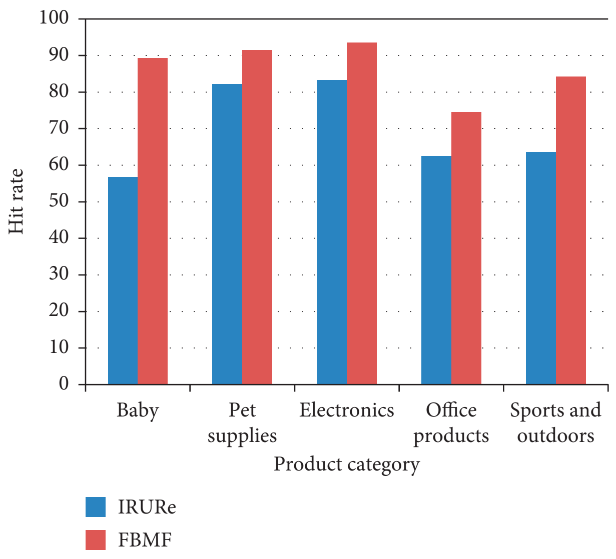


Figure 5: The hit rate in each dataset.

IRURe and FBMF in capturing the true rank ordering of the products.

As we can see from Tables 3–7, on each product, the aggregate score calculated by our model is greater than the corresponding value obtained by IRURe. This is attributed to our relevance model, which is detailed in Section 4. The results among the five datasets show that the pairwise orderings generated by FBMF always capture the relative ranking of the products and are better than (or as good as) the ones generated by IRURe. For example, on Baby's dataset, IRURe missed five pairwise orderings, but FBMF missed only two ones. Particularly, on Electronics and Pet Supplies, FBMF hits at all.

In each dataset, there are tens of thousands of product reviews, so we cannot list all the pairwise products in a table. For simplicity, the respective five products corresponding to the top five aggregate scores are chosen to be displayed in Tables 3–7. However, for each dataset, we did all the pairwise comparisons, where those products with reviews and sales ranks were all covered. Figure 5 is the statistical results about hit rates throughout the five datasets. As shown in Figure 5, FBMF has a higher hit rate than IRURe in each dataset. Particularly, in Electronics, FBMF even has a hit rate of 93.56%. The results for FBMF vs. IRURe reconfirm that FBMF is able to capture the true relative order, although IRURe also has the same capability in the most cases.

6. Conclusions

Consumer feedback, for example, product review, is an important source of information for customers to support their buying decision. Though product reviews are really helpful for customers, aggregate responses from the participants indicated that current rating systems also have their weaknesses, especially when review scales are large. It is an important but difficult task to develop a new feedback mechanism and management of feedback aggregation. In this paper, we propose a hierarchical aggregation model for reputation feedback, which is based on a generalized feature-based matrix factorization model. This model aims to aggregate consumer feedback from large-scale sale data in order to support personalized and scalable recommendation and demand-forecasting systems. We conduct several groups of experiments to evaluate the efficiency and robustness of our approach. Experiments show that FBMF performs well. Currently, we mainly consider ratings. Our future work is to investigate how to incorporate the information of "also viewed products" and "also bought products" into our approach.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the Natural Science Foundation of Hubei Province under Grant no. 2019CFC888 and Research Development Fund of Hubei University of Science and Technology under Grant no. 2019–21GP06.

References

[1] M. Luca, "Reviews, reputation, and revenue: the case of yelp.com," Harvard Business School NOM, Boston, MA, USA, 2016.

[2] G. Lackermair, D. Kailer, and K. Kanmaz, "Importance of online product reviews from a consumers perspective," *Advances in Economics and Business*, vol. 1, no. 1, 2013.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the Tenth International Conference on World Wide Web*, Hong Kong, May 2001.

[4] L. Hao, K. Li, J. An et al., "MSGD: a novel matrix factorization approach for large-scale collaborative filtering recommender systems on GPUs," *IEEE Transactions on Parallel & Distributed Systems*, vol. 29, pp. 1530–1544, 2018.

[5] R. Yang and B. Li, "Reusing service process fragments with a linguistic approach for user qualitative preferences," in *Proceedings of The 2014 International Conference on Cloud Computing and Big Data (CCBD2014, EI)*, pp. 152–159, Wuhan, China, November 2014.

[6] F. Dahan, H. Mathkour, and M. Arafah, "Two-step artificial bee colony algorithm enhancement for QoS-aware web service selection problem," *IEEE Access*, vol. 7, pp. 21787–21794, 2019.

[7] S. Badsha, X. Yi, I. Khalil, D. Liu, S. Nepal, and K.-Y. Lam, "Privacy preserving user based web service recommendations," *IEEE Access*, vol. 6, pp. 56647–56657, 2018.

[8] N. Lin, E. Sirin, and E. Sirin, "Web service composition with user preferences," in *Proceedings of the European Semantic Web Conference on the Semantic Web: Research & Applications*, Tenerife, Island, June 2008.

[9] A. V. Dastjerdi and R. Buyya, "Compatibility-aware cloud service composition under fuzzy preferences of users," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 1–13, 2014.

[10] Z. Wei, J. Wen, G. Min et al., "A QoS preference-based algorithm for service composition in service-oriented network," *Optik—International Journal for Light and Electron Optics*, vol. 124, no. 20, pp. 4439–4444, 2013.

[11] K. Benouaret, D. Benslimane, A. Hadjali et al., "Web service compositions with fuzzy preferences: a graded dominance relationship-based approach," *Acm Transactions on Internet Technology*, vol. 13, no. 4, p. 12, 2014.

[12] M. Karanik, R. Bernal, J. I. Peláez, and J. A. Gomez-Ruiz, "Combining user preferences and expert opinions: a criteria synergy-based model for decision making on the Web," *Soft Computing*, vol. 23, no. 4, pp. 1357–1373, 2019.

[13] H. Jiang, Z. Hu, X. Zhao, L. Yang, and Z. Yang, "Exploring the users' preference pattern of application services between different mobile phone brands," *IEEE Transactions on Computational Social Systems*, vol. 5, no. 4, pp. 1163–1173, 2018.

[14] D. Ke, L. Yanhua, Z. Jia et al., "User preference analysis for most frequent peer/dominator," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, pp. 1421–1425, 2018.

[15] W. Shuai, S. Ali, Y. Tao et al., "Integrating weight Assignment strategies with NSGA-II for supporting user preference multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 3, pp. 378–393, 2018.

[16] P. Lops, M. De Gemmis, and G. Semeraro, "Content-based recommender systems: state of the art and trends," in *Recommender Systems Handbook*, pp. 73–105, Springer, Berlin, Germany, 2011.

[17] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The Adaptive Web*, pp. 325–341, Springer, Berlin, Germany, 2007.

[18] Y. Koren, R. Bell, C. Volinsky et al., "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[19] G.-N. Hu, X.-Y. Dai, F.-Y. Qiu et al., "Collaborative filtering with topic and social latent factors incorporating implicit feedback," *Acm Transactions on Knowledge Discovery from Data*, vol. 12, no. 2, pp. 1–30, 2018.

[20] R. Burke, "Hybrid recommender systems: survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[21] A. Gunawardana and C. Meek, "A unified approach to building hybrid recommender systems," in *Proceedings of the Third ACM Conference on Recommender Systems—RecSys '09*, New York, NY, USA, October 2009.

[22] H. Yan, G. Liao, Z. Zhen et al., "Fast narrowband RFI suppression algorithms for SAR systems via matrix-factorization techniques," *IEEE Transactions on Geoscience & Remote Sensing*, vol. 57, pp. 250–262, 2018.

[23] Žitnik, Marinka, and B. Zupan, "NIMFA: a python library for nonnegative matrix factorization," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 849–853, 2018.

[24] C. Y. Lin, L. C. Wang, and K. H. Tsai, "Hybrid real-time matrix factorization for implicit feedback recommendation systems," *IEEE Access*, vol. 6, 2018.

[25] C. Févotte and M. Kowalski, "Estimation with low-rank time-frequency synthesis models," *IEEE Transactions on Signal Processing*, vol. 66, no. 15, pp. 4121–4132, 2018.

[26] D. Kotzias, M. Lichman, and P. Smyth, "Predicting consumption patterns with repeated and novel events," *IEEE Transactions on Knowledge & Data Engineering*, vol. 31, no. 2, pp. 371–384, 2018.

[27] X. Li and K. C. Wong, "A comparative study for identifying the chromosome-wide spatial clusters from high-throughput chromatin conformation capture data," *IEEE/ACM Transactions on Computational Biology & Bioinformatics*, vol. 15, no. 3, pp. 774–787, 2018.

[28] L. Baltrunas, B. Ludwig, and F. Ricci, "Matrix factorization techniques for context aware recommendation," in *Proceedings of the Fifth ACM Conference on Recommender Systems—RecSys '11*, Chicago, IL, USA, October 2011.

[29] P. Forbes and M. Zhu, "Content-boosted matrix factorization for recommender systems: experiments with recipe recommendation," in *Proceedings of the Fifth ACM Conference on Recommender Systems—RecSys '11*, Chicago, IL, USA, October 2011.

[30] D. Agarwal and B.-C. Chen, "Regression-based latent factor models," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD '09*, New York, NY, USA, June 2009.

[31] A. Ahmed, B. Kanagal, S. Pandey, V. Josifovski, L. G. Pueyo, and J. Yuan, "Latent factor models with additive and hierarchically-smoothed user preferences," in *Proceedings of the Sixth ACM International Conference on Web search and Data Mining—WSDM '13*, Rome Italy, February 2013.

[32] X. Ning and G. Karypis, "Sparse linear methods with side information for top-n recommendations," in *Proceedings of the Sixth ACM Conference on Recommender Systems—RecSys '12*, Dublin Ireland, September 2012.

[33] S. Park, Y.-D. Kim, and S. Choi, "Hierarchical bayesian matrix factorization with side information," in *Proceedings of The Twenty-Third International Joint Conference on Artificial Intelligence*, Pohang-si, South Korea, August 2013.

[34] I. Porteous, A. U. Asuncion, and M. Welling, "Bayesian matrix factorization with side information and dirichlet process mixtures," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, Atlanta, GA, USA, July 2010.

[35] T. Chen, W. Zhang, Q. Lu, K. Chen, Z. Zheng, and Y. Yu, “SVDFeature: a toolkit for feature-based collaborative filtering,” *JMLR*, vol. 13, pp. 3619–3622, 2012.

[36] S. Rendle, “Factorization machines with libfm,” *TIST*, vol. 3, no. 3, p. 57, 2012.

[37] X. Zhang, Y. Zhou, Y. Ma, B. Chen, L. Zhang, and D. A. garwal, “Glmix: generalized linear mixed models for large-scale response prediction,” in *Proceedings of the 22nd ACM SIGKDD International Conference*, San Francisco, CA, USA, August 2016.

[38] F. Wu and B. A. Huberman, “How public opinion forms,” in *Proceedings of the 4th International Workshop on Internet and Network Economics*, pp. 334–341, Shanghai, China, 2008.

[39] K. Floyd, R. Freling, S. Alhoqail, H. Y. Cho, and T. Freling, “How online product reviews affect retail sales: a meta-analysis,” *Journal of Retailing*, vol. 90, no. 2, pp. 217–232, 2014.

[40] A. Ghose and P. G. Ipeirotis, “Designing novel review ranking systems: predicting the usefulness and impact of reviews,” in *Proceedings International Conference on Electronic Commerce*, pp. 303–310, Minneapolis, MN, USA, August 2007.

[41] S. Xiao, C.-P. Wei, and M. Dong, “Crowd intelligence: analyzing online product reviews for preference measurement,” *Information & Management*, vol. 53, no. 2, pp. 169–182, 2016.

[42] L. Banic, A. Mihanovic, and M. Brakus, “Using big data and sentiment analysis in product evaluation,” in *Proceedings International Convention on Information and Communication Technology, Electronics and Microelectronics*, pp. 1149–1154, Opatija, Croatia, May 2013.

[43] K. Zhang, Y. Cheng, W.-k. Liao, and A. Choudhary, “Mining millions of reviews: a technique to rank products based on importance of reviews,” in *Proceedings of the International Conference on Electronic Commerce*, p. 12, Liverpool, UK, August 2012.

[44] A. Jøsang, R. Ismail, and C. Boyd, “A survey of trust and reputation systems for online service provision,” *Decision Support Systems*, vol. 43, no. 2, pp. 618–644, 2007.

[45] D. Bianculli, W. Binder, L. Drago, and C. Ghezzi, “Transparent reputation management for composite web services,” in *Proceedings of the IEEE International Conference on Web Services*, pp. 621–628, Beijing, China, September 2008.

[46] Z. Malik and A. Bouguettaya, “Rateweb: reputation assessment for trust establishment among web services,” *The VLDB Journal*, vol. 18, no. 4, pp. 885–911, 2009.

[47] S. Wang, Z. Zheng, Q. Sun, H. Zou, and F. Yang, “Evaluating feedback ratings for measuring reputation of web services,” in *Proceedings of the IEEE International Conference on Services Computing*, pp. 192–199, Washington, DC, USA, July 2011.

[48] L. Li and Y. Wang, “A subjective probability based deductive approach to global trust evaluation in composite services,” in *Proceeding of the IEEE International Conference on Web Services*, pp. 604–611, Washington, DC, USA, July 2011.

[49] O. Netzer, O. Toubia, E. T. Bradlow et al., “Beyond conjoint analysis: advances in preference measurement,” *Marketing Letters*, vol. 19, no. 3-4, pp. 337–354, 2008.

[50] M. Halme and M. Kallio, “Estimation methods for choice-based conjoint analysis of consumer preferences,” *European Journal of Operational Research*, vol. 214, no. 1, pp. 160–167, 2011.

[51] O. Toubia, M. G. De Jong, D. Stieger, and J. Füller, “Measuring consumer preferences using conjoint poker,” *Marketing Science*, vol. 31, no. 1, pp. 138–156, 2012.

[52] P. S. Fader and B. G. S. Hardie, “Modeling consumer choice among SKUs,” *Journal of Marketing Research*, vol. 33, no. 4, pp. 442–452, 1996.

[53] S. K. Hui, P. S. Fader, and E. T. Bradlow, “Path data in marketing: an integrative framework and prospectus for model building,” *Marketing Science*, vol. 28, no. 2, pp. 320–335, 2009.

[54] R. Decker and M. Trusov, “Estimating aggregate consumer preferences from online product reviews,” *International Journal of Research in Marketing*, vol. 27, no. 4, pp. 293–307, 2010.

[55] Y.-M. Li, H.-M. Chen, J.-H. Liou, and L.-F. Lin, “Creating social intelligence for product portfolio design,” *Decision Support Systems*, vol. 66, pp. 123–134, 2014.

[56] S. Boccaletti, V. Latora, Y. Moreno et al., “Complex networks,” *Structure and Dynamics*, vol. 424, no. 4-5, pp. 175–308.

[57] L. D. F. Costa, F. A. Rodrigues, and G. P. R. Travieso, “Characterization of complex networks: a survey of measurements,” *Advances in Physics*, vol. 56, no. 1, pp. 167–242, 2007.

[58] L. A. N. Villas Boas and J. M. Ottino, “Complex networks: augmenting the framework for the study of complex systems,” *European Physical Journal B—Condensed Matter*, vol. 38, no. 38, pp. 147–162, 2004.

[59] H. Wang, K. He, B. Li et al., “On some recent advances in complex software networks: modeling, analysis, evolution and applications,” *International Journal of Bifurcation and Chaos*, vol. 22, no. 2, 2012.

[60] S. C. Lin, P. Wang, I. F. Akyildiz et al., “Towards optimal network planning for software-defined networks,” *IEEE Transactions on Mobile Computing*, vol. 17, no. 12, pp. 2953–2967, 2018.

[61] L. G. Moyano, M. L. Mouronte, and M. L. Vargas, “Communities and dynamical processes in a complex software network,” *Physica A: Statistical Mechanics and Its Applications*, vol. 390, no. 4, pp. 741–748, 2011.

[62] W. Pan, B. Li, J. Liu, Y. Ma, and B. Hu, “Analyzing the structure of Java software systems by weightedK-core decomposition,” *Future Generation Computer Systems*, vol. 83, pp. 431–444, 2018.

[63] T. Wood, K. K. Ramakrishnan, J. Liu, and W. Zhang, “Toward a software-based network: integrating software defined networking and network function virtualization,” *IEEE Network*, vol. 29, no. 3, pp. 36–41, 2015.

[64] W. Pan, B. Song, K. Li, and K. Zhang, “Identifying key classes in object-oriented software using generalizedk-core decomposition,” *Future Generation Computer Systems*, vol. 81, pp. 188–202, 2018.

[65] W. Pan, H. Ming, C. Chang, Z. Yang, and D. K.. ElementRank, “Ranking Java software classes and packages using multilayer complex network-based approach,” *IEEE Transactions on Software Engineering*, 2019.

[66] E. Hai-Hong, M. N. Song, J. D. Song et al., “The research of service network based on complex network,” in *Proceedings of the international conference on Service Sciences (ICSS), 2010*, May 2010.

[67] S. Zhou and Y. Wang, “Clustering services based on community detection in service networks,” *Mathematical Problems in Engineering*, vol. 2019, Article ID 1495676, 11 pages, 2019.

[68] S. Dräxler, H. Karl, M. Peuster et al., “SONATA: Service Programming and Orchestration for Virtualized Software Networks,” in *Proceedings of the IEEE International Conference on Communications Workshops (ICC Workshops)*, Paris, France, May 2017.

[69] S. Correia, A. Boukerche, and R. I. Meneguette, “An architecture for hierarchical software-defined vehicular networks,”

IEEE Communications Magazine, vol. 55, no. 7, pp. 80–86, 2017.

[70] R. Ranchal, S. P. Singh, P. Angin et al., "RaaS and hierarchical aggregation revisited," in *Proceedings of the IEEE International Conference on Web Services*, June 2017.

[71] X. Zhang, Y. Zhou, Y. Ma, B. Chen, L. Zhang, and D. A.. Glmix, "Generalized linear mixed models for large-scale response prediction," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, San Francisco, CA, USA, August 2016.

[72] N. T. Longford, "Random coefficient models," in *Handbook of Statistical Modeling for the Social and Behavioral Sciences*, pp. 519–570, Springer, Berlin, Germany, 1995.

[73] P. Umberto, "Developing a price-sensitive recommender system to improve accuracy and business performance of ecommerce applications," *International Journal of Electronic Commerce Studies*, vol. 6, no. 1, p. 1, 2015.

[74] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, "Image-based recommendations on styles and substitutes," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval—SIGIR '15*, pp. 43–52, Santiago, Chile, July 2015.

Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 5346498, 7 pages
https://doi.org/10.1155/2020/5346498

Review Article

Modeling Software Systems as Complex Networks: Analysis and Their Applications

Hao Li,[1] **Tian Wang**,[1] **Xinxin Xu**,[1] **Bo Jiang**,[1] **Jianliang Wei**,[2] **and Jiale Wang**[1]

[1]*School of Computer Science and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, China*
[2]*School of Management Engineering and E-commerce, and Contemporary Business and Trade Research Center, Zhejiang Gongshang University, Hangzhou 310018, China*

Correspondence should be addressed to Jianliang Wei; 2356862895@qq.com

Received 10 January 2020; Accepted 25 February 2020; Published 27 April 2020

Guest Editor: Hua Ming

Software systems are of great importance, whose quality will influence every walk of our life. However, with increase in their scale and complexity, we are unable to control their quality since only little is known about their actual internal structure. "We cannot control what we cannot measure." Thus, to control these complex software systems, the first task that we should do is to measure their internal structure. In recent years, people applied the theories and techniques in the field of complex networks to systematically investigate the structure of software systems by representing software systems as networks (i.e., software networks), and many interesting and useful results have been revealed. In this work, we aim to briefly review some recent research advances in the interdisciplinary research between complex networks and software engineering, including modeling, analysis, and applications. Specifically, we first describe some novel techniques to model the structural details of a specific software system. Then, based on these modeling techniques, we introduce some research work on characterizing the static and dynamic structural properties of software systems. Third, we describe some promising applications of software networks in real-world scenarios. Finally, we suggest some future research topics.

1. Introduction

Nowadays, software systems have almost been used in every walk of life. Thus, how to provide a piece of software with high quality has been a problem attracting a lot of attention. However, with increase in the scale and complexity of software systems, it is a hard task to control the quality of a specific piece of software, especially when we know very little about the internal complexity of a specific software system [1, 2]. It is well known that we cannot control what we cannot measure. Thus, to control the quality of software systems, the first task that we should do is to measure their internal complexity [3]. Software structure which is defined as the software elements (e.g., attributes, methods, classes, interfaces, and packages) and their couplings (e.g., "method-call" couplings between methods and "inheritance" between classes) have been one of the most important factors that may influence the software complexity and further influence the quality of the software. Thus, how to measure and even to control the complexity of a software system has been a challenge faced by many researchers [4]. There is an urgent need to develop a systematic approach to deeply explore the internal structure of software systems.

Networks (or graphs) provide a natural and most adequate representation of the software structure; i.e., software elements are nodes and the couplings between software elements are edges (or links). Though network representation is not novel in software engineering, its form is simple and intelligible, which makes it feasible to perform the network analysis of software structure by using theories and techniques in the field of complex networks, and many significant discoveries and research results have been provided in the last decade [1, 2]. Note that such a network representation of the internal software structure of a specific software system is usually termed "software networks," a notion similar to "complex networks" [5].

In this work, we aim to briefly review some recent research advances in the field of software networks, highlighting different techniques in modeling, analysis, and applications. Specifically, we first describe some novel techniques to model the structural details of a specific software system. Then, based on these modeling techniques, we introduce some research work on characterizing the static and dynamic structural properties of software systems. Third, we describe some promising applications of software networks in real-world scenarios. Finally, we provide some future research topics. Note that, in this work, we only review the most recent research work which is published in the last seven years (2013 to 2019). For research work published before 2013, please refer to the reviews [1, 2]. In this work, we only focus on the brief review of the most recent research work in the field of software networks, rather than their detailed comparison.

The rest of this paper is organized as follows: Section 2 introduces the related reviews on software networks. Section 3 describes the data set we used. Section 4 introduces the related advances in software networks from three perspectives, i.e., modeling, analysis, and applications. Section 5 outlines some future research topics. Finally, in Section 6, we conclude the paper.

2. Related Work

To the best of our knowledge, there are a total of three reviews related to software networks.

Li et al. [6] reviewed 36 research papers related to software networks in 2008. They organized the existing work into three groups, i.e., related work on discoveries of software structural properties, related work on models of software growth, and related work on software metrics based on software networks. In the related work on discoveries of software structural properties, they discussed some research work on revealing shared structural properties in software networks. In the related work on models of software growth, they discussed the proposed evolution models to characterize the software growth. In the related work on software metrics, they discussed the metrics which are based on software networks and are used to characterize software quality.

Pan's review examined 32 research papers on software networks published before 2011 [2]. He discussed the existing research from four perspectives, i.e., characterization of software networks, modeling of software networks growth, measurement of software networks, and application of software networks in software engineering. In the "characterization of software networks," he reviewed the work that aims to characterize the properties of software structures such as scale-free and small world at different levels of granularity. In the "modeling of software networks growth," he reviewed the work that aims to propose an evolution model to explain the growth of software structures. In the "application of software networks in software engineering," he reviewed the work that applied software networks in software engineering practices such as software refactoring and software selection.

Sbelj and Bajec [1] also reviewed the related work on software networks. First, they reviewed the work on discovering the shared structural properties such as scale-free and small world phenomena. Second, they reviewed the work on characterizing the dynamical properties of software networks such as bug propagation. Third, they reviewed some work on the application of software networks such as refactoring and software abstraction.

Our current review is different from those of Li, Pan, and Sbelj. We cover a different time period from 2013 to 2019. Thus, our focus is to review the very recent research work in the field of software networks and shed some lights on the future research topic.

3. Data Set

We searched the 7 most popular digital libraries, i.e., ACM, IEEE, Springer, Scopus, ISI, ScienceDirect, and Compendex and Inspec, to obtain a relatively complete list of the primary research work. When searching the digital libraries, we used the following search string:

(Java OR OO OR object-oriented OR object oriented OR package OR packages OR class OR classes OR OR interface OR interfaces OR method OR methods OR attribute OR attributes) AND (software network OR software networks OR complex networks OR complex network OR graph OR graphs).

The search string contains the major research terms and their alternative spellings from the titles and keywords of related work on software networks. We use Boolean expressions "AND" and "OR" to connect the major research terms and their alternative spellings, respectively. Note that, in a specific digital library, the search string should be adapted slightly according to the grammar that library uses. For example, in the Springer library, the above string should be written as follows:

(Java OR OO OR "object-oriented" OR "object oriented" OR package OR packages OR class OR classes OR interface OR interfaces OR method OR methods OR attribute OR attributes) AND ("software network" OR "software networks" OR "complex networks" OR "complex network" OR graph OR graphs).

Obviously, the results returned by each digital library may overlap. Thus, we should identify and remove the redundant results. Furthermore, we also exclude research papers based on their titles, abstracts, and full texts. Finally, our data set contains 30 research papers (see the References section).

4. Analysis and Discussion

The research papers in our data set can be roughly categorized into three groups, i.e., modeling, analysis, and applications. Papers in the "modeling" category focus on the novel techniques to model the structural details of a specific software system. Papers in the "analysis" category focus on characterizing the static and dynamic structural properties of software systems. Papers in the "application" category try to apply the software networks to solve some real-world problems in software engineering. The three categories will be detailed in the following sections.

4.1. Modeling of Software Structure. Different types of software networks have been proposed to represent the structural details of a specific software system at different levels of granularity, such as associated software graphs [7], class diagram [8], and cyclic dependency graphs [9] (see Table 1). These software networks can be differentiated from the levels of granularity (i.e., package level, class level, and method (or attribute) level). Furthermore, these software networks can also be differentiated from the nature of couplings, i.e., whether their couplings are directed or weighted.

As is shown in Table 1, we can observe that, in the method level software networks, nodes represent methods and edges (or links) represent the method call relations between methods. We can use the frequency of method calls to weigh the edge (or link) with the aim to signify the coupling intensity that might exist between the two methods. Edge can also be directed to denote the coupling direction. The existing two software networks at the method level (i.e., weighted networks [10] and FCN [12]) are all not very accurate to describe the software structure. Weighted networks ignored the reference relations between methods and attributes, while FCN ignored the coupling direction. Thus, we can combine the two software networks to build a much more accurate software network, i.e., weighted directed feature coupling network (WDFCN). We use this feature to denote methods and attributes. In the WDFCN, nodes denote features, edges denote method class relations and method-attribute reference relations, weights on the edges denote the coupling frequencies, and the direction of edges denotes coupling directions.

In the class level software networks, nodes represent classes (or interfaces) and edges (or links) represent the couplings between classes (i.e., inheritance and implement) and couplings between the methods and attributes the classes contain (i.e., parameter, global variable, local variable, return type, and method call). Edges can also be assigned weights to signify the coupling intensity between classes (or interfaces) and can also be directed to denote the coupling direction. In the existing class-level software networks, CCN and MCN are much more accurate than others. However, they ignored two important coupling types between classes, i.e., the reference relations between methods and attributes the classes contain and the instantiate relations between classes. Thus, CCN and MCN can be improved by considering the two coupling types.

In the package-level software networks, nodes represent packages and edges (or links) represent the couplings between packages, which are derived from the couplings between classes. Edges can also be assigned weights to signify the coupling intensity and can also be directed to denote the coupling direction. In the existing two types of package-level software networks, PDN is much more accurate than MPN. However, as that in CCN and MCN, PDN and MPN also ignored two important coupling types between classes, i.e., the reference relations between methods and attributes the classes contain and the instantiate relations between classes. Thus, PDN and MPN can also be improved by considering the two coupling types.

Note that, compared with the research work on software networks published before 2013, the main difference of the software recently built is that they took into consideration much more information in the software systems such as different coupling types, coupling frequencies, and the nature of couplings. The software networks recently built are much more accurate. But they still ignored some information, e.g., the reference relations between methods and attributes the classes contain and the instantiate relations between classes. If the software networks we built are not very accurate, the results or findings that we obtained from experimental studies may contain errors. Thus, there is still much more work that we can do.

In fact, how accurately the software networks can describe the software systems depends on the tools that are used to extract the information enclosed in the software. To the best of our knowledge, many research papers only provide their software network models and show their results or findings. They usually do not mention the tools that they used to build software networks. Pan et al. [4, 5, 12] developed a software network analysis platform (SNAP) to build many types of software networks at different levels of granularity. Their tools can be obtained via the URLs provided in their work. By their tools, we can build all the abovementioned software networks.

4.2. Analysis of Software Networks. Chaikalis and Chatzigeorgiou [18] proposed a network-based prediction model to characterize the growth of software systems. Their model took into consideration both of the information from past data and domain-related rules.

Wang and Xiao [10] represented the runtime structure of the Linux operating system as a weighted network, where nodes represent functions and edges represent function calls. Based on the weighted network, they explored the execution process of Linux by using theories and techniques in complex networks. They found that the weight distribution follows a power-law distribution, the process management component of Linux plays the most important role, and the reliability of Linux declines with the versions from 3.15 to 4.4.

Yang et al. [11] modeled software systems as Function Call Networks (FCNs), where nodes represent functions and edges represent function calls. Based on the FCNs, they characterized the software structure using a set of measurements from the perspective of modularity, hierarchy, complexity, and fault propagation. They also proposed a model to quantify software structural quality, which gives a better understanding of the evolution of software systems.

Trindade et al. [19] represented software at the class level as Little House, where nodes represent classes and edges represent dependencies among classes. Based on the Little House, they analyzed 81 versions of 6 software systems and found some software evolution patterns. These patterns are

Table 1: Summary of the existing software networks.

Name	Level	Nodes	Couplings	Directed?	Weighted?
Weighted network [10], FCNs [11]	Method	Methods	Method call	Yes	Yes
FCN [12]	Method	Methods and attributes	Method call, method-attribute reference	No	Yes
Associated software graphs [7]	Class	Classes	Inheritance, composition, dependence	No	No
Class diagram [8]	Class	Classes	Dependency, common association, qualified association, association class, aggregation association, composition association, generalization, binding, generalization, realize	Yes	Yes
Cyclic dependency graphs [9]	Class	Classes and interfaces	Inheritance	No	No
DTMC [13]	Class	Classes	Method call, method-attribute reference	Yes	Yes
WCCN [3, 4, 12, 14]	Class	Classes and interfaces	Inheritance, implement, parameter, global variable, local variable, return type	No	Yes
CCN [15]	Class	Classes and interfaces	Inheritance, implement, parameter, global variable, local variable, method call, return type	Yes	Yes
MCN [5, 16]	Class	Classes and interfaces	Inheritance, implement, parameter, global variable, local variable, method call, return type	Yes	Yes
MPN [17]	Package	Packages	Inheritance, implement, global variable, method call	Yes	Yes
PDN [5]	Package	Packages	Inheritance, implement, parameter, global variable, local variable, method call, return type	Yes	Yes

further applied to define a software evolution model to characterize software evolution and growth.

Pan et al. [16] use a multilayer network at the class level to model software systems, where nodes are classes and interfaces and edges are different coupling types between classes (or interfaces). In their model, a specific type of coupling forms a layer. They used an aggregation approach to analyze the multilayer structure of a specific software system by using a set of 10 topological measures from complex networks. It is the first work to represent software systems as multilayer networks, providing a novel perspective to analyze software systems.

Note that the abovementioned papers on analysis of software networks used a traditional way to explore the growth of software systems from a structural perspective and also the structural properties enclosed in the software systems, but some of them took a new perspective. Specifically, Chaikalis and Chatzigeorgiou characterize software evolution from a network perspective, Wang and Xiao built the software network from execution process of the software, Yang et al. characterized the software structure by using the dynamic process of faults, Trindade and Orfano tried to use a model to characterize software evolution and growth, and Pan et al. used a multilayer networks, which is a much more accurate software network model.

4.3. Applications of Software Networks

4.3.1. Software Metrics. Gu et al. [20] proposed metrics to quantify the class cohesion from a complex network perspective.

Pan and Chai [3] modeled software systems at the class level as a weighted directed software network, and based on the network, they proposed a metric, NIN, to quantify the coupling intensity of two classes. They further proposed a metric to quantify the class stability. In [14], they further proposed a simulation way to calculate the software stability, which is based on the analysis of change propagation dynamics in the software structure.

In [12], Pan et al. modeled software systems as feature coupling networks (FCNs), where nodes are methods and attributes and edges are method-call relations and method-attribute reference relations. Based on the FCNs, they borrowed some idea from community detection techniques in complex networks and used the metric "modularity" to quantify the modularity of a specific software system.

Obviously, the recent research work on software metrics followed the traditional line of thoughts of the related work published before 2013. The only difference is they used a much more accurate software network model and characterized the software structure from a different perspective.

4.3.2. Bug Prediction. Concas et al. [7] used an Associated Software Graph (ASG) to represent software systems at the class level, where nodes represent classes and edges represent the "inheritance," "composition," and "dependence" relations between classes. Based on the ASG, they computed the number of communities, modularity of the software network, and other network metrics such as clustering coefficient, average path length, and mean degree. Then, they analyzed the correlation between these metrics and the number of bugs in the software. They found that medium-size systems with community structures tend to be buggy.

Yang et al. [11] proposed a software class network to represent software systems at the class level. In the software network, classes are nodes, and the calling relations between the methods that every pair of classes contain constitute the edges. Based on the software network, they proposed a set of metrics to characterize the software network structure and used some machine-learning algorithms to construct defect prediction models. Their results showed promising results.

Zakari et al. [21] proposed a software network at the statement level, where statements are nodes and the execution traces between statements are edges. Based on the software network, they computed two centrality metrics (i.e., degree centrality and closeness centrality) for fault diagnosis. Experimental results showed their approach is promising and better than existing fault localization techniques.

Obviously, in the existing research work on fault prediction, software networks are usually used to calculate some structural metrics and the structural metrics can be used to correlate with bugs or be used in traditional prediction models to improve fault prediction performance. However, the software networks the existing approaches used are not very accurate, which makes the metrics obtained inaccurate. Thus, in the future, we can use a much more accurate software network to compute structural metrics.

4.3.3. Software Refactoring. Pan et al. [22] modeled the software structure at the method level as SFN, where nodes represent methods and attributes and edges represent method-call relations and method-attribute reference. Then, they applied an evolutionary algorithm to optimize software structure and detect the methods to be moved. In their algorithm, they optimized a function which is based on software modularity. In [23], Pan et al. proposed a similar approach to identify the classes to be moved.

In [24], Wang et al. represented software at the class level as a Class-Level Multirelation Directed Network (CMDN), where nodes are classes and edges are the coupling between classes, i.e., inheritance, association, and aggregation. Based on the CMDN, they used the community detection algorithm to identify many refactoring opportunities simultaneously. Experimental results showed that their approach is better than some existing approaches.

There are many other refactorings in object-oriented software systems. However, the existing work only considered three refactorings, i.e., move method refactoring, move field refactoring, and extract class refactoring. Many other refactorings such as extract method, pull up method, and inline class need further exploration.

4.3.4. Key Class Identification. Meyer et al. [25] modeled software systems at the class level as a software network and applied the coreness in the k-core decomposition to measure class importance in the software network. The coreness is further used as a criterion to rank classes.

Şora and Chirila [26] recently modeled software systems as graph, where nodes represent classes and edges represent the couplings between classes. Weights are assigned to the edges to measure the coupling intensity. Then, they applied PR-U2-W, CONN-TOTAL, and CONN-TOTAL-W to measure class importance, respectively.

In [27], Luo et al. proposed an extend call graph to represent methods and their calling relations and utilized a VertexRank algorithm to quantify the importance of methods.

In [28], He et al. modeled software systems as a weighted software network, where nodes are methods and edges are their calling relations. Then, they applied a PageRank-like algorithm to quantify the importance of methods.

In [15], Pan et al. modeled software systems at the class level as a weighted directed software network, where nodes are classes and interfaces, edges are the 7 types of couplings between classes (or interfaces), and the weights on the edges are the coupling frequencies. Then, they proposed a generalized k-core decomposition to quantify the importance of classes. In [5], they further proposed a multilayer software network at the class level. Based on the software network, they compute the importance of classes at each layer and further combine the class importance at each layer to obtain the final importance.

The software network proposed in [5] is the best accurate one in the existing research work. But the authors ignored two important types of couplings, i.e., the reference relations between methods and attributes the classes contain and the instantiate relations between classes. Thus, there is still much room for improving the existing work on key classes identification. We can also use improved ranking algorithm to improve the performance of the existing approaches. To the best of our knowledge, there is no work on identifying important software elements at other levels of granularity.

5. Future Research Topics

Based on the brief review of the related work on software network, we proposed the following research topics that we can carry out in the future:

(i) Much more accurate software networks at different levels of granularity: for example, in the existing software networks, no one considered all the coupling types that might exist between classes. Thus, much work should be performed to consider much more information in the software.

(ii) Runtime software networks: the majority of the software networks is constructed statically from the source code or bytecode of a specific software system. Only one research paper [10] reported constructing a runtime software network. Thus, much work can be carried out on runtime software network modeling, analysis, and applications.

(iii) Software evolution model: software evolution models are used to characterize the software evolution and growth. They should reflect the properties enclosed in software systems. Thus, if we find more properties of software, the evolution model can be updated.

(iv) Bug prediction: we can propose many software metrics to characterize software structure and further use them to improve any bug prediction models.

(v) Software refactoring: much more work can be proposed to identify other refactoring opportunities such as extract methods, pull up methods, and inline classes.

(vi) Software comprehension: identifying important software elements can be used to aid people understand a specific software system. Much more work can be carried out on identifying important software elements at other levels of granularity (i.e., package level, method level, or even statement level). Furthermore, much more work can also be performed to guide the specific comprehension process of a software system.

(vii) Service-oriented system analysis: software networks have also been used in service-oriented software systems. Pan et al. [29, 30] used software networks to represent API and their couplings in service-oriented systems and applied community detection algorithm to organize APIs into clusters. Thus, in the future, we can also perform service-oriented software modeling, analysis, and applications.

6. Conclusions

This paper briefly reviewed the recent advances in the research field of software networks from 2013 to 2019. First, we described the data set we used, i.e., the research work published in the time period of 2013 to 2019. Then, we briefly described the existing work from three perspectives, i.e., modeling, analysis, and applications. Specifically, we reviewed the software networks that used to model the structural details of specific software systems and highlighted the problems in the existing models. We briefly introduced some research work on characterizing the static and dynamic structural properties of software systems. We also described some promising applications of software networks in real-world scenarios such as software metrics, bug prediction, refactoring, and key element identification. Finally, we outlined some future research topics.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Key R&D Program of Zhejiang Province (Grant nos. 2019C01004 and 2019C03123) and the Humanities and Social Science Foundation of Chinese Ministry of Education (Grant nos. 17YJA870020 and 18YJA870010).

References

[1] L. Šubelj and M. Bajec, "Software systems through complex networks science: review, analysis and applications," in *Proceedings of the First International KDD Workshop on Software Mining (SoftwareMining 2012)*, pp. 9–16, Beijing, China, August 2012.

[2] W. F. Pan, "Applying complex network theory to software structure analysis," *World Academy of Science, Engineering and Technology*, vol. 60, pp. 1636–1642, 2011.

[3] W. Pan and C. Chai, "Measuring software stability based on complex networks in software," *Cluster Computing*, vol. 22, no. s2, pp. 2589–2598, 2019.

[4] W. Pan, B. Li, J. Liu, Y. Ma, and B. Hu, "Analyzing the structure of Java software systems by weighted K-core decomposition," *Future Generation Computer Systems*, vol. 83, pp. 431–444, 2018.

[5] W. F. Pan, H. Ming, C. K. Chang, Z. J. Yang, and D.-K. Kim, "ElementRank: ranking Java software classes and packages using multilayer complex network-based approach," *IEEE Transactions on Software Engineering*, p. 1, 2019.

[6] B. Li, Y. T. Ma, J. Liu et al., "Advances in the studies on complex networks of software systems," *Advances in Mechanics*, vol. 38, no. 6, pp. 805–814, 2009.

[7] G. Concas, C. Monni, M. Orru et al., "Software systems through complex networks science: review, analysis and applications," in *Proceedings of the 4th International Workshop on Emerging Trends in Software Metrics (WeTSOM 2013)*, pp. 14–20, San Francisco, CA, USA, May 2012.

[8] C. Y. Chong and S. P. Lee, "Analyzing maintainability and reliability of object-oriented software using weighted complex network," *Journal of Systems and Software*, vol. 110, pp. 28–53, 2015.

[9] T. D. Oyetoyan, J. R. Falleri, J. Dietrich, and K. Jezek, "Circular dependencies and change-proneness: an empirical study," in *Proceedings of the 22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER 2015)*, pp. 241–250, Montreal, Canada, March 2015.

[10] H. Wang and G. Xiao, "Analysis of the runtime Linux operating system as a complex weighted network," in *Proceedings of the 2016 International Conference on Software Analysis, Testing and Evolution (SATE 2016)*, pp. 7–11, Kunming, China, November 2016.

[11] Y. Yang, J. Ai, X. Li, and W. E. Wong, "MHCP model for quality evaluation for software structure based on software complex network," in *Proceedings of the IEEE 27th International Symposium on Software Reliability Engineering (ISSRE 2016)*, pp. 298–308, Ottawa, Canada, October 2016.

[12] Y. Xiang, W. Pan, H. Jiang, Y. Zhu, and H. Li, "Measuring software modularity based on software networks," *Entropy*, vol. 21, no. 4, p. 344, 2019.

[13] S. M. Srinivasan, R. S. Sangwan, and C. J. Neill, "On the measures for ranking software components," *Innovations in Systems and Software Engineering*, vol. 13, no. 2-3, pp. 161–175, 2017.

[14] W. F. Pan, H. B. Jiang, H. Ming, C. Chai, B. Chen, and H. Li, "Characterizing software stability via change propagation simulation," *Complexity*, vol. 2019, Article ID 9414162, 17 pages, 2019.

[15] W. Pan, B. Song, K. Li, and K. Zhang, "Identifying key classes in object-oriented software using generalizedk-core decomposition," *Future Generation Computer Systems*, vol. 81, pp. 188–202, 2018.

[16] W. Pan, B. Hu, J. Dong, K. Liu, and B. Jiang, "Structural properties of multilayer software networks: a case study in Tomcat," *Advances in Complex Systems*, vol. 21, no. 2, Article ID 1850004, 2018.

[17] W. F. Pan, B. Li, Y. T. Ma et al., "Identifying the key packages using weighted PageRank algorithm," *Acta Electronica Sinica*, vol. 42, no. 11, pp. 2174–2183, 2014, in Chinese.

[18] T. Chaikalis and A. Chatzigeorgiou, "Forecasting Java software evolution trends employing network models," *IEEE Transactions on Software Engineering*, vol. 41, no. 6, pp. 582–602, 2015.

[19] R. P. F. Trindade, T. S. Orfano, K. A. M. Ferreira, and E. F. Wanner, "The dance of classes - a stochastic model for software structure evolution," in *Proceedings of the 4th*

International Workshop on Emerging Trends in Software Metrics (WeTSOM 2017), pp. 22–28, Buenos Aires, Argentina, May 2017.

[20] A. Gu, X. Zhou, Z. Li, Q. Li, and L. Li, "Measuring object-oriented class cohesion based on complex networks," *Arabian Journal for Science and Engineering*, vol. 42, no. 8, pp. 3551–3561, 2017.

[21] A. Zakari, S. P. Lee, and C. Y. Chong, "Simultaneous localization of software faults based on complex network theory," *IEEE Access*, vol. 6, pp. 23990–24002, 2018.

[22] W. F. Pan, J. Wang, and M. C. Wang, "Identifying the move method refactoring opportunities based on evolutionary algorithm," *International Journal of Modelling, Identification and Control*, vol. 18, no. 2, pp. 182–189, 2013.

[23] W.-F. Pan, B. Jiang, and B. Li, "Refactoring software packages via community detection in complex software networks," *International Journal of Automation and Computing*, vol. 10, no. 2, pp. 157–166, 2013.

[24] Y. Wang, H. Yu, Z. Zhu, W. Zhang, and Y. Zhao, "Automatic software refactoring via weighted clustering in method-level networks," *IEEE Transactions on Software Engineering*, vol. 44, no. 3, pp. 202–236, 2018.

[25] P. Meyer, H. Siy, and S. Bhowmick, "Identifying important classes of large software systems through K-core decomposition," *Advances in Complex Systems*, vol. 17, no. 07n08, Article ID 1550004, 2014.

[26] I. Şora and C.-B. Chirila, "Finding key classes in object-oriented software systems by techniques based on static analysis," *Information and Software Technology*, vol. 116, Article ID 106176, 2019.

[27] H. Luo, Y. Dong, Y. Y. Ng, and S. Wang, "VertexRank: importance rank for software network vertices," in *Proceedings of the 2014 IEEE 38th Annual Computer Software and Applications Conference (COMPSAC 2014)*, pp. 251–260, Vasteras, Sweden, 2014.

[28] H. He, C. Shan, X. Tian, Y. Wei, and G. Huang, "Analysis on influential functions in the weighted software network," *Security and Communication Networks*, vol. 2018, Article ID 1525186, 10 pages, 2018.

[29] W. Pan and C. Chai, "Structure-aware mashup service clustering for cloud-based internet of things using genetic algorithm based clustering algorithm," *Future Generation Computer Systems*, vol. 87, pp. 267–277, 2018.

[30] W. Pan, J. Dong, K. Liu, and J. Wang, "Topology and topic-aware service clustering," *International Journal of Web Services Research*, vol. 15, no. 3, pp. 18–37, 2018.

Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 4984375, 16 pages
https://doi.org/10.1155/2020/4984375

Research Article

HyOASAM: A Hybrid Open API Selection Approach for Mashup Development

Bo Jiang, Pengxiang Liu, Ye Wang, and Yezhi Chen

School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, China

Correspondence should be addressed to Ye Wang; yewang@zjgsu.edu.cn

Received 30 December 2019; Revised 20 March 2020; Accepted 1 April 2020; Published 25 April 2020

Guest Editor: Hua Ming

At present, Mashup development has attracted much attention in the field of software engineering. It is the focus of this article to use existing open APIs to meet the needs of Mashup developers. Therefore, how to select the most appropriate open API for a specific user requirement is a crucial problem to be solved. We propose a Hybrid Open API Selection Approach for Mashup development (HyOASAM), which consists of two basic approaches: one is a user-story-driven open API discovery approach, and the other is multidimensional-information-matrix- (MIM-) based open API recommendation approach. The open API discovery approach introduces user stories in agile development to capture Mashup requirements. First, it extracts three components from user stories, and then, it extracts three corresponding properties from open API descriptions. Next, the similarity calculation is performed on two sets of data. The open API recommendation approach first uses MIM to store open APIs, Mashups, and the invoking relationship between them. Second, it enters the matrix obtained in the previous step into a factorization machine model to calculate the association scores between the Mashups and the open APIs, and TOP-N open API lists for creating the Mashup are obtained. Finally, experimental comparison and analysis are carried out on the PWeb dataset. The experimental results show that our approach has improved significantly.

1. Introduction

Unlike object-oriented software engineering [1, 2], service-oriented software engineering (SOSE) is used to design, develop, and maintain software systems [3] that use the principles of service-oriented architecture (SOA) [4]. Open APIs are the basis of SOSE [5]. Mashup development is a novel development practice of SOSE for building multiservice applications by integrating single-function open APIs, which becomes more and more popular. Mashup refers to a temporary combination of web applications that allows users to create entirely new APIs using content retrieved from external data sources [6]. As Mashup application developers face the explosive growth of open APIs on the Internet, they often suffer from the overload of API information.

At present, there are a large number of open APIs over the Internet with similar descriptions but different functionalities and qualities, which undoubtedly affect Mashup application developers' decisions. In addition, unstructured description documents of open APIs increase the difficulty of semantic extraction. All of the above problems make it more and more difficult for developers to select the appropriate high-quality open APIs to build Mashup applications. Therefore, a major challenge to Mashup application development has emerged [7]: how to effectively and efficiently select the most appropriate open APIs from a large number of available resources to match the needs of Mashup developers.

In response to the above challenges, a number of open APIs selection approaches have been proposed, including keyword-based discovery approaches [8, 9], topic model-based discovery approaches [10, 11], content-based recommendation approaches [12, 13], and QoS-based recommendation approaches [14, 15]. Yet, there are some problems with these studies: (1) Most established approaches use nonnatural language (NL) text to describe Mashup requirements [16], such as WSDL, which is not user friendly. The existing techniques do not work well with text modeling. (2) Some users do not even know how to describe the requirements of Mashup exactly [17], which makes the API search based on keywords difficult. Therefore, a comprehensive open API selection approach should not only include searching or discovering APIs based on keywords but also include actively recommending APIs to developers based on his/her preference. Yet, current approaches separate open API discovery from open API recommendation, leading to inefficient results.

To overcome the above problems, we propose a Hybrid Open API Selection Approach for Mashup development (HyOASAM), which consists of two basic approaches: one is a user-story-driven open API discovery approach, and the other is a multidimensional-information-matrix- (MIM-) based open API recommendation approach. The user-story-driven open API discovery approach is to tackle the first problem. By introducing user stories into Mashup development, the Mashup developers can easily capture the role, aim, and motivation of a Mashup and then describe them with NL-based user stories. In agile development, user stories are used to capture and describe the rapidly changing user requirements. The MIM-based open API recommendation approach is to tackle the second problem. We make use of the historical information of Mashup developers (such as the search history and the access history) to profile each developer, elicit their preferences, and then recommend the most suitable open APIs to them. The open API discovery approach can be divided into three steps: (1) extracting three components from user stories, (2) extracting three corresponding elements from open API descriptions, and (3) calculating the similarity based on two sets of data. The open API recommendation approach can be divided into two steps: (1) extracting open APIs, Mashups, and the invoking relationships between them using MIM and (2) calculating the association scores between the Mashup and the open APIs using a factorization machine (FM) model to recommend TOP-N open APIs. Our approach can perform attribute extraction well for both long and short texts, and deeper associations can be better mined through FM.

In summary, the contributions of this paper are as follows:

(1) We propose a novel hybrid open API selection approach, HyOASAM, enabling developers to quickly and accurately find their wanted open APIs by both discovering APIs and recommending APIs.

(2) We propose an approach that breaks the restrictions of open API documents, described by user stories, which can be used to capture and describe Mashup developers' requirements more accurately and effectively.

(3) We tailor the current MIM matrix [26] and introduce factorization machine (FM) into the open API recommendation approach to calculate the semantic similarity more accurately.

(4) We validate the effectiveness of HyOASAM through various evaluation criteria based on the real data of PWeb. The evaluation results show that HyOASAM has a better improvement over other approaches.

The rest of the paper is organized as follows: Section 2 introduces previously established open API discovery and recommendation approaches; Section 3 introduces our proposed HyOASAM in detail; Section 4 compares the HyOASAM with other approaches; Section 5 draws the conclusions.

2. Related Work

In this work, we treat open APIs and services as synonyms and use the two concepts interchangeably.

2.1. Open APIs Discovery.

2.1. Open APIs Discovery. Open APIs discovery is the efficient and accurate retrieval of a set of open APIs or services that achieve the needs of users from a service database based on the demand statements entered by users [18]. Generally, the discovery approaches can be categorized into the following two main classes.

2.1.1. Syntax-Level Discovery Approaches. Syntax-level open API discovery is the earliest proposed discovery technique. It matches through several keywords and the grammatical features of the service interface, and the matching mechanism is relatively simple. Typical approaches are as follows.

Massimo and Erl [8] proposed a solution based on DAML-S (DARPA Agent Markup Language) to perform semantic matching between user requirements and service description. Mateos et al. [19] use metaprogramming and other related technologies to develop a set of tools for text mining and service processing of WSDL documents. Paliwal et al. [20] proposed clustering services by clustering approaches based on service descriptions and service registration information on UDDI and then used latent semantic indexing (LSI) to achieve matching. Elgazzar et al. [21] improved the accuracy of service discovery by searching key information such as content, service types, messages, and ports in the WSDL document and used Quality Threshold (QT) clustering algorithms to group services based on key information.

In general, the syntax-level open APIs discovery approaches are relatively simple to implement and easy to maintain. However, the deep semantics cannot be understood using such approaches. For example, the polysemy is a common problem, which inevitably leads to a low precision.

2.1.2. Semantic-Level Discovery Approaches. Ontology is used to solve the heterogeneity of grammatical-level service descriptions at the early stage, so that the semantic description of services functions and behaviors is strengthened. The matching algorithms in semantic-level service discovery rely on logical deduction and reasoning. The continuous development of artificial intelligence makes the service discovery algorithms smarter, faster, and accurate.

Ke et al. [22] transformed user requirements and service description documents into ontology trees; calculated the conceptual similarity, attribute similarity, and structural similarity of corresponding nodes in a hierarchical and classified manner; and thereafter effectively avoided complex reasoning. Huang et al. [23] proposed a semantic similarity calculation approach based on ontology distance calculation. The AGNES clustering algorithm clustered semantic service sets to improve the efficiency of service discovery. Klusch [24] used service profiles to select services described by OWL-S on a semantic level. Wei et al. [25] proposed a customizable SAWSDL service matcher that extends XQuery by using various similarity measures to support multiple matching strategies based on different application requests.

Most of the above open API discovery approaches cannot meet the dynamic changing needs of Mashup developers to provide service choices. Moreover, most of the Mashup developers' requirement descriptions are inaccurate and cannot describe their real needs better. This will greatly affect the results of service discovery. In traditional topic-based data extraction models, such as LDA, TF-IDF, HDP, and other topic models, the topic extraction method is the generalization of the user's overall needs, and the extracted data will have a certain deviation from the actual needs of the user; that is, it cannot describe user preferences and needs.

2.2. Open APIs Recommendation. With the rapid increase of open APIs, some APIs that are of interest to Mashup developers are difficult to search because of the small number of visits. On the other hand, developers often lack reasonable and effective requirement description skills. In such cases, open APIs discovery cannot be applied appropriately. Open APIs recommendation maintains the ecology of the service platform to tackle the problem. At present, the existing recommendation research work can be roughly divided into the following three categories.

2.2.1. Recommendation Based on Functional Characteristics. The functional characteristics are mainly extracted from service description documents, and the most similar services are recommended by measuring the similarity between the description documents. For example, Cao et al. [26] used the topic model to calculate the relationship between Mashup, services, and the invoking relationship between them. By integrating the popularity of the service into the model, they predicted the link between the Mashup and the service and then recommended the appropriate service for the Mashup developer. Most service recommendation approaches based on functional characteristics adopt traditional topic models or keywords for similarity calculation. However, traditional topic models need to specify the number of topics in advance while extracting topic vectors, which has a direct impact on the recommendation results.

2.2.2. Recommendation Based on Quality of Service (QoS). QoS refers to the nonfunctional features of the service, such as the user's history of invoking services or the quality of services. Zheng et al. [27] used collaborative filtering to calculate the quality of services through user historical behaviors. Huang et al. [28] proposed a Mashup component recommendation approach to establish a relationship between Mashup components through a generic layer model and guide users to select components from a large-scale Mashup component library. Xu et al. [29] proposed a socially perceived approach, in which the coupling matrix model was used to store the multidimensional social network between potential users, topics, Mashups, and services, and the relationships were predicted by existing relational networks. However, those approaches have the problem of matrix sparsity, which affects the recommendation accuracy.

2.2.3. Recommendation Based on Hybrid Characteristics. Such approaches take into consideration not only the functional characteristics of the service but also the QoS. By combining the two characteristics, the accuracy of service recommendation is improved. For example, Gao et al. [30] proposed a manifold ordering framework. Based on the similarity between Mashups and the heterogeneous relationship between Mashups and services, a manifold ranking algorithm is applied to recommendation services. The similarity between Mashups and the heterogeneous relationship between Mashups and services are calculated by a manifold sorting algorithm. Li et al. [31] proposed an approach for integrating multidimensional information, using HDP to extract service, and the subject vector of Mashups was used to calculate the similarity between Mashups, the similarity between services, the fluency of services, and the co-occurrence of services. Then they used the FM model to score and recommend the highest rated N services to Mashup developers. Xia et al. [32] proposed a new class-aware service clustering and distributed approach. First, the services were clustered by extending the *K*-means clustering algorithm, and then, the service ordering was predicted through a distributed machine learning framework. At present, the service recommendation approaches based on hybrid characteristics are among research hotspots, due to its high precision. HyOASAM has taken the advantage of such approaches.

From Table 1, we can see that the established open APIs discovery approaches have the problem of the randomness of user demand descriptions and open APIs description texts, which leads to unsatisfactory results, whereas open APIs recommendation approaches have the problem of inability to fully mine hidden information. Besides, it is hard to meet the real needs of developers to take either the discovery approaches or the recommendation approaches. A hybrid manner is a more accurate and comprehensive manner.

3. HyOASAM Approach

If a Mashup developer enters a user story of a requirement "as a user, I want to upload and edit photos online so that I can process photos on the server," then user-story-driven open APIs discovery approach is applied to calculate the similarity

Table 1: The comparison of related work.

Approach	Category	Pros	Cons
Service discovery based on DAML-S [8]	Open API discovery	The earliest service described by DARPA agent markup language	The randomness of user demand descriptions and service description texts leads to unsatisfactory results
Service discovery based on text mining [19]		It combines text mining and metaprogramming techniques	The approach is unable to mine deep relationships
Web service discovery based on an ontology [20, 22, 24]		They address the issue of nonexplicit service description semantics that match a specific service request	The semantic extension is not enough
Web service discovery based on WSDL documents clustering [21]		Narrowing the search space and improving results	Each feature is not assigned its own weight
Web service discovery based on hierarchical clustering [23]		The vector space model improves the accuracy and efficiency	It does not take the semantics into consideration
Web service discovery based on SAWSDL-iMatcher [25]		Multiple matching strategy extensions via XQuery can effectively aggregate similar values	The approach is only useful in one specific domain, not effective in other domains
Open API recommendation based on topic model [26, 31]	Open API recommendation	The document probability distribution is obtained, and the distance is used to calculate the semantic distance	The topic model is not well trained
Web service recommendation based on collaborative filtering [27]		Collaborative filtering does not require specialized domain knowledge and can be easily modeled	Collaborative filtering cannot mine hidden information
Model-based recommendation [28]		The use of a generic hierarchical graph model can improve efficiency and effectiveness	This approach cannot get synthesis of multiple constraints for more personalized recommendation
Social-aware recommendation [29]		It can predict unobserved relationships	The matrix sparsity affects accuracy
Manifold-learning-based recommendation [30]		Mashup can use manifold sorting algorithm for better clustering	The approach cannot handle dynamically added services
Combining machine learning and distributed recommendation [32]		More accurate prediction	The approach ignores QoS
HyOASAM	Open API discovery and recommendation	HyOASAM can handle random description text and make accurate recommendations for unclear user needs description	The modeling process is a little more complicated than other approaches

between the requirement and the open API description text and returns a list of open APIs for developers according to the level of similarities. If no requirements are entered, the MIM-based open API recommendation approach is applied to calculate the score between the developer's profiles including the Mashup usage and the information of the open APIs and returns a list of open APIs for developers to choose from based on the score. Below we will use the example to illustrate the whole process of HyOASAM.

The framework of the HyOASAM approach is shown in Figure 1. HyOASAM takes two scenarios into account: (1) When the Mashup developer can describe his/her requirements with user stories, the user-story-driven open API discovery approach is applied to return a list of open APIs to the developer. (2) When the Mashup developer does not input any requirement, the MIM-based open API recommendation is applied to return a list of open APIs to the developer.

3.1. User-Story-Driven Open APIs Discovery Approach. The user-story-driven open APIs discovery approach analyzes syntactic dependencies [33] of user stories by the Mashup developer's requirements and then uses NLP technology to extract the requirements components (Step 1). At the same time, open API properties are extracted in the open API description (Step 2). Then, it calculates the similarity between the requirement components and open API properties. Finally, it sorts the open APIs by the similarity to get the open API list with the most similar N for developers to choose from (Step 3). Figure 2 shows the overall framework of the open API discovery approach.

3.1.1. Requirements Components Extraction (Step 1)

Definitions. Developer requirement descriptions are often too casual, so we use user story to describe open API requirement components [34]. For example, "as a user, I want to upload and edit photos online so that I can process photos on the server". Requirement components are the key information of open API requirements, and the detailed description is shown as follows:

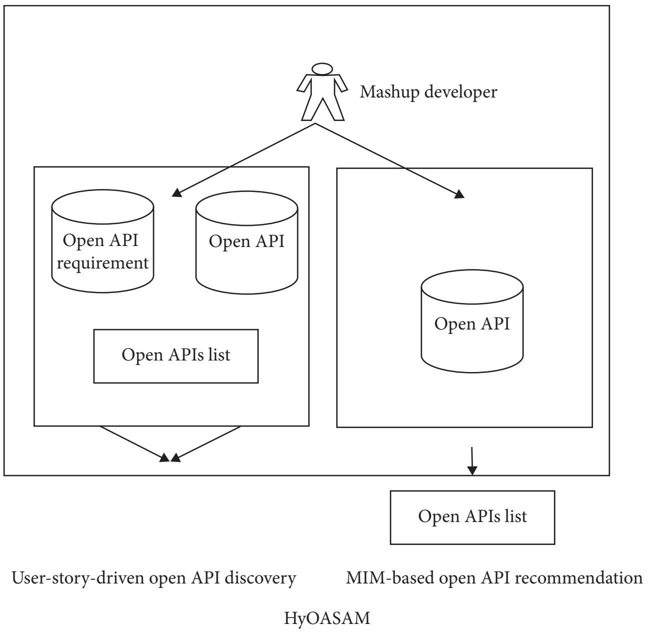


Figure 1: Framework of HyOASAM.

(1) Role (ro) = <noun, adj>: the role to carry out the functionality. The noun is the role, and the adj is a modification of the noun. In the above example, the role is <user, null>.

(2) Aim (ai) = <verb, do, io, adj>: the aim that developers want to achieve. Verb is act of a role, do is the direct object, io is the indirect object, and adj is the extension of the corresponding noun. In the above example, the aim is <{upload, edit}, photo, null, online>.

(3) Motivation (mo) = <verb, do, io, adj>: the developers' purpose. Components in mo and go are the same. In the above example, the motivation is <process, photo, server, null>.

Definition 1 (requirements components). rc = <ro, ai, mo>. Requirements component represents developers' actual needs and is composed of role, aim, and motivation.

In the process of requirements components extraction, we tag each word in the user story, because polysemy will affect the final result. Then, we extract requirement components based on grammatical dependencies. We use Stanford Parser [35] to parse text and extract Stanford Dependency (SD) set. Proceed as follows:

(i) *Role extraction*. We have found through several experiments that the following three SDs can completely extract the components of role, including pobj(As, dep), nn(gov, dep), and amod(gov, dep). Readers can refer to the white paper of Stanford Parser [35] for the detailed explanation of each SD.

(ii) *Aim extraction*. The following eight SDs and their combinations can extract all aim components, including xcomp(want, dep), dobj(gov, dep), iobj(gov, dep), conj(gov, dep), pobj(gov, dep), nn(gov, dep), and amod(gov, dep).

(iii) *Motivation extraction*. As aim and motivation have similar structures, their components extraction processes are similar. We only need to change the first SD here into aux(dep, can).

For example, when the Mashup developer enters the requirement "as a user, I want to upload and edit photos online so that I can process photos on the server," through Step 1, the final extraction result of the requirement component is like this: <user, null>, <{upload, edit}, photo, null, online>, <process, photo, server, null>.

3.1.2. Open API Properties Extraction (Step 2). Next, we extract open API properties from the open API description text. The open API description text is generally a text describing the API function written by the API developer. It is mainly a text that helps the developer understand the API and how to use it. Currently, the open API description text is written in NL, for example, "customers can use the service to edit photos and video over the Internet." Open API properties contain the following properties:

(1) Agent (ag) = <noun, adj>. The subject that the open API is served to. In the above example, the agent is <customer, null>.

(2) Activity (ac) = <verb, do>. The activity provided by the open API. In the above example, the activity is <edit, {photo, video}>.

(3) Scenario (sc) = <io, adj>. The scenario of the open API. In the above example, the scenario is <Internet, over>.

Definition 2 (open APIs properties). oap = <ag,ac,sc> represents the properties of the entire open APIs.

As the style of the open API description text is not limited, we extract the following 14 SDs to comprise open API properties: nsubj, nsubjpass, xsubj, agent, csubj, csubjpass, cop, nn, dobj, iobj, prep&pobj, pobj, amod, and conj.

For example, the open API description text is "customers can use the service to edit video and photos over the Internet." Through Step 2, the final result of open API properties is <customer, null>, <edit, {photo, video}>, <Internet, over>.

3.1.3. Similarity Calculation (Step 3). This section presents the similarity formula between the user story q and open API s through requirements components and open APIs properties.

The overall formula is as follows:

$$\operatorname{sim}(q,s) = a \times \operatorname{usim}\left(u_q, u_s\right) + b \times \operatorname{asim}\left(a_q, a_s\right) + c \times \operatorname{gsim}\left(g_q, g_s\right), \tag{1}$$

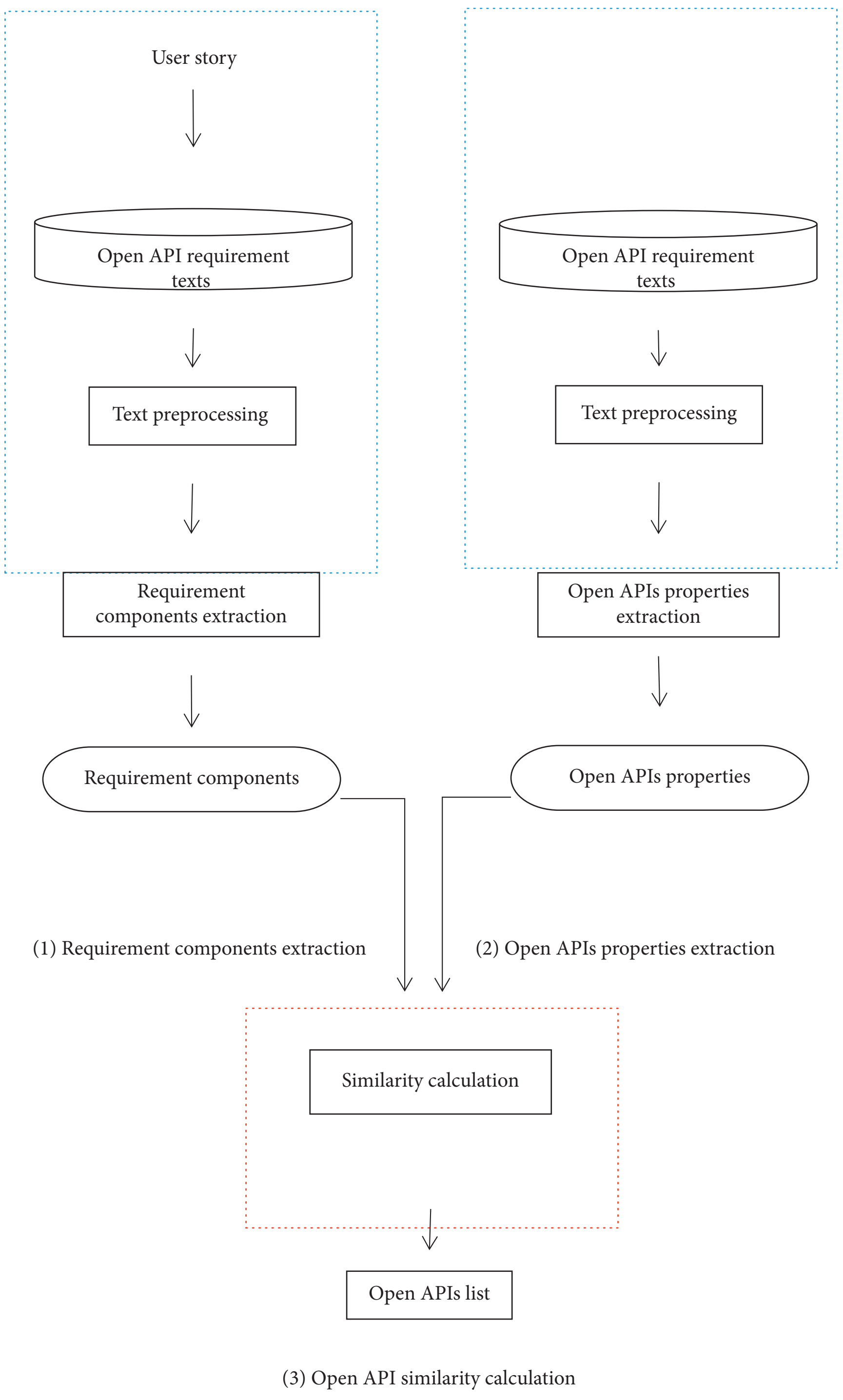

User story
Open API requirement texts
Text preprocessing
Requirement components extraction
Requirement components
Open API requirement texts
Text preprocessing
Open APIs properties extraction
Open APIs properties
(1) Requirement components extraction
(2) Open APIs properties extraction
Similarity calculation
Open APIs list
(3) Open API similarity calculation

Figure 2: Framework of the user-story-driven open API discovery approach.

where $\text{usim}(u_q, u_s)$ represents the similarity between the role components u_q in user story q (e.g., <user, null>) and the agent properties u_s in open API description text s (e.g., <customer, null>); $\text{asim}(a_q, a_s)$ represents the similarity between part of aim and motivation components a_q (verb and do) in user story q (e.g., <{upload, edit}, photo>,<process, photo>) and the activity properties a_s (verb and do) in open API description text s (e.g., <edit, {photo, video}>); $\text{gsim}(g_q, g_s)$ represents the similarity between part of aim components g_q (io and adj) in user story u (e.g., <null, online><server, null>) and scenario properties g_s (io and adj) in open API description text s (e.g., <Internet, over>). The parameters a, b, and c represent the weight of the three variables in

(1), and $a + b + c = 1$. The specific formula is as follows:

$$\text{usim}\left(u_q, u_s\right) = \begin{bmatrix} \text{sim}\left(w_{q_1} w_{s_1}\right) & \cdots & \text{sim}\left(w_{q_1} w_{s_j}\right) \\ \vdots & \ddots & \vdots \\ \text{sim}\left(w_{q_k} w_{s_1}\right) & \cdots & \text{sim}\left(w_{q_k} w_{s_j}\right) \end{bmatrix}, \quad (2)$$

$$\text{gsim}\left(g_q, g_s\right) = \begin{bmatrix} \text{sim}\left(w_{q_1} w_{s_1}\right) & \cdots & \text{sim}\left(w_{q_1} w_{s_j}\right) \\ \vdots & \ddots & \vdots \\ sim\left(w_{q_k} w_{s_1}\right) & \cdots & sim\left(w_{q_k} w_{s_j}\right) \end{bmatrix}, \quad (3)$$

$$\text{sim}\left(w_{q_k}, w_{s_j}\right) = \frac{\vec{w}_{q_k} \vec{w}_{s_j}}{\| w_{q_k} \| \| w_{s_j} \|}. \quad (4)$$

In (2), K is the amount of words in u_q and J is the amount of words in u_s. Equation (3) is the same as (2). In (4), first, we vectorize the word with Word2Vec [36] and then calculate the similarity. w_{q_k} represents the word vector corresponding to Word2Vec:

$$\alpha_{k,j} = \frac{\exp\left(\text{sim}\left(w_{q_k}, w_{s_j}\right)\right)}{\sum_{i=1}^{j} \exp\left(\text{sim}\left(w_{q_k}, w_{s_j}\right)\right)}. \quad (5)$$

We normalize the similarity of each row to calculate the weight $\alpha_{k.j}$:

$$u_k = \sum_{j=1}^{J} \alpha_{k,j} \text{sim}\left(w_{q_k}, w_{s_j}\right). \quad (6)$$

We use the accumulation of the multiplication of weights and similarities as the similarity of each row in u:

$$\text{usim}\left(u_q, u_s\right) = \frac{1}{k} \sum_{i=1}^{k} u_i,$$

$$\text{gsim}\left(g_q, g_s\right) = \frac{1}{k} \sum_{i=1}^{k} u_i. \quad (7)$$

The formula for $\text{asim}\left(a_q, a_s\right)$ is as follows:

$$\text{asim}\left(a_q, a_s\right) = \frac{\sum_{i=1}^{n} \max\left(\text{masim}\left(a_{q_i}, a_{s_{1\sim m}}\right)\right)}{n}. \quad (8)$$

Here n is the amount of verbs in the aim and motivation of u and m is the amount of verbs in the activities of s. $\text{masim}\left(a_{q_i}, a_{s_i}\right)$ represents the similarity between an element in a_q and an element in a_s:

$$\text{masim}\left(a_q, a_s\right) = w_1 \times \text{sim}\left(V_q, V_s\right) + w_2 \times \frac{\sum_{i=1}^{I} \max\left(\sum_{j=1}^{J} \text{sim}\left(N_{q_i}, N_{s_j}\right)\right)}{I}. \quad (9)$$

In (9), V_q is a verb in aim or motivation, V_s is another verb in activity, I and J represent the number of nouns in a_q and a_s, respectively, $N_{q_i}$is the i-th noun in a_q, N_{s_j} is the j-th noun in a_s, and w_1 and w_2 are the weights of verbs and nouns, respectively. Using (4), $\text{sim}\left(V_q, V_s\right)$ and $\text{sim}\left(N_{q_i}, N_{s_j}\right)$ are calculated as similarities between words.

For example, when calculating the similarity between requirements and the open API description text in step 3, $\text{usim}\left(u_q, u_s\right)$ and $\text{gsim}\left(g_q, g_s\right)$ are calculated in the same way. Here, we take $\text{gsim}\left(g_q, g_s\right)$ as an example. At this time, g_q is <null, online><server, null>, and g_s is <Internet, over>. First establish the similarity matrix $\begin{bmatrix} \text{sim (null, Internet)} & \text{sim (null, over)} \\ \text{sim (server, Internet)} & \text{sim (server, over)} \\ \text{sim (online, Internet)} & \text{sim (online, over)} \\ \text{sim (null, Internet)} & \text{sim (null, over)} \end{bmatrix}$ of gsim, where sim(online, Internet) is calculated using Word2Vec similarity. After obtaining the similarity matrix, calculate the $a_{k,j}$ weight for every similarity of each row; the weight indicates the importance of each element in each row in the entire row; then, linearly combine the weight with the corresponding elements, and finally take the similarity of all rows; the average of the values is taken as the final similarity of gsim.

Compared with the calculation of gsim, asim needs to calculate the similarity between verbs and verbs, nouns and nouns, and then perform linear combination of weights. Specifically, first calculate the similarity masim (a_q, a_s) between an element in a_q and an element in a_s; a_q is composed of aim and motivation (io and adj), that is, <{upload, edit}, photo> and <process, photo>; a_s is composed of scenario (io and adj), that is, <edit, {photo, video}>. Because a_q has 3 verbs and a_s has only one, $n = 3$ and $m = 1$. First calculate the similarity between < upload, photo> and <edit, {photo, video}>; the verb upload and verb edit are calculated with Word2Vec, for the similarity of the nouns: photo and photo, video; because < upload, photo > has only one noun in a_q, <edit, {photo, video}> in a_s has two nouns, $I = 1$, $J = 2$; calculate sim(photo, photo) and sim(photo, video) respectively; take the maximum value of the similarity of a noun k in a_q to all nouns in a_s as the similarity of k to a_s nouns, and then take the average value of the similarity of all nouns in a_q to the a_s noun as the noun similarity between a_q and a_s. Finally, the weight of the verb and the similarity of the verb, and the weight of the noun and the similarity of the noun are linearly combined to obtain the final similarity, that is, the final masim(a_q, a_s). Similarly, calculate the similarity of <edit, photo> and <edit, {photo, video}>, <process, photo> and <edit, {photo, video}>. Take the maximum value of the similarity of an element in a_q to each element in a_s as the similarity of the a_q element to a_s, and at this time $m = 1$, so the similarity of each element in a_q to a_s is the maximum. $n = 3$. Finally, calculate the average similarity of all elements in a_q to a_s as the final gsim.

Linearly combine usim, asim, and gsim to get the similarity between the final user story and the open API description text.

3.2. MIM-Based Open API Recommendation Approach. Usually, there are a lot of existing open APIs and Mashups in the API registration platforms. Inspired by Li et al. [31], we make use of these existing open APIs and Mashups as well as their properties as multiple-dimensional information to recommend open APIs to Mashup developers when these developers cannot describe their requests clearly. In Figure 3, the MIM-based open API recommendation approach is generally divided into two steps:

Step 1. MIM construction. The MIM contains an active open API matrix, a target Mashup matrix, a similar open API matrix, a similar Mashup matrix, a co-occurrence matrix, and a popularity matrix. The active open APIs matrix is the open API currently used for scoring prediction, and the target Mashup matrix is the Mashup served by the recommended method. First, to build the similar open API and similar Mashup matrix, we extract three elements (i.e., agent, activity, scenario) of open APIs properties and two elements (i.e., Mashup activity, Mashup scenario) of Mashup properties separately by the dependency syntax relationship, so as to combine these elements, respectively, into a connected network to characterize the relationships. Then, the co-occurrence of open APIs is the external manifestation of open API composition relationships. Finally, the QoS property, i.e., the open API popularity, is measured by the open API category and the historical invoking times.

Step 2. FM model score prediction. Different types of information such as the open API popularity, the connection model, and the co-occurrence are trained by the FM to calculate the interaction between Mashups and open APIs, obtain corresponding prediction scores, and then return the list of TOP-N open APIs with the highest score to Mashup developer.

3.2.1. MIM Matrix Construction (Step 1). As mentioned before, MIM is an integrated matrix including multiple-dimensional information. In MIM, the active open APIs matrix represents the open API currently used for the score prediction. The target Mashup matrix represents the Mashup to be developed. The similar open API matrix presents the similarities between open APIs. The similar Mashup matrix represents the similarity between the Mashup in the Mashups matrix and their similar Mashups. The co-occurrence matrix represents the co-occurrence relationships between the active open APIs and other open APIs. The popularity matrix represents the total frequency of the invoking history of the active open APIs.

(1) The Similar Open APIs Matrix Construction. Open API properties extraction. Open APIs properties are selected as the functional characteristics for open API recommendation, i.e., the agent, activity, and scenario. The extraction process is the same as what we have shown in Section 3.1 and will not be described here.

Similarity calculation: the calculation of similarity is the same as what we have shown in Section 3.1.3 and will not be described here. A similar open APIs matrix can be obtained by multiple calculations of the above similarity approach.

(2) The Similar Mashup Matrix Construction. Mashup properties extraction. Two Mashup properties are extracted from the Mashup description text and the number of identical open APIs invoked between Mashups.

(1) Mashup description text extraction:

The Mashup description text recorded the detailed information of the Mashup and a storage carrier, which is written in NL and in any format, for example, "customers can use the service to upload and edit photos over the Internet." We randomly extract 10,000 Mashup application descriptions from the application library and extract SDs on the application descriptions. Compared with open API, Mashup lacks user description information. Therefore, we only extract the activity (named as Mashup activity) and scenario from the Mashup description text:

(1) Mashup activity (ma): the activity provided by the Mashup, which in the example refers to "upload and edit photos."

(2) Mashup scenario (ms): the scenario of the Mashup, which in the example refers to "over the Internet."

(2) The number of identical open APIs invoked between Mashups

Each Mashup to be developed is composed of two or more open APIs, and different Mashups may invoke the same open API. Therefore, the invoked open APIs can reflect the similarity between two Mashups.

Definition 3 (Mashup properties). mp = <ma, ms>, where ma represents open API key activity information consisting of verbs and nouns. ms represents open API key scenario information consisting of verbs and nouns

Definition 4 (Mashup activity). ma = <verb, object>; the verb is a user-initiated operation. Object exists in the form of a noun, a noun phrase, or a noun phrase in a binary group, such as the open API key activity <{upload, edit}, photo>.

Definition 5 (Mashup scenario). ms = <object>; the verb exists in the form of a verb in a binary group and is a user-initiated operation, <Internet>.

We have identified 10 SDs and classified the SDs into the following six scenarios to extract the Mashup properties, as shown in Table 2.

Similarity calculation: suppose there are two Mashup descriptions "customers can use the service to upload and edit photos over the Internet" and "this Mashup allows users to watch pictures on the server or Internet." We calculate the similarity between Mashups through Mashup description text and the number of identical open APIs invoked between Mashups.

The specific formula for calculating the similarity of two Mashups is shown in

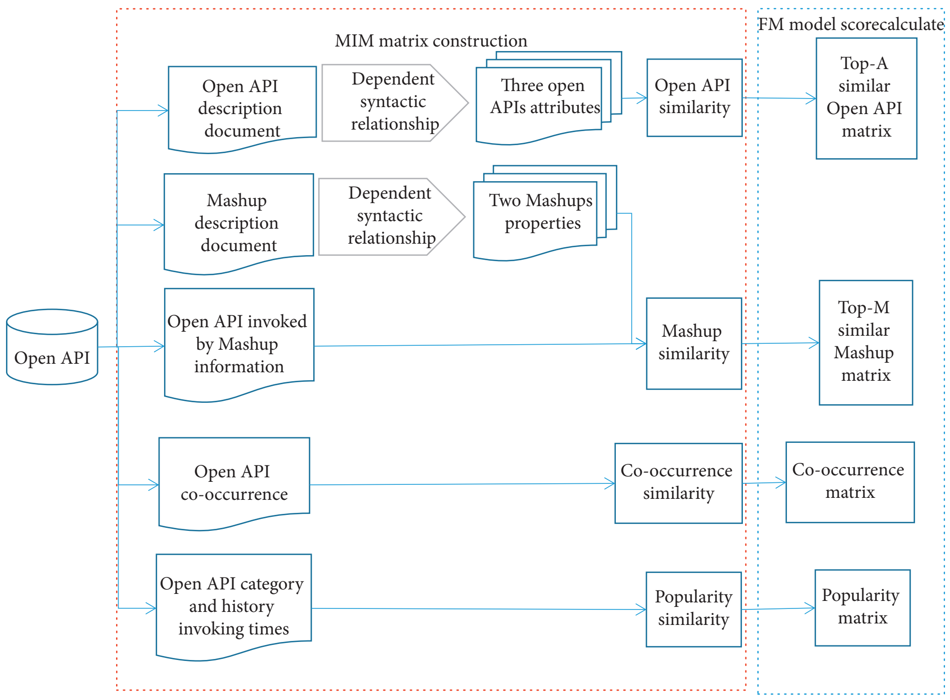

MIM matrix construction
FM model scorecalculate
Open API description document
Dependent syntactic relationship
Three open APIs attributes
Open API similarity
Top-A similar Open API matrix
Mashup description document
Dependent syntactic relationship
Two Mashups properties
Open API
Open API invoked by Mashup information
Mashup similarity
Top-M similar Mashup matrix
Open API co-occurrence
Co-occurrence similarity
Co-occurrence matrix
Open API category and history invoking times
Popularity similarity
Popularity matrix

Figure 3: MIM-based open API recommendation approach.

Table 2: Related SDs when extracting Mashup properties.

Scenarios	Related SDs
x, y are placed in the ma verb list	conj(x, y), csubj(x, y), csubjpass(x, y)
x is placed in the ma verb list	
y is placed in the ma verb list	dobj(x, y), iobj(x, y), prep(x, y) & pobj(y, z) (note: x is a verb)
x is placed in the ma noun list	
z is placed in the mp noun list	prep(x, y) & pobj(y, z) (note: x is a noun)
y is placed in the ma noun list	nn(x, y) (when x is already in the noun list of ma), cop(x, y)
y is placed in the mp noun list	nn(x, y) (when x is already in the noun list of ms), pobj (x, y)
y is placed in the mp adjective list	amod(x, y)

$$\mathrm{sim}(M_1, M_2) = u \times \mathrm{sim}_{\mathrm{ma}}(M_1, M_2) + v \times \mathrm{sim}_{\mathrm{ms}}(M_1, M_2) + w \times \mathrm{sim}_{\mathrm{se}}(M_1, M_2), \tag{10}$$

where the weight parameter $u = 0.2$, $v = 0.6$, $w = 0.2$ $(u + v + w = 1)$. M_1, M_2 represent two different Mashups. $\mathrm{sim}_{\mathrm{ma}}(M_1, M_2)$ represents the similarity of activities between two Mashups (e.g., <{upload, edit}, photo> and <watch, picture>). $\mathrm{sim}_{\mathrm{ms}}(M_1, M_2)$ represents the similarity of the scenarios between two Mashups (e.g., <Internet> and <{server, Internet}>). $\mathrm{sim}_{\mathrm{se}}(M_1, M_2)$ represents the similarity of open APIs invoked between Mashups. The formula of $\mathrm{sim}_{\mathrm{ma}}$ (M_1, M_2) is as follows:

$$\mathrm{sim}_{\mathrm{ma}}(M_1, M_2) = \frac{\sum_{i=1}^{n} \max\left(\mathrm{msim}_{\mathrm{ma}}\left(M_{1_i}, M_{2_{1\sim m}}\right)\right)}{n}, \tag{11}$$

where n is the amount of verbs in the ma of M_1, and m is the amount of verbs in the ma of M_2. masim (a_{q_i}, a_{s_i}) represents the similarity between an element in ma of M_1 and an element in ma of M_2:

$$\mathrm{msim}_{\mathrm{ma}}(M_1, M_2) = w_1 \times \mathrm{sim}(V_1, V_2) + w_2 \times \frac{\sum_{i=1}^{I} \max\left(\sum_{j=1}^{J} sim\left(N_{1_i}, N_{2_j}\right)\right)}{I}. \tag{12}$$

In (11), V_1 is a verb in the ma of M_1, V_2 is another verb in the ma of M_2, I and J represent the number of nouns in the ma of M_1 and ma of M_2, respectively, N_{1_i} is the i-th noun in the ma of M_1, N_{2_j} is the j-th noun in the ma of M_2, and w_1 and w_2 are the weights of verbs and nouns, respectively. Using (4), $\text{sim}(V_1, V_2)$ and $\text{sim}(N_{1_i}, N_{2_j})$ are calculated as similarities between words.

The formula of Sim_{ms} (M_1, M_2) is as follows:

$$\text{sim}_{\text{ms}}(M_1, M_2) = \frac{1}{k}\sum_{i=1}^{k} u_i,$$
$$u_i = \begin{bmatrix} \max[\text{sim}(M_{11}, M_{21})] & \dots & \max[\text{sim}(M_{11}, M_{2j})] \\ \vdots & \ddots & \vdots \\ \max[\text{sim}(M_{1k}, M_{21})] & \dots & \max[\text{sim}(M_{1k}, M_{2j})] \end{bmatrix}, \tag{13}$$

where u_i represents a set of word combination's similarities. k, j respectively represent the amount of words contained in the ma or ms of the two Mashups M_1 and M_2. $M_{11}, \ldots, M_{1k}$ represent each word corresponding to ma or ms of M_1. $M_{21}, \ldots, M_{2j}$ represent each word corresponding to ma or ms of M_2. The two groups of words are compared one-to-one to calculate the similarity, and a similarity matrix of $k \times j$ is formed. Take the maximum value of the similarity result of each row to represent the similarity of the corresponding words, and finally a matrix of $1 \times k$ is obtained.

We adopted the Jaccard similarity calculation idea to calculate the similarity of open APIs invoked between Mashups:

$$\text{sim}_{\text{se}}(M_{1i}, M_{2j}) = \frac{|A_i \cap A_j|}{|A_i \cup A_j|}. \tag{14}$$

Here, $|A_i \cap A_j|$ represents the amount (the intersection) of the same open API in the corresponding open API composition of the two Mashups, and $|A_i \cup A_j|$ represents the total amount of open APIs (the union) in the open API composition corresponding to the two Mashups.

For example, sim(M_1, M_2) is composed of three parts: $\text{sim}_{\text{ma}}(M_1, M_2)$, $\text{sim}_{\text{ms}}(M_1, M_2)$, and $\text{sim}_{\text{se}}(M_1, M_2)$. The calculation formula of $\text{sim}_{\text{ma}}(M_1, M_2)$ and $\text{sim}_{\text{ms}}(M_1, M_2)$ is similar to that in Section 3.1.3, which is described in detail here. $\text{sim}_{\text{ma}}(M_1, M_2)$ is the similarity between <{upload, edit}, photo> and <watch, picture>, and $\text{sim}_{\text{ms}}(M_1, M_2)$ is the similarity between <Internet> and <{server, Internet}>. In $\text{sim}_{\text{se}}(M_1, M_2)$, suppose M_1 has invoked a total of 10 open APIs, M_2 has invoked a total of 15 open APIs; both M_1 and M_2 invoke the same 5 APIs, and the final result of $\text{sim}_{\text{se}}(M_1, M_2)$ is $5/(10 + 15) = 0.2$.

Next we linearly combine $\text{sim}_{\text{ma}}(M_1, M_2)$, $\text{sim}_{\text{ms}}(M_1, M_2)$, and $\text{sim}_{\text{se}}(M_1, M_2)$ to get the similarity between M_1 and M_2.

(3) The Co-Occurrence Matrix Construction. Co-occurrence refers to the external connection between open APIs formed in Mashup development. If two different open APIs S_1 and S_2 are directly invoked by the same Mashup, there is co-occurrence between S_1 and S_2. The following formula calculates the co-occurrence between two different open APIs:

$$co(a_i, a_j) = \frac{|a_i \cap a_j|}{|a_i \cup a_j|}. \tag{15}$$

Here, ai and aj represent different open APIs, respectively. $|a_i \cap a_j|$ represents the total times ai and aj were invoked by the same Mashup. $|a_i \cup a_j|$ represents the sum of the times the open API ai and aj were invoked in the past.

(4) The Popularity Matrix Construction. Quality of open API (QoS) is the basis for ensuring the performance of the open API. However, the QoS information is usually vague and dynamic. Therefore, in this work, we have integrated the open API popularity with QoS information to improve the recommendation effect. The open API popularity is calculated by the amount of historical invokes. Therefore, the more popular the domain is, the more invoked the open API in this domain will be. We use (16) to calculate the open API popularity:

$$\text{pop}(ai) = \frac{\text{Fre}(ai) - \min(\text{Fre}(\text{Category}(ai))}{\max(\text{Fre}(\text{Category}(ai)) - \min(\text{Fre}(\text{Category}(ai))}, \tag{16}$$

where Fre(.) calculates the amount of times the corresponding open API has been invoked by the Mashup, Category(.) represents all open APIs that exist in the same domain, max(.) calculates the maximum amount of times the open API has been invoked in history, and min(.) calculate the minimum amount of times the open API has been invoked in history.

3.2.2. FM Model Score Prediction (Step 2). FM can learn the characteristic interaction between Mashup and open API, so as to calculate the correlation information between them. The specific formula of FM is as follows:

$$Y(X) := w_0 + \sum_{i=1}^{n} w_i x_i + \frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{n} v_{i,f} x_i\right)^2 - \sum_{j=i+1}^{n} v_{j,f}^2 x_j^2\right), \tag{17}$$

where n is the amount of the features, w_0 is the initial bias term, w_i is the weight of the i-th feature, x_i, x_j represent the interaction between the paired feature variables, $v_{i,f}$, $v_{j,f}$ represent a hidden factor between Mashup x_i and open API x_j in the factorization model, and k is the factorization matrix dimension.

Figure 4 shows an example of an FM model's input and output. The training data consists of two parts. In this work, we use the MIM matrix as input data and a score value Y as output data. Finally, FM can calculate the target value Y between Mashup and the open API and offer recommendations for the Mashup developers by sorting Y.

In the training set, if the active open API is historically invoked by the Mashup, the value in the vector Y is 1; otherwise it is 0. In the test set, the value in vector Y represents the calculated score of the active open API relative to the Mashup. Finally, the final set of recommended open APIs is obtained by ranking the predicted scores.

In the FM model, the model parameters w_0, w, and v are obtained from the training examples. In order to get the optimal parameters, a loss function needs to be defined to obtain the optimal parameter model. We define the loss function as

$$l(Y(x_i), Y') = \log(1 + \exp(-Y(x_i)Y')). \qquad (18)$$

For example, suppose that there are a total of 3 open APIs and 2 Mashups in the entire dataset. The score between the second open API and the first Mashup is calculated, so the active open API is a vector [0, 1, 0] and the target Mashup is a vector [0, 1], the similar open APIs matrix calculated by the "Similar Open APIs Matrix Construction" section is a vector [0.4, 0, 0.7], the similar Mashup matrix calculated by the "Similar Mashup Matrix Construction" section is a vector [0, 0.3], the co-occurrence matrix calculated by (15) is a vector [0.5, 0], and the popularity matrix calculated by formula (16) is a vector [3]. The final MIM is [0, 1, 0, 1, 0, 0.4, 0, 0.7, 0, 0.3, 0.5, 0, 3] and will be entered into the trained FM to get the final score.

4. Experimental Results and Analysis

We conducted a series of experiments on user-story-driven open API discovery and MIM-based open API recommendation approach to evaluate the effectiveness of the HyOASAM approach [37].

4.1. Experimental Data Collection and Analysis. We crawled all open APIs, Mashups, and information about the relationships between open APIs and Mashups on PWeb. We have collected 15,928 open API items with 398 categories, 6,973 Mashups, and 13,613 links between open APIs and Mashups.

After crawling the data, we performed a series of data processing operations on the data, including filtering special characters and removing open API description texts that are not related to open APIs. After text preprocessing, we selected 1,438 open APIs.

In order to evaluate the effectiveness of HyOASAM, we have established a standard set. We appointed four graduate students with extensive experience in Mashup development to build the establishment of the standard set. The four students built four different sets of open API list standards based on the practical experience they developed. In the end, we used precision as criteria. Due to different standard sets, the final results are also different, and we take the average of the four results as the final standard set of the experiment.

4.2. The Experimental Analysis for Open API Discovery. Three requirements components are extracted from each user story. Table 3 shows the result; we select the five different user stories domains in the table as the experimental open API requirement texts.

4.2.1. Metric Selection. We use precision to evaluate the effectiveness of user-story-driven open API discovery. The precision formula is as follows:

$$\text{precision} = \frac{|S_A \cap S_M|}{|S_A|}. \qquad (19)$$

Here, S_A represents the requirements components extracted by HyOASAM and S_M represents the manually extracted open API properties.

4.2.2. Parameter Selection. In (1), a represents the weight of users, b represents the weight of functions, and c represents the weight of motivation. We compared the open API sets from three different domains user stories as input to the three standard open API sets. In Figure 5, it can be seen that the parameters of $a = 0.2$, $b = 0.6$, and $c = 0.2$ are in most cases better than the precision of other parameters. This shows that function is the main factor of the overall similarity.

4.2.3. Comparative Experiment. We compared the user-story-driven open API discovery approach (USDOAD) with other established open API discovery approaches [38, 39]. The two established approaches are as below:

(1) Open API discovery approach based on vector space model (VSMOAD): we used VSM to vectorize the data processed user story $u = \{u_1, u_2, u_3, u_4, \ldots, u_i\}$ and open API description text $s = \{s_1, s_2, s_3, \ldots, s_i\}$, where i is the size of the corpus vocabulary, and then used cosine similarity to calculate similarity:

$$\cos(\vec{q}, \vec{s}) = \frac{\vec{q} \cdot \vec{s}}{\|\vec{q}\|\|\vec{s}\|} = \frac{\sum_{i=1}^{v} \vec{q} \cdot \vec{s}}{\sqrt{\sum_{i=1}^{v} \vec{q}_i^{\,2} \sum_{i=1}^{v} \vec{s}_i^{\,2}}}. \qquad (20)$$

(2) Open API discovery approach based on LDA (LDAOAD): LDA is a topic model, which can give the topic of each document in the document set as a probability distribution, so we used LDA to extract

MIM																					Y
Active open API				Target Mashup				Similar open API				Similar Mashup				Co-occurrence				Popularity	Score
1	0	0	…	1	0	0	…	0	0.6	0.5	…	0	0.3	0.7	…	0	0.5	0.5	…	12	1
0	1	0	…	1	0	0	…	0.4	0	0.7	…	0	0.3	0.7	…	0.5	0	1	…	3	0
0	0	1	…	0	0	1	…	0.5	0.8	0	…	0.3	0.4	0	…	0.5	1	0	…	5	1
1	0	0	…	0	0	1	…	0	0.6	0.3	…	0.3	0.4	0	…	0	0.5	0.5	…	7	0
0	1	0	…	0	1	0	…	0.6	0	0.7	…	0.5	0	0.6	…	0.5	0	1	…	1	1
0	0	1	…	0	1	0	…	0.5	0.7	0	…	0.5	0	0.6	…	0.5	1	0	…	8	1

Figure 4: The example of FM model input and output.

Table 3: User story extraction examples. According to Section 3.1, we extract open API properties on PWeb. Table 4 shows specific examples of open API properties.

No.	User story	Open API category	Requirements components
S1	As an editor, I want to download and edit pictures on the website	Photo	<editor, null>, <{download, edit},picture, null, null>, <null, null, website, null>
S2	As a producer, I want to search and upload music online	Music	<producer, null>, <{search, upload},music, null, null>, <null, null, null, online>
S3	As an editor, I want to download and upload videos from the app	Video	<editor, null>, <{download, upload},video, null, null>, <null, null, app, null>
S4	As a fan, I want to find music so that I can listen to music online	Music	<fan, null>, <find, music, null, null>, <listen, music, null, online>
S5	As a traveler, I want to find routes and hotel online	Travel	<traveler, null>, <find, {route, hotel}, null, null >, <null, null, null, online>

Table 4: Examples of open API properties extraction.

Name	Category	Open APIs properties
PicMonkey	Photos	<{maker}, null> <{upload, edit, save, contact}, {image, provider}> <{documentation,talk@picmonkey.com}, null}>
Google maps	Mapping	<map, null> <{utilize, access, embedding}, {language, localization, api, developer, geocoding, service, intranet}> <{service, customer, connection}, null>
Google-AdSense	Advertising	<{developer, blog}, null> <{create, generate, choose, generate, share, sense},{content, report, revenue, program, site}> <{account, snippet, filter}, null>

the subject distribution vector of user story u and open API description text s and then used enhanced cosine similarity to calculate similarity. The formula is as follows:

$$\mathrm{Sim}(a,u)=\frac{\sum_{i\in I}\left(r_{a,i}-\overline{r}_a\right)\left(r_{u,i}-\overline{r}_u\right)}{\sqrt{\sum_{i\in I}\left(r_{a,i}-\overline{r}_a\right)^2}\sqrt{\sum_{i\in I}\left(r_{u,i}-\overline{r}_u\right)^2}}. \quad (21)$$

Figure 6 shows that our approach is significantly better than the VSMOAD and LDAOAD approach, but the precision between TOP-20 and TOP-25 is significantly reduced. Because Stanford Parse cannot fully extract the open API demand components in some scenarios, open API function is represented by (1) noun phrases, such as "video upload," and (2) sentences with grammatical errors or missing structural components.

4.3. The Experimental Analysis for Open API Recommendation

4.3.1. Metric Selection.

We adopt the precision, recall, and F-measure to evaluate the efficiency of the MIM-based open API approach:

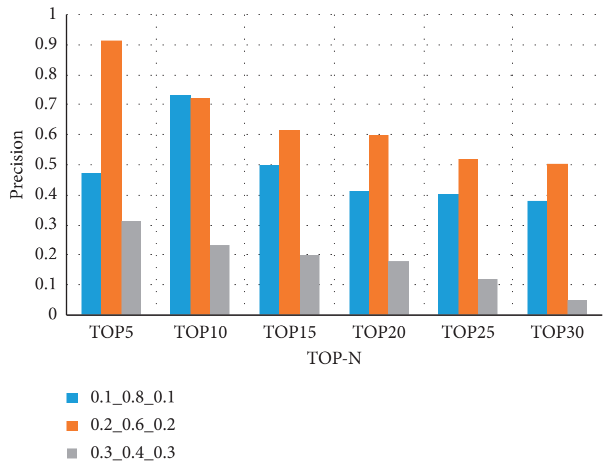


Figure 5: The result of selecting different parameters.

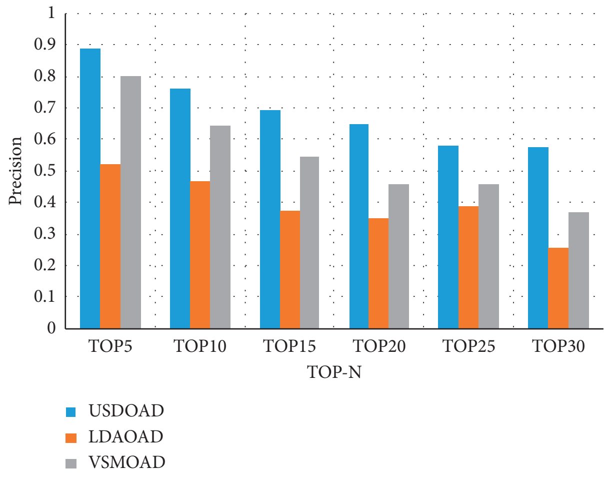


Figure 6: The precision results of comparing USDOAD, LDAOAD, and VSMOAD.

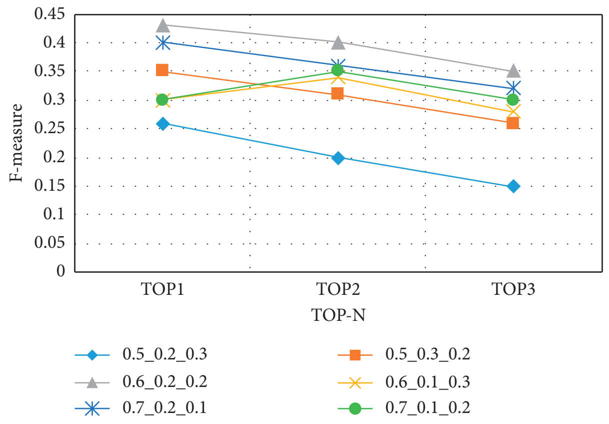


Figure 7: The *F*-measure value corresponding to each parameter.

$$P_{\text{recall}} = \frac{|R(A_i) \cap RM(A_i)|}{RM(A_i)},$$
$$P_{\text{precision}} = \frac{|R(A_i) \cap RM(A_i)|}{R(A_i)}. \quad (22)$$

In the above two formulas, $R(A_i)$ represents the open API actually invoked by the target Mashup and $RM(A_i)$ represents the recommended open API from our approach.

F-measure is the unified average of recall rate and accuracy:

$$F\_\text{measure} = \frac{2 \times P_{\text{recall}} \times P_{\text{precision}}}{P_{\text{recall}} + P_{\text{precision}}}. \quad (23)$$

The relationship between the three metrics and the performance of the recommended algorithm is roughly positively correlated. The larger the recall, precision, and *F*-measure are, the better the performance of the recommended approach is; otherwise it has poor performance.

4.3.2. Parameter Selection. The similarity calculation formula for the open API-recommended algorithm invoked Mashup that is proposed in this paper contains three parameters: u, v, and w, which correspond to the function of Mashup, the application scenario and the similarity calculation of the invoked open API, and $u + v + w = 1$. They directly affect the construction of MIM, which indirectly influences the effect of FM model. Figure 7 shows the effect of the values of the five groups u, v, and w on the recommended results. It can be seen from Figure 7 that when the values of u, v, and w are 0.6, 0.2, and 0.2, the recommended effect is higher than other groups, so our parameters are configured as $u = 0.6$, $v = 0.2$, $w = 0.2$.

We choose the best values of TOP-A open APIs and TOP-M Mashups similar to achieve the best recommended results, as shown in Figures 8 and 9, respectively; when the number of recommended open APIs is 1, 2, and 3, the values of TOP-A and TOP-M (A is from 5 to 30 and M is from 5 to 30) affect the recommendation result. The experimental results show that when TOP-A remains unchanged and TOP-M = 20, F-measure is the best; when TOP-M remains unchanged and TOP-A = 10, the F-measure of MIM-based reaches its peak value. It turns out that selecting the appropriate TOP-A value and TOP-M value for the Mashup creation in the open API recommendation is very important.

4.3.3. Comparative Experiment. We compared MIM-based open API recommendation approach with other three recommendation approaches, which are TF-IDF [40], E-LDA [41], and LDA-FM [31].

(1) TF-IDF: This approach starts from the degree of similarity between the active open API description document and the target Mashup description

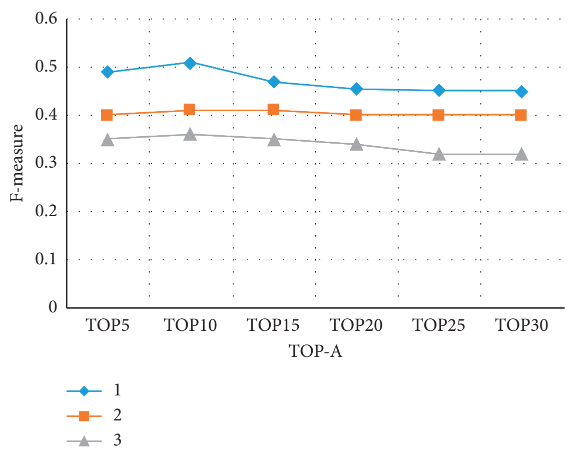


Figure 8: TOP-A's F-measure values.

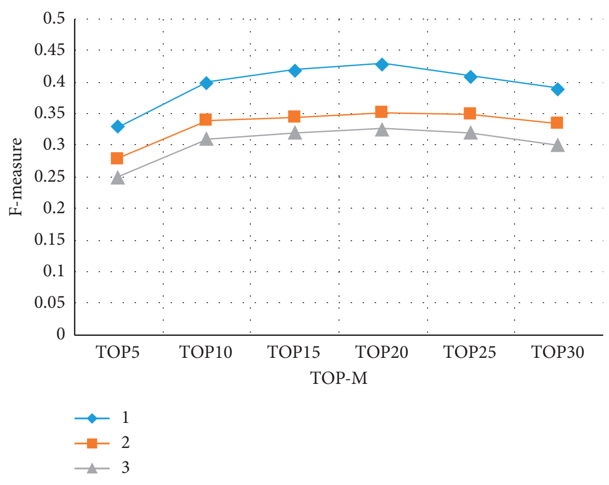


Figure 9: TOP-M's F-measure values.

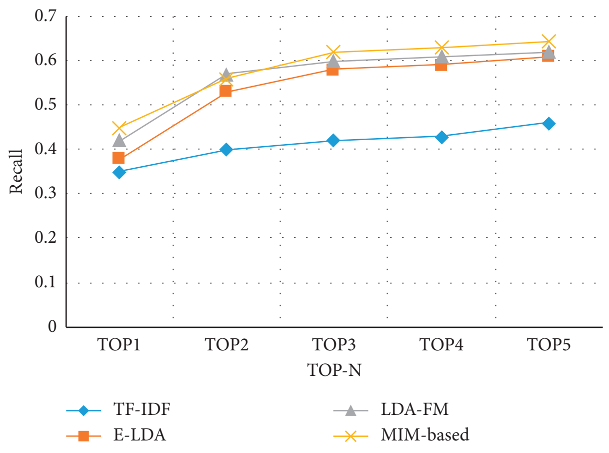


Figure 10: Results of the four approaches on recall.

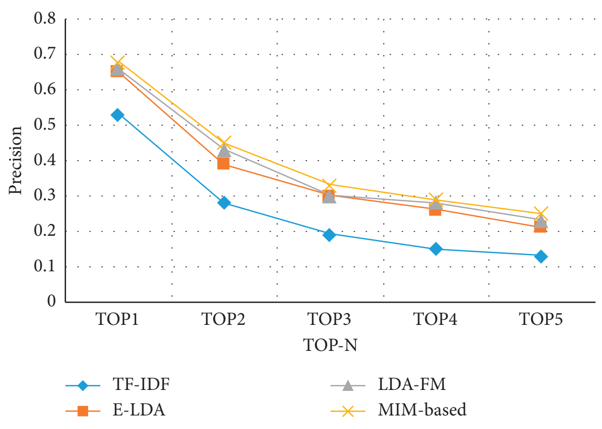


Figure 11: Results of the four approaches on precision.

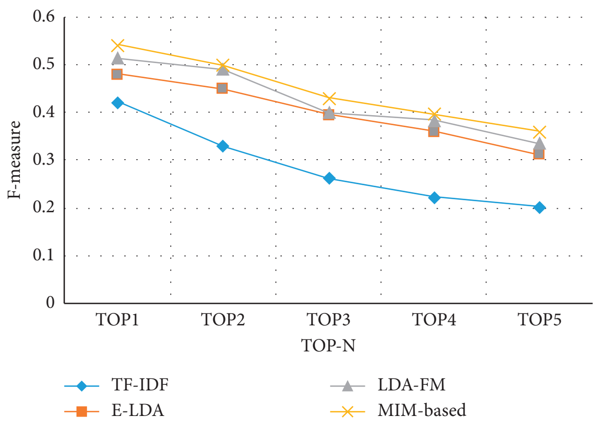


Figure 12: Results of the four approaches on *F*-measure.

document, and the score is calculated in conjunction with the open API popularity. First, we used the TF-IDF to calculate the word vector between the open API and the Mashup. Then, using the similarity of the word vector, we measured the similarity between the active open API and the target Mashup. Finally, we combined the similarity with the open API popularity to obtain the final open API recommendation score.

(2) E-LDA: This approach first calculates the topic vector of target Mashup and open API with LDA, then calculates their similarity, and then integrates the open API popularity for recommendation.

(3) LDA-FM: First, it extracts the topic distribution of the target Mashup and the active open API through the LDA model. The topic information is trained with FM to calculate the probability distribution of the open API. Similarly, we can get the similarity between target Mashup and active open API. In addition, it also takes the co-occurrence and popularity into account.

The evaluation result is calculated in terms of the recall, precision, and F-measure [42]. The comparison shows that our approach has the highest accuracy in all the three metrics.

In Figure 10, our approach is better on recall than the other three approaches, and recall increases as *N* increases. In Figure 11, although the precision decreases as *N* increases, our approach is still the best. As shown in Figure 12, the average *F*-measure value of the MIM-based approach is 2.21% higher than LDA-FM, 4.60% higher than E-LDA, and 15.81% higher than TF. In all cases, TF-IDF has the worst performance. TF-IDF just uses the frequency of word occurrences to vectorize words, regardless of the underlying semantic relevance behind them. MIM-based approach, LDA-FM, and E-LDA reveal the semantic relevance of open APIs and Mashup description documents, so they can calculate their similarities with higher accuracy.

5. Conclusions and Future Work

In order to enable Mashup developers to select the most suitable open API from a large set of open APIs in a rapid agile development mode, we propose a Hybrid Open API Selection Approach for Mashup development (HyOASAM). HyOASAM is composed of two basic approaches: a user-story-driven open API discovery approach and a MIM-based open API recommendation approach. Through HyOASAM, Mashup developers can (1) use user stories to describe open API requirements clearly and quickly, and (2) dynamically get a list of open APIs that match the requirements and select the open APIs they want. It can be seen through experiments that HyOASAM has improved in precision and recall. In the future we will consider employing Word Embedding and Attention Model into NLP techniques, so that the semantic relationship between words can be fully extracted.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors wish to thank all the participants in the evaluation process for their help. This work was supported by the Zhejiang Provincial Natural Science Foundation of China (Grant no. LY19F020003) and the National Natural Science Foundation of China (Grant no. 61672459).

References

[1] W. Pan, B. Song, K. Li, and K. Zhang, "Identifying key classes in object-oriented software using generalizedk-core decomposition," *Future Generation Computer Systems*, vol. 81, pp. 188–202, 2018.

[2] W. Pan, B. Li, J. Liu, Y. Ma, and B. Hu, "Analyzing the structure of Java software systems by weightedK-core decomposition," *Future Generation Computer Systems*, vol. 83, pp. 431–444, 2018.

[3] W. Pan, H. Ming, C. Chang, Z. Yang, and D. K. Kim, "ElementRank: ranking java software classes and packages using a multilayer complex network-based approach," *IEEE Transactions on Software Engineering*, 2019.

[4] W. Tan, Y. Fan, A. Ghoneim, M. A. Hossain, and S. Dustdar, "From the service-oriented architecture to the web API economy," *IEEE Internet Computing*, vol. 20, no. 4, pp. 64–68, 2016.

[5] Z. Li, H. Zhang, and L. O'Brien, "Facing service-oriented system engineering challenges: an organizational perspective," in *Proceedings of the 2010 IEEE International Conference on IEEE Service-Oriented Computing and Applications (SOCA)*, pp. 1–4, Perth, Australia, December 2010.

[6] X. Liu, Y. Hui, W. Sun, and H. Liang, "Towards service composition based on mashup," in *Proceedings of 2007 IEEE Congress on Services (Services' 07)*, pp. 332�339, IEEE, Salt Lake City, UT, USA, Salt Lake City, UT, USA, July 2007.

[7] S. Aghaee and C. Pautasso, "Mashup development with HTML5," in *Proceedings of the 3rd and 4th International Workshop on open APIs and services Mashups*, p. 10, ACM, Ayia Napa, Cyprus, December2010.

[8] P. Massimo and T. Erl, *SOA Design Patterns (Paperback)*, Pearson Education, London, UK, 2008.

[9] F. Chen, M. Li, H. Wu, and L. Xie, "Web service discovery among large service pools utilising semantic similarity and clustering," *Enterprise Information Systems*, vol. 11, no. 3, pp. 452–469, 2017.

[10] M. Aznag, M. Quafafou, and Z. Jarir, "Leveraging formal concept analysis with topic correlation for service clustering and discovery," in *Proceedings of 2014 IEEE International Conference on Services (ICWS'14)*, pp. 153–160, IEEE, Anchorage, AK, USA, Anchorage, AK, USA, June-July 2014.

[11] M. Aznag, M. Quafafou, E. M. Rochd, and Z. Jarir, "Probabilistic topic models for web services clustering and discovery," *Service-Oriented and Cloud Computing*, pp. 19–33, Springer, Berlin, Heidelberg, 2013.

[12] L. Yao, Q. Z. Sheng, A. H. H. Ngu, J. Yu, and A. Segev, "Unified collaborative and content-based web service recommendation," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 453–466, 2015.

[13] L. Yao, Q. Z. Sheng, A. Segev, and J. Yu, "Recommending web services via combining collaborative filtering with content-based features," in *Proceedings of 2013 IEEE 20th International Conference on Services (ICWS'13)*, pp. 42–49, IEEE, Santa Clara, CA, USA, June-July 2013.

[14] J. Li, J. Wang, Q. Sun, and A. Zhou, "Temporal influences-aware collaborative filtering for QoS-based service recommendation," in *Proceedings of 2017 IEEE International Conference on Services Computing (SCC'17)*, pp. 471–474, IEEE, Honolulu, HI, USA, June 2017.

[15] T. Liang, L. Chen, J. Wu, H. Dong, and A. Bouguettaya, "Meta-path based service recommendation in heterogeneous information networks," in *Service-Oriented Computing*, pp. 371–386, Springer, Cham, Switzerland, 2016.

[16] Y. Wang, T. Wang, and J. Sun, "PASER: a pattern-based approach to service requirements analysis," *International Journal of Software Engineering and Knowledge Engineering*, vol. 29, no. 4, pp. 547–576, 2019.

[17] Y. Wang and L. Zhao, "Eliciting user requirements for e-collaboration systems: a proposal for a multi-perspective modeling approach," *Requirements Engineering*, vol. 24, no. 2, pp. 205–229, 2019.

[18] M. Crasso, A. Zunino, and M. Campo, "A survey of approaches to web service discovery in service-oriented architectures," *Journal of Database Management*, vol. 22, no. 1, pp. 102–132, 2011.

[19] C. Mateos, J. M. Rodriguez, and A. Zunino, “A tool to improve code-first Web services discoverability through text mining techniques,” *Software: Practice and Experience*, vol. 45, no. 7, pp. 925–948, 2015.

[20] A. V. Paliwal, B. Shafiq, J. Vaidya, H. Hui Xiong, and N. Adam, “Semantics-based automated service discovery,” *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 260–275, 2012.

[21] K. Elgazzar, A. E. Hassan, and P. Martin, “Clustering wsdl documents to bootstrap the discovery of web services,” in *Proceedings of 2010 IEEE International Conference on Services (ICWS'10)*, pp. 147–154, IEEE, Miami, FL, USA, July 2010.

[22] C.-B. Ke, Z.-Q. Huang, L.-Y. Liu, and Z.-N. Cao, “Research on constraint-oriented web service discovery,” *Journal of Software*, vol. 23, no. 10, pp. 2665–2678, 2012.

[23] H. Gao, S. Wang, L. Sun, and F. Nian, “Hierarchical clustering based web service discovery,” in *Service Science and Knowledge Innovation*, pp. 281–291, Springer, Berlin, Germay, 2014.

[24] M. Klusch, *Semantic Service Coordination CASCOM: Intelligent Service Coordination in the Semantic Web*, Springer, Berlin, Germay, 2008.

[25] D. Wei, T. Wang, J. Wang, and A. Bernstein, “SAWSDL-iMatcher: a customizable and effective Semantic web service matchmaker,” *Journal of Web Semantics*, vol. 9, no. 4, pp. 402–417, 2011.

[26] B. Cao, J. Liu, Y. Wen, H. Li, Q. Xiao, and J. Chen, “QoS-aware service recommendation based on relational topic model and factorization machines for IoT Mashup applications,” *Journal of Parallel and Distributed Computing*, vol. 132, pp. 177–189, 2019.

[27] Z. Zheng, H. Ma, M. R. Lyu, and L. King, “QoS-aware web service recommendation by collaborative filtering,” *IEEE Transactions on Services computing*, vol. 4, no. 2, pp. 140–152, 2010.

[28] G. Huang, Y. Ma, X. Liu, Y. Lou, and X. Lu M. B. Blake, “Model-based automated navigation and composition of complex service Mashups,” *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 494–506, 2014.

[29] W. Xu, J. Cao, L. Hu, J. Wang, and M. Li, “A social-aware service recommendation approach for Mashup creation,” in *Proceedings of the 2013 IEEE 20th International Conference on Web Services (ICWS'13)*, pp. 107–114, IEEE, Santa Clara, CA, USA, June-July 2013.

[30] W. Gao, L. Chen, J. Wu, and H. Gao, “Manifold-learning based api recommendation for Mashup creation,” in *Proceedings of the 2015 IEEE International Conference on web Services (ICWS'15)*, IEEE, New York, NY, USA, pp. 432–439, June-July 2015.

[31] H. Li, J. Liu, B. Cao, and M. Shi, “Topic-adaptive open API recommendation method via integrating multidimensional information,” *Journal of Software*, vol. 11, p. 10, 2018.

[32] B. Xia, Y. Fan, W. Tan, K. Huang, J. Zhang, and C. Wu, “Category-aware API clustering and distributed recommendation for automatic Mashup creation,” *IEEE Transactions on Services Computing (TSC'14)*, vol. 8, no. 5, pp. 674–687, 2014.

[33] Y. Meng, A. Rumshisky, and A. Romanov, “Temporal information extraction for question answering using syntactic dependencies in an lstm-based architecture,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, (EMNLP'2017)*, pp. 9–11, Copenhagen, Denmark, September 2017.

[34] Z. Li, J. Wang, and N. Zhang, “A topic-oriented clustering approach for domain service,” *Journal of ComPuter Resraech and Develpment*, vol. 51, no. 2, pp. 408–419, 2014.

[35] M. C. D. Marneffe and C. D. Manning, “The Stanford typed dependencies representation,” in *Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation, Coling 2008*, Association for Computational Linguistics, Manchester, UK, pp. 1–8, August 2008.

[36] I. Lizarralde, J. M. Rodriguez, C. Mateos, and A. Zunino, “Word embeddings for improving REST services discoverability,” in *Proccedings of the 2017 XLIII Latin American Computer Conference (CLEI)*, pp. 1–8, IEEE, Cordoba, Argentina, September 2017.

[37] W. Pan and C. Chai, “Measuring software stability based on complex networks in software,” *Cluster Computing*, vol. 22, no. s2, pp. 2589–2598, 2019.

[38] C. Platzer and S. Dustdar, “A vector space search engine for web services,” in *Proccedings of the Third European Conference on Services (ECOWS'05)*, p. 9, IEEE, Vaxjo, Sweden, November 2005.

[39] N. Zhang, J. Wang, K. He, and Z. Li, “An approach of service discovery based on service goal clustering,” in *Proccedings of the 2016 IEEE International Conference on Services Computing (SCC)*, IEEE, San Francisco, CA, USA, pp. 114–121, June-July 2016.

[40] B. Cao, J. Liu, M. Tang, Z. Zheng, and G. Wang, “Mashup service recommendation based on usage history and service network,” *International Journal of Web Services Research*, vol. 10, no. 4, pp. 82–101, 2013.

[41] C. Li, R. Zhang, J. Huai, and H. Sun, “A novel approach for API recommendation in Mashup development,” in *Proccedings of the 2014 IEEE International Conference on Web services(ICWS)*, IEEE, Anchorage, AK, USA, pp. 289–296, June-July 2014.

[42] W. Pan, H. Jiang, H. Ming, C. Chai, B. Chen, and H. Li, “Characterizing software stability via change propagation simulation,” *Complexity*, vol. 2019, Article ID 9414162, 17 pages, 2019.

Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 8278743, 17 pages
https://doi.org/10.1155/2020/8278743

Research Article

An Edge-Based Smoothed Finite Element for Free Vibration Analysis of Functionally Graded Porous (FGP) Plates on Elastic Foundation Taking into Mass (EFTIM)

Trung Thanh Tran,[1] Quoc-Hoa Pham ,[2,3] and Trung Nguyen-Thoi[2,3]

[1]*Department of Mechanics, Le Quy Don Technical University, Hanoi, Vietnam*
[2]*Division of Computational Mathematics and Engineering, Institute for Computational Science, Ton Duc Thang University, Ho Chi Minh City, Vietnam*
[3]*Faculty of Civil Engineering, Ton Duc Thang University, Ho Chi Minh City, Vietnam*

Correspondence should be addressed to Quoc-Hoa Pham; phamquochoa@tdtu.edu.vn

Received 31 December 2019; Revised 10 March 2020; Accepted 17 March 2020; Published 25 April 2020

Academic Editor: Xesús Nogueira

In this paper, free vibration analysis of the functionally graded porous (FGP) plates on the elastic foundation taking into mass (EFTIM) is presented. The fundamental equations of the FGP plate are derived using Hamilton's principle. The mixed interpolation of the tensorial components (MITC) approach and the edge-based smoothed finite element method (ES-FEM) is employed to avoid the shear locking as well as to improve the accuracy for the triangular element. The EFTIM is a foundation model based on the two-parameter Winkler–Pasternak model but added a mass parameter of foundation. Materials of the plate are FGP with a power-law distribution and maximum porosity distributions in the forms of cosine functions. Some numerical examples are examined to demonstrate the accuracy and reliability of the proposed method in comparison with those available in the literature.

1. Introduction

The plate resting on the elastic foundation (EF) is one of the most common types of structures which have practical applications in civil and industrial constructions, especially in transportation and irrigation. In particular, the structures of beams and plates on the EF are subjected to moving loads of means of transport such as roadbeds affected by vehicles, railway tracks, and aircraft runways. In most publications, when investigating the mechanical behavior of structures on the EF, the researchers mainly use the one-parameter Winkler foundation model [1] or two-parameter Winkler–Pasternak foundation model [2, 3]. The analysis of plates resting on the Winkler–Pasternak foundation was previously addressed by several authors. For instance, Fazzolari [3] used an analytical method to consider free vibration and buckling of porous FG Sandwich beams resting on the EF with the Winkler–Pasternak foundation model. Leissa [4] presented results for the free vibration of rectangular plates. Xiang et al. [5] studied free vibration for Mindlin plates on the Winkler–Pasternak foundation using an analytical method. Omurtag et al. [6] used the finite element method (FEM) for the free vibration analysis of the Kirchhoff plates resting on the EF. Özçelikörs et al. [7] analyzed the exact solutions of bending, buckling, and vibration problems of a Levy-plate on the two-parameter foundation. Matsunaga [8] used a special higher-order plate theory (HSDT) to analyze vibration and buckling of thick plates on the EF. Ayvaz et al. [9] developed a modified Vlasov model to consider the earthquake response of thin plates on the EF. Shen et al. [10] based on the Rayleigh–Ritz method to study free and forced vibration of the Reissner–Mindlin plates resting on the EF. Liew et al. [11, 12] and Han and Liew [13] analyzed the free vibration of rectangular plates resting on the EF using a differential quadrature method. Zhou et al. [14] considered the vibration of rectangular plates on the EF using the Ritz method. Chucheepsakul and Chinnaboon [15] investigated plates by a two-parameter foundation model using a boundary element method. Civalek and Acar [16] investigated the bending of Mindlin plates on the EF by developing a singular convolution method. Ferreira et al. [17] presented bending and free vibration analyses of the FGP plates on the Winkler–Pasternak foundation by using radial basis functions. Shahsavari et al. [18] used a new quasi-3D hyperbolic theory for the free vibration analysis of the FGP plates resting on the EF. Zenkour and Radwan [19] proposed an exact analytical approach for free vibration analysis of laminated composite and Sandwich plates resting on the EF using a four-unknown plate theory. Duc et al. [20] presented the analysis of nonlinear thermal dynamic response of shear deformable functionally graded plates on the EF. Mahmoudi et al. [21] developed a refined quasi-three-dimensional shear deformation theory to analyze the functionally graded Sandwich plates resting on the two-parameter EF under thermomechanical loading. Duc et al. [22] used an analytical method to calculate static bending and free vibration of FG carbon nanotube-reinforced composite plate resting on Winkler–Pasternak foundations. Thang et al. [23] considered the effects of variable thickness on buckling and postbuckling behavior of imperfect sigmoid FGM plates on elastic medium subjected to compressive loading. Banh-Thien et al. [24] developed the isogeometric analysis for buckling analysis of nonuniform thickness nanoplates in an elastic medium.

In recent years, the FGP materials have attracted great interest from many researchers over the world due to their lightness and high strength. As a result, they are widely applied for civil engineering, aerospace structures, nuclear plants, and other applications. Kim et al. [25] investigated bending, vibration, and buckling of the FGP microplates using a modified couples stress based on the analytical method. Coskun et al. [26] presented analytical solutions to analyze the bending, vibration, and buckling of the FG microplates based on the third-order shear deformation theory (TSDT). Chen et al. [27] investigated the static bending and buckling of the FGP beams by using the Timoshenko beam theory. Rezaei and Saidi [28, 29] studied the vibration of rectangular and porous-cellular plates based on an analytical method. The vibration of the FGP shallow shells using an improve Fourier method was examined by Zhao et al. [30]. Moreover, the dynamics of the FGP doubly-curved panels and shells were also investigated in [31]. Li et al. [32] analyzed the nonlinear vibration and dynamic buckling of the Sandwich FGP plates with graphene platelet reinforcement (GPL) on the EF. For nonlinear problems, Sahmani et al. [33] used the nonlocal method to analyze nonlinear large-amplitude vibrations of the FGP micro-/nanoplates with GPL reinforce. Wu et al. [34] studied the dynamic of the FGP structures by using FEM. Thang et al. [35] investigated the elastic buckling and free vibration of porous cellular plates based on the first-order shear deformation theory (FSDT). Although the FGP materials have many different types, in this paper, the authors only use the distribution of porosity as presented in [25, 26].

In the other front of the development of numerical methods for computational mechanics, Liu et al. [36] have recently proposed an edge-based smoothed FEM (ES-FEM) using triangular elements which show some following excellent properties for the 2D solid mechanics analyses such as (1) the numerical results are often found superconvergent and very accurate; (2) the method is stable and works well for dynamic analysis; and (3) the implementation of the method is straightforward and no penalty parameter is used. The ES-FEM has been developed for n-sided polygonal elements [37], viscoelastoplastic analyses [38], 2D piezoelectric [39], primal-dual shakedown analyses [40], fluid structure interaction [41, 42], and various applications [43–45]. Recently, in an effort to improve the accuracy of the plate and shell structural analyses, the classical MITC3 element [46] incorporated with the ES-FEM [36], has been proposed to give the so-called ES-MITC3 element [47–51]. In the formulation of the ES-MITC3, the system stiffness matrix is employed using strains smoothed over the smoothing domains associated with the edges of the MITC3 elements. The numerical results demonstrated that the ES-MITC3 has the following great properties [47]: (1) the ES-MITC3 can eliminate transverse shear locking even with the ratio of the thickness to the length of the structures reach 10^{-8} and (2) the ES-MITC3 has better accuracy than the existing triangular elements such as MITC3 [46], DSG3 [52], and CS-DSG3 [53] and is a good competitor with the quadrilateral element MITC4 element [54].

The objective of this research now is to further extend the ES-MITC3 method for free vibration analyses of the FGP plates resting on the EFTIM. The governing equations are derived from the FSDT and the Reissner–Mindlin plate theory due to simplicity and computational efficiency. Besides, the EFTIM is modelled based on a two-parameter Winkler–Pasternak foundation model and added in a mass parameter of foundation. The plate is made from the FGP materials with a power-law distribution (k) and maximum porosity distributions (Ω) in the forms of cosine functions. The accuracy and reliability of the present formulation are verified by comparing with those of other available numerical results. Moreover, the effects of some geometric parameters and material properties on the free vibration of the FGP plates are also examined in detail.

2. Functionally Graded Porous Material Plates on Elastic Foundation

Let us consider an FGP plate resting on EFTIM, as shown in Figure 1. The FGP materials with a variation of two constituents and three different distributions of porosity through-thickness are presented as follows [25, 26]:

$$\begin{aligned}&\text{Case 1: } \Lambda(z) = \Omega\cos\left(\frac{\pi z}{h}\right),\\ &\text{Case 2: } \Lambda(z) = \Omega\cos\left[\frac{\pi}{2}\left(\frac{z}{h}+0.5\right)\right],\\ &\text{Case 3: } \Lambda(z) = \Omega\cos\left[\frac{\pi}{2}\left(\frac{z}{h}-0.5\right)\right],\end{aligned} \tag{1}$$

where Ω is the maximum porosity value. A typical material property of the FGP materials can be considered as in the following power-law relations:

$$P(z) = \left[(P_t - P_b)\left(\frac{z}{h}+0.5\right)^k + P_b\right](1-\Lambda(z)), \tag{2}$$

where P_t and P_b are the typical material properties at the top and the bottom surfaces, respectively; and k is the power-law index. The normalized distributions of porosity through the thickness are shown in Figure 2(a). As shown in Figure 2(a), the porosity distribution of Case 1 is symmetric with respect to the midplane of plates. Case 2 and Case 3 are bottom and top surface-enhanced distributions, respectively. Besides, Figures 2(b)–2(d) show the distributions of a normalized typical property associated with three different cases of porosity distributions with parameters: $\Omega = 0.5$, $k = 1, 5, 10$, and $P_t/P_b = 10$.

The EFTIM is built based on the Winkler–Pasternak foundation by adding a mass parameter of foundation:

$$\mathbf{q}_e = k_1 w(x,y,t) - k_2\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)w(x,y,t) + m_f\frac{\partial^2 w(x,y,t)}{\partial t^2}, \tag{3}$$

where w is the displacement of FGP plate; k_1 and k_2 are, respectively, Winkler foundation stiffness and shear layer stiffness of the Pasternak foundation. In order to mention the effectiveness of the foundation mass involved in the oscillation as well as the continuous interaction of the spring with the plate, the parameter β with unit kg^{-1} is added. It characterizes the effective level of the foundation mass involved in vibration, which is determined based on an experimental basis and the ratio of the density of the foundation to the density of plate material which is defined as $\boldsymbol{\mu}_F = \rho_F/\rho$. Thus, the density of mass m_f involved vibration with the foundation is determined $m_f = \beta\mu_F\rho$. From equation (3) we see that, for the static problem, the EFTIM model and Winkler–Pasternak foundation model are the same. But, for the dynamic problems, these two models have differences, and when omitting the influence parameters of the foundation mass, the EFTIM model is equivalent to the Winkler–Pasternak foundation model. In addition, this foundation model also covers the Winkler foundation model when the influence of shear parameters and foundation mass parameters ignored. It was found that the EFTIM model closely resembles the true feature of the foundation, including the Pasternak and Winkler foundation models.

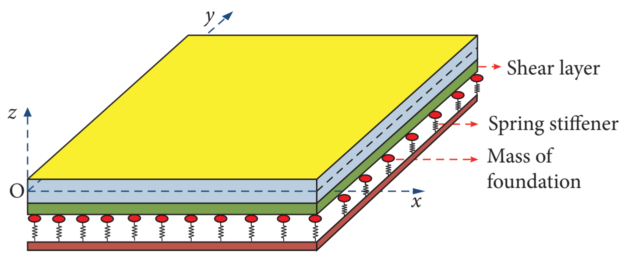


Figure 1: Modeling the FGP plate resting on EFTIM.

3. The First-Order Shear Deformation Theory and Weak Form of the FGP Plates

3.1. First-Order Shear Deformation Theory for FGP Plates. The displacement of the FGP plates in the present work based on the FSDT model can be expressed as

$$\begin{cases} u(x,y,z) = u_0(x,y) + z\theta_x(x,y),\\ v(x,y,z) = v_0(x,y) + z\theta_y(x,y),\\ w(x,y,z) = w_0(x,y),\end{cases} \tag{4}$$

where u, v, w, θ_x, and θ_y are five unknown displacements of the midsurface of the plate. For a bending plate, the strain field can be expressed as follows:

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}_m + z\boldsymbol{\kappa}, \tag{5}$$

where the membrane strain is given as

$$\boldsymbol{\varepsilon}_m = \begin{Bmatrix} u_{0,x}\\ v_{0,y}\\ u_{0,y}+v_{0,x}\end{Bmatrix}. \tag{6}$$

The bending and transverse shear strains are written as

$$\boldsymbol{\kappa} = \begin{Bmatrix} \theta_{x,x}\\ \theta_{y,y}\\ \theta_{x,y}+\theta_{y,x}\end{Bmatrix}, \tag{7}$$

$$\boldsymbol{\gamma} = \begin{Bmatrix} w_{0,x}+\theta_x\\ w_{0,y}+\theta_y\end{Bmatrix}. \tag{8}$$

From Hooke's law, the linear stress-strain relations of the FGP plates can be expressed as

$$\begin{Bmatrix}\sigma_x\\ \sigma_y\\ \tau_{xy}\\ \tau_{xz}\\ \tau_{yz}\end{Bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} & 0 & 0 & 0\\ Q_{21} & Q_{22} & 0 & 0 & 0\\ 0 & 0 & Q_{66} & 0 & 0\\ 0 & 0 & 0 & Q_{55} & 0\\ 0 & 0 & 0 & 0 & Q_{44}\end{bmatrix}\begin{Bmatrix}\varepsilon_x\\ \varepsilon_y\\ \gamma_{xy}\\ \gamma_{xz}\\ \gamma_{yz}\end{Bmatrix}, \tag{9}$$

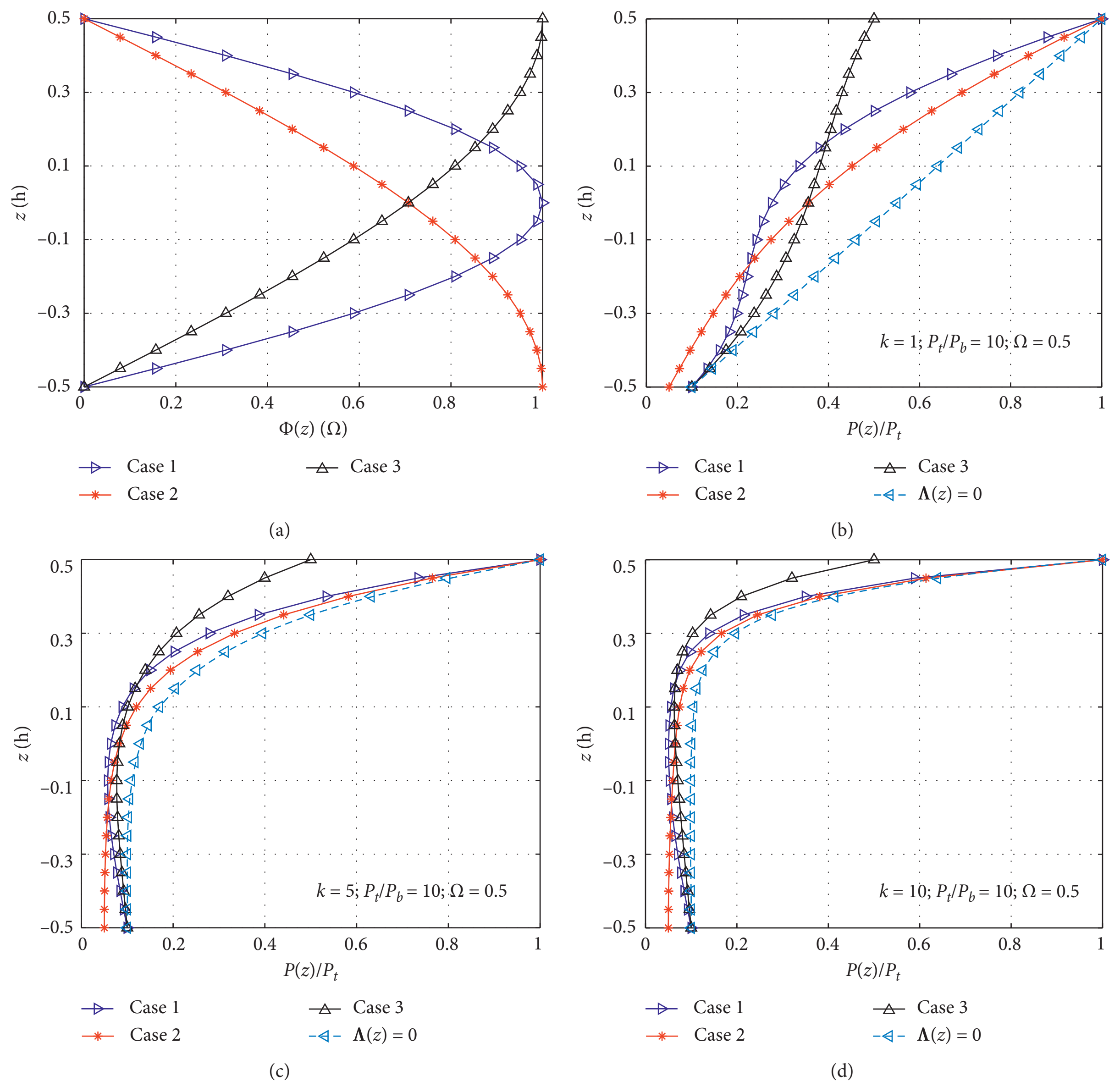


FIGURE 2: Distributions of porosity and typical material property. (a) Distribution of porosity along of z-axis, (b) distribution material property with $k = 1$, (c) distribution material property with $k = 5$, and (d) distribution material property with $k = 10$.

where

$$Q_{11} = Q_{22} = \frac{E(z)}{1 - \upsilon^2},$$
$$Q_{12} = Q_{21} = \frac{\upsilon E(z)}{1 - \upsilon^2}, \tag{10}$$
$$Q_{44} = Q_{55} = Q_{66} = \frac{E(z)}{2(1 + \upsilon)}.$$

where $E(z)$ presents for effective Young's modulus and $\boldsymbol{\upsilon}$ represents Poisson's ratio.

3.2. Weak Form Equations. To obtain the motion equations of the FGP plates for the free vibration analysis, Hamilton's principle is applied with the following form:

$$\int_{t_1}^{t_2} (\delta\mathscr{U} - \delta\mathscr{K})\mathrm{d}t = 0, \tag{11}$$

where $\mathscr{U}$ and $\mathscr{K}$ are the strain and kinetic energies, respectively. The strain energy is expressed as

$$\mathscr{U} = \mathscr{U}^p + \mathscr{U}^f, \tag{12}$$

where $\mathscr{U}^f$ is the strain energy

$$\mathscr{U}^f = \frac{1}{2}\int_{\psi}\left(k_1 w^2 - k_2\left[\left(\frac{\partial^2 w}{\partial x^2}\right)^2 + \left(\frac{\partial^2 w}{\partial y^2}\right)^2\right]\right)\mathrm{d}\psi, \tag{13}$$

and $\mathscr{U}^p$ is the strain energy

$$\mathscr{U}^p = \frac{1}{2}\int_{\psi}\left(\boldsymbol{\varepsilon}^T\mathbf{D}\boldsymbol{\varepsilon} + \boldsymbol{\gamma}^T\mathbf{C}\boldsymbol{\gamma}\right)\mathrm{d}\psi. \tag{14}$$

where $\boldsymbol{\varepsilon} = \begin{bmatrix} \boldsymbol{\varepsilon}_m & \boldsymbol{\kappa} \end{bmatrix}^T$ and

$$\mathbf{D} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B} & \mathbf{F} \end{bmatrix}, \tag{15}$$

and $\mathbf{A}$, $\mathbf{B}$, $\mathbf{F}$, and $\mathbf{C}$ can be given by

$$(\mathbf{A}, \mathbf{B}, \mathbf{F}) = \int_{-h/2}^{h/2} \left(1, z, z^2\right) \begin{bmatrix} Q_{11} & Q_{12} & 0 \\ Q_{21} & Q_{22} & 0 \\ 0 & 0 & Q_{66} \end{bmatrix} \mathrm{d}z, \tag{16}$$

$$\mathbf{C} = \int_{-h/2}^{h/2} \begin{bmatrix} Q_{55} & 0 \\ 0 & Q_{44} \end{bmatrix} \mathrm{d}z. \tag{17}$$

The kinetic energy in equation (11) is given by

$$\mathcal{K} = \mathcal{K}^p + \mathcal{K}^f, \tag{18}$$

where $\mathcal{K}^p$ is the kinetic energy

$$\mathcal{K}^p = \frac{1}{2} \int_\psi \dot{\mathbf{u}} \mathbf{m}_\mathrm{p} \mathbf{u} \, \mathrm{d}\psi, \tag{19}$$

where $\mathbf{u}^T = \begin{bmatrix} u_0 & v_0 & w_0 & \theta_x & \theta_y & \phi_x & \phi_y \end{bmatrix}$ is the displacement field and $\mathbf{m}_\mathrm{p}$ is the mass matrix defined by

$$\mathbf{m}_\mathrm{p} = \begin{bmatrix} I_1 & 0 & 0 & I_2 & 0 \\ & I_1 & 0 & 0 & I_2 \\ & & I_1 & 0 & 0 \\ & & & I_3 & 0 \\ & & & & I_3 \end{bmatrix}, \tag{20}$$

where $(I_1, I_2, I_3) = \int_{-h/2}^{h/2} \rho(1, z, z^2) \mathrm{d}z$.

In equation (11), the kinetic energy of the mass of foundation $\mathcal{K}^f$ is defined as

$$\mathcal{K}^f = \frac{1}{2} \int_\psi \dot{\mathbf{w}} \mathbf{m}_\mathrm{f} \mathbf{w} \, \mathrm{d}\psi. \tag{21}$$

Substituting equations (12) and (18) into equation (11), the weak formulation for the free vibration of the FGP plate is finally obtained as

$$\begin{aligned} &\int_\psi \delta\boldsymbol{\varepsilon}^T \mathbf{D} \boldsymbol{\varepsilon} \, \mathrm{d}\psi + \int_\psi \delta\boldsymbol{\gamma}^T \mathbf{C} \boldsymbol{\gamma} \, \mathrm{d}\psi + \int_\psi \delta\mathbf{w}^T \\ &\quad \cdot \left[k_1 w - k_2 \left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} \right) \right] \mathrm{d}\psi = \int_\psi \dot{\mathbf{u}} \mathbf{m}_\mathrm{p} \mathbf{u} \, \mathrm{d}\psi \\ &\quad + \int_\psi \dot{\mathbf{w}} \mathbf{m}_\mathrm{f} \mathbf{w} \, \mathrm{d}\psi. \end{aligned} \tag{22}$$

4. Formulation of an ES-MITC3 Method for FGP Plates

4.1. Formulation of the Finite Element Using the MITC3 Element. The middle surface of plate ψ is discretized into n^e finite three-node triangular elements with n^n nodes such that $\psi \approx \sum_{e=1}^{n^e} \psi_e$ and $\psi_i \cap \psi_j = \varnothing$, $i \neq j$. Then, the generalized displacements at any point $\mathbf{u}^e = [u_j^e, v_j^e, w_j^e, \theta_{xj}^e, \theta_{yj}^e]^T$ of element ψ_e can be approximated as

$$\begin{aligned} \mathbf{u}^e(\mathbf{x}) &= \sum_{j=1}^{n^{ne}} \begin{bmatrix} N_I(\mathrm{x}) & 0 & 0 & 0 & 0 \\ 0 & N_I(\mathrm{x}) & 0 & 0 & 0 \\ 0 & 0 & N_I(\mathrm{x}) & 0 & 0 \\ 0 & 0 & 0 & N_I(\mathrm{x}) & 0 \\ 0 & 0 & 0 & 0 & N_I(\mathrm{x}) \end{bmatrix} \mathbf{d}_j^e \\ &= \sum_{j=1}^{n^{ne}} \mathbf{N}(\mathbf{x}) \mathbf{d}_j^e, \end{aligned} \tag{23}$$

where n^{ne} is the number of nodes of ψ_e; $\mathbf{N}(\mathbf{x})$ is the shape function matrix; and $\mathbf{d}_j^e = [u_j^e, v_j^e, w_j^e, \theta_{xj}^e, \theta_{yj}^e]^T$ is the nodal degrees of freedom (d.o.f) associated with the jth node of ψ_e.

The membrane bending strains of MITC3 element can be expressed in matrix forms as follows:

$$\boldsymbol{\varepsilon}_m^e = \begin{bmatrix} \mathbf{B}_{m1}^e & \mathbf{B}_{m2}^e & \mathbf{B}_{m3}^e \end{bmatrix} \mathbf{d}^e = \mathbf{B}_m^e \mathbf{d}^e, \tag{24}$$

$$\boldsymbol{\kappa}^e = \begin{bmatrix} \mathbf{B}_{b1}^e & \mathbf{B}_{b2}^e & \mathbf{B}_{b3}^e \end{bmatrix} \mathbf{d}^e = \mathbf{B}_b^e \mathbf{d}^e, \tag{25}$$

where

$$\mathbf{B}_{m1}^e = \frac{1}{2A_e} \begin{bmatrix} b-c & 0 & 0 & 0 & 0 \\ 0 & d-a & 0 & 0 & 0 \\ d-a & b-c & 0 & 0 & 0 \end{bmatrix}, \tag{26}$$

$$\mathbf{B}_{m2}^e = \frac{1}{2A_e} \begin{bmatrix} c & 0 & 0 & 0 & 0 \\ 0 & -d & 0 & 0 & 0 \\ -d & c & 0 & 0 & 0 \end{bmatrix}, \tag{27}$$

$$\mathbf{B}_{m3}^e = \frac{1}{2A_e} \begin{bmatrix} -b & 0 & 0 & 0 & 0 \\ 0 & a & 0 & 0 & 0 \\ a & -b & 0 & 0 & 0 \end{bmatrix}, \tag{28}$$

$$\mathbf{B}_{b1}^e = \frac{1}{2A_e} \begin{bmatrix} 0 & 0 & 0 & b-c & 0 \\ 0 & 0 & 0 & 0 & d-a \\ 0 & 0 & 0 & d-a & b-c \end{bmatrix}, \tag{29}$$

$$\mathbf{B}_{b2}^e = \frac{1}{2A_e} \begin{bmatrix} 0 & 0 & 0 & c & 0 \\ 0 & 0 & 0 & 0 & -d \\ 0 & 0 & 0 & -d & c \end{bmatrix}, \tag{30}$$

$$\mathbf{B}_{b3}^e = \frac{1}{2A_e} \begin{bmatrix} 0 & 0 & 0 & -b & 0 \\ 0 & 0 & 0 & 0 & a \\ 0 & 0 & 0 & a & -b \end{bmatrix}, \tag{31}$$

To eliminate the shear locking phenomenon as the thickness of the plate becomes small, the formulation of the transverse shear strains of the MITC3 element based on FSDT [36] in this study can be written as follows:

$$\boldsymbol{\gamma}^e = \mathbf{B}_s^e \mathbf{d}^e, \tag{32}$$

where

$$\mathbf{B}_s^e = [\,\mathbf{B}_{s1}^e \;\; \mathbf{B}_{s2}^e \;\; \mathbf{B}_{s3}^e\,], \tag{33}$$

with

$$\mathbf{B}_{s1}^e = \mathbf{J}^{-1}\begin{bmatrix} 0 & 0 & -1 & \frac{a}{3}+\frac{d}{6} & \frac{b}{3}+\frac{c}{6} \\ 0 & 0 & -1 & \frac{d}{3}+\frac{a}{6} & \frac{c}{3}+\frac{b}{6} \end{bmatrix}, \tag{34}$$

$$\mathbf{B}_{s2}^e = \mathbf{J}^{-1}\begin{bmatrix} 0 & 0 & 1 & \frac{a}{2}-\frac{d}{6} & \frac{b}{2}-\frac{c}{6} \\ 0 & 0 & 0 & \frac{d}{6} & \frac{c}{6} \end{bmatrix}, \tag{35}$$

$$\mathbf{B}_{s3}^{e(0)} = \mathbf{J}^{-1}\begin{bmatrix} 0 & 0 & 0 & \frac{a}{6} & \frac{b}{6} \\ 0 & 0 & 1 & \frac{d}{2}-\frac{a}{6} & \frac{c}{2}-\frac{b}{6} \end{bmatrix}, \tag{36}$$

where

$$\mathbf{J}^{-1} = \frac{1}{2A_e}\begin{bmatrix} c & -b \\ -d & a \end{bmatrix}. \tag{37}$$

Here, $a = x_2 - x_1$, $b = y_2 - y_1$, $c = y_3 - y_1$, and $d = x_3 - x_1$ are pointed out in and A_e is the area of the three-node triangular element as shown in Figure 3.

Substituting the discrete displacement field into equation (22), we obtained the discrete system equations for free vibration analysis of FGP plate resting on the EF, respectively, as

$$\left(\mathbf{K} - \omega^2\mathbf{M}\right)\mathbf{d} = 0, \tag{38}$$

where $\mathbf{K}$ and $\mathbf{M}$ are the stiffness and mass matrices, respectively.

The stiffness matrix in equation (38) can be written as

$$\mathbf{K} = \sum_{e=1}^{n^e}\left(\mathbf{K}_p^e + \mathbf{K}_f^e\right), \tag{39}$$

where

$$\mathbf{K}_p^e = \int_{\psi_e}\mathbf{B}^T\mathbf{D}\mathbf{B}\,\mathrm{d}\psi_e + \int_{\psi_e}\mathbf{B}_s^T\mathbf{C}\mathbf{B}_s\mathrm{d}\psi_e, \tag{40}$$

$$\mathbf{K}_f^e = k_1\int_{\psi_e}\mathbf{N}_w^{\mathrm{T}}\mathbf{N}_w\mathrm{d}\psi_e + k_2\int_{\psi_e}\left[\left(\frac{\partial\mathbf{N}_w}{\partial x}\right)^{\mathrm{T}}\left(\frac{\partial\mathbf{N}_w}{\partial x}\right) + \left(\frac{\partial\mathbf{N}_w}{\partial y}\right)^{\mathrm{T}}\left(\frac{\partial\mathbf{N}_w}{\partial y}\right)\right]\mathrm{d}\psi_e, \tag{41}$$

where

$$\mathbf{B}^e = [\,\mathbf{B}_m^e \;\; \mathbf{B}_b^e\,], \tag{42}$$

$$\mathbf{N}_w = [0\,0\,N_1\,0\,0,\; 0\,0\,N_2\,0\,0,\; 0\,0\,N_3\,0\,0]. \tag{43}$$

Next, the mass matrix in equation (38) can be defined as

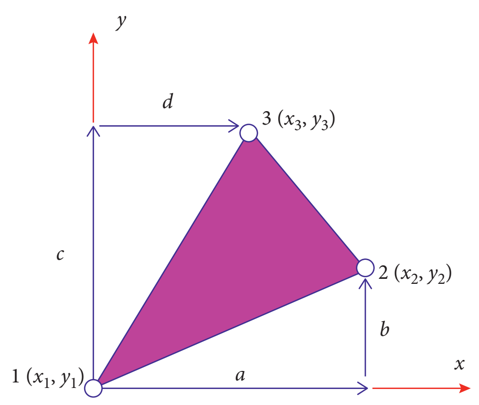


Figure 3: Three-node triangular element in the local coordinates.

$$\mathbf{M} = \sum_{e=1}^{n^e}\left(\mathbf{M}_p^e + \mathbf{M}_f^e\right), \tag{44}$$

where

$$\mathbf{M}_p^e = \int_{\psi_e}\mathbf{N}^T\mathbf{m}_\mathrm{p}\mathbf{N}\,\mathrm{d}\psi_e, \tag{45}$$

$$\mathbf{M}_f^e = \mathbf{m}_\mathrm{f}\int_{\psi_e}\mathbf{N}_w^T\mathbf{N}_w\,\mathrm{d}\psi_e. \tag{46}$$

4.2. Formulation of an ES-MITC3 Method for FGP Plates. The smoothing domains ψ^k is constructed based on edges of the triangular elements such that $\psi = \cup_{k=1}^{n^k}\psi^k$ and $\psi_i^k \cap \psi_j^k = \varnothing$ for $i \neq j$. An edge-based smoothing domain ψ^k for the inner edge k is formed by connecting two end-nodes of the edge to the centroids of adjacent triangular MITC3 elements, as shown in Figure 4.

Applying the edge-based smooth technique [36], the smoothed membrane, bending, and shear strain $\tilde{\boldsymbol{\varepsilon}}_m^k$, $\tilde{\boldsymbol{\kappa}}^k$, $\tilde{\boldsymbol{\gamma}}^k$ over the smoothing domain ψ^k can be created by

$$\tilde{\boldsymbol{\varepsilon}}_m^k = \int_{\psi^k}\boldsymbol{\varepsilon}_m\Phi^k(\mathbf{x})\mathrm{d}\psi, \tag{47}$$

$$\tilde{\boldsymbol{\kappa}}^k = \int_{\psi^k}\boldsymbol{\kappa}\Phi^k(\mathbf{x})\mathrm{d}\psi, \tag{48}$$

$$\tilde{\boldsymbol{\gamma}}^k = \int_{\psi^k}\boldsymbol{\gamma}\Phi^k(\mathbf{x})\mathrm{d}\psi, \tag{49}$$

where $\boldsymbol{\varepsilon}_m$, $\boldsymbol{\kappa}$, and $\boldsymbol{\gamma}$ the compatible membrane, bending, and the shear strains, respectively; $\Phi^k(\mathbf{x})$ is a given smoothing function that satisfies at least unity property $\int_{\psi^k}\Phi^k(\mathbf{x})\mathrm{d}\psi = 1$.

In this study, we use the constant smoothing function

$$\Phi^k(\mathbf{x}) = \begin{cases} \dfrac{1}{A^k}, & x \in \psi^k, \\ 0, & x \notin \psi^k, \end{cases} \tag{50}$$

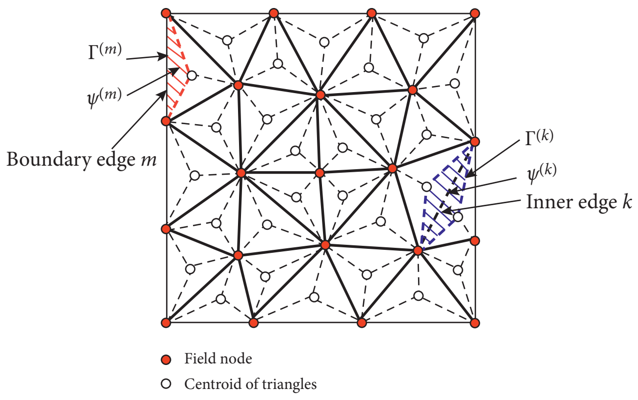


Figure 4: The smoothing domain ψ^k is formed by triangular elements.

where A^k is the area of the smoothing domain ψ^k and is given by

$$A^k = \int_{\psi^k} \mathrm{d}\psi = \frac{1}{3}\sum_{i=1}^{n^{ek}} A^i, \tag{51}$$

where n^{ek} is the number of the adjacent triangular elements in the smoothing domain ψ^k; and A^i is the area of the ith triangular element attached to the edge k.

By substituting equations (47)–(49) into equations (24), (25), and (32), the approximation of the smoothed strains on the smoothing domain ψ^k can be expressed as follows:

$$\begin{aligned}\tilde{\boldsymbol{\varepsilon}}_m^k &= \sum_{j=1}^{n^{nk}} \tilde{\mathbf{B}}_{mj}^k \mathbf{d}_j^k;\\ \tilde{\mathbf{k}}^k &= \sum_{j=1}^{n_{sh}^{nk}} \tilde{\mathbf{B}}_{bj}^k \mathbf{d}_j^k;\\ \boldsymbol{\gamma}^k &= \sum_{j=1}^{n_{sh}^{nk}} \tilde{\mathbf{B}}_{sj}^k \mathbf{d}_j^k;\end{aligned} \tag{52}$$

where n_{sh}^{nk} is the total number of nodes of the triangular MITC3 elements attached to edge k ($n_{sh}^{nk} = 3$ for boundary edges and $n_{sh}^{nk} = 4$ for inner edges as given in Figure 4; $\mathbf{d}_j^k$ is the nodal d.o.f associated with the smoothing domain ψ^k; and $\tilde{\mathbf{B}}_{mj}^k$, $\tilde{\mathbf{B}}_{bj}^k$, and $\tilde{\mathbf{B}}_{sj}^k$ are the smoothed membrane, bending, and shear strain gradient matrices, respectively, at the j^{th} node of the elements attached to edge k computed by

$$\tilde{\mathbf{B}}_{mj}^k = \frac{1}{A^k}\sum_{i=1}^{n^{ek}} \frac{1}{3} A^i \mathbf{B}_{mj}^e, \tag{53}$$

$$\tilde{B}_{bj}^k = \frac{1}{A^k}\sum_{i=1}^{n^{ek}} \frac{1}{3} A_i \mathbf{B}_{\mathrm{bj}}^e, \tag{54}$$

$$\tilde{B}_{\mathrm{sj}}^k = \frac{1}{A^k}\sum_{i=1}^{n^{ek}} \frac{1}{3} A^i \mathbf{B}_{\mathrm{sj}}^e. \tag{55}$$

The stiffness matrix of the FGP plate using the ES-MITC3 is assembled by

$$\tilde{\mathbf{K}} = \sum_{\mathrm{k}=1}^{n_{\mathrm{sh}}^k} \tilde{\mathbf{K}}^k, \tag{56}$$

where $\tilde{\mathbf{K}}^k$ is the ES-MITC3 stiffness matrix of the smoothing domain ψ^k and given by

$$\tilde{\mathbf{K}}^k = \int_{\psi^k}\left(\tilde{\mathbf{B}}^{\mathrm{KT}}\mathbf{D}\tilde{\mathbf{B}}^k + \tilde{\mathbf{B}}_s^{kt}\mathbf{C}\tilde{\mathbf{B}}_s^k\right)\mathrm{d}\psi = \tilde{\mathbf{B}}^{\mathrm{KT}}\mathbf{D}\tilde{\mathbf{B}}^k A^k + \tilde{\mathbf{B}}_s^{kt}\mathbf{C}\tilde{\mathbf{B}}_s^k A^k, \tag{57}$$

where

$$\tilde{\mathbf{B}}^{kT} = \left[\tilde{\mathbf{B}}_{\mathrm{mj}}^k \;\; \tilde{\mathbf{B}}_{\mathrm{bj}}^k\right]. \tag{58}$$

5. Accuracy of the Proposed Method

In this section, the various numerical examples are solved to verify the reliability and accuracy of the proposed method. For convenience, the stiffness factors and nondimensional frequencies of the plates are defined as the following equations:

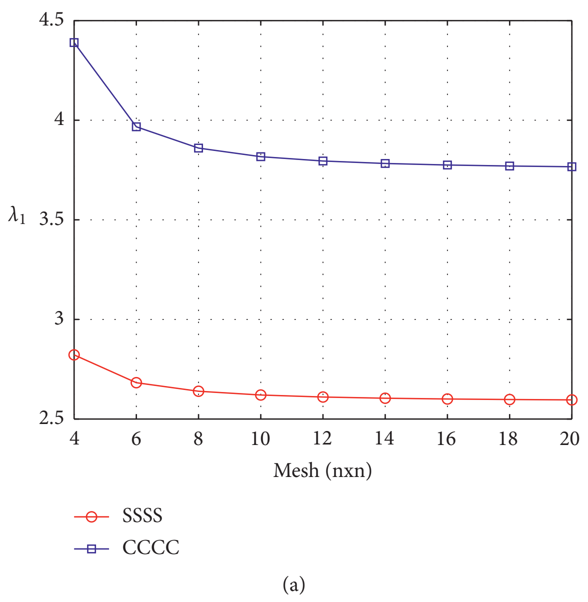
$$\begin{aligned}K_1 &= \frac{k_1 a^4}{H};\\ K_2 &= \frac{k_2 a^2}{H};\\ \lambda &= \frac{\omega a^2}{\pi^2}\sqrt{\frac{\rho h}{H}}, \quad \text{with } H = \frac{Eh^3}{12(1-\nu^2)}.\end{aligned} \tag{59}$$

To demonstrate the performance of numerical results, the relative frequency error is defined by

$$\Delta\ (\%) = 100 \times \frac{\left|\lambda_{pr} - \lambda_{re}\right|}{\left|\lambda_{re}\right|}, \tag{60}$$

where λ_{pr} and λ_{re} are nondimensional frequencies of present method and nondimensional frequencies in [17, 18], respectively.

The results of the convergence of the first two nondimensional frequencies of the plate in the case of fully simple support (SSSS) plate and a fully clamped (CCCC) plate with $h/a = 0.1$, $K_1 = 100$, $K_2 = 10$, respectively, are shown in Figure 5. From these results, it can be seen that almost all frequencies corresponding to different cases of boundary conditions (BC) converge with 18×18 element mesh. For 18×18 mesh, we compare the first three nondimensional frequencies of a plate resting on the Winkler–Pasternak foundation with the published results as shown in Table 1. It can be seen that the present results agree well with the results of the authors using analytical methods [5, 14, 17] and are more accurate than those using the original MITC3 element and FEM [6]. In addition, from Table 2, it is obvious that the relative error of the present results compared to [18] is less than 2%. In [18], they used a new quasi-3D hyperbolic theory to investigate the free vibration of the FGP plate resting on the EF. These results are the basis to analyze the free vibration of FGP plates on the EFTIM.



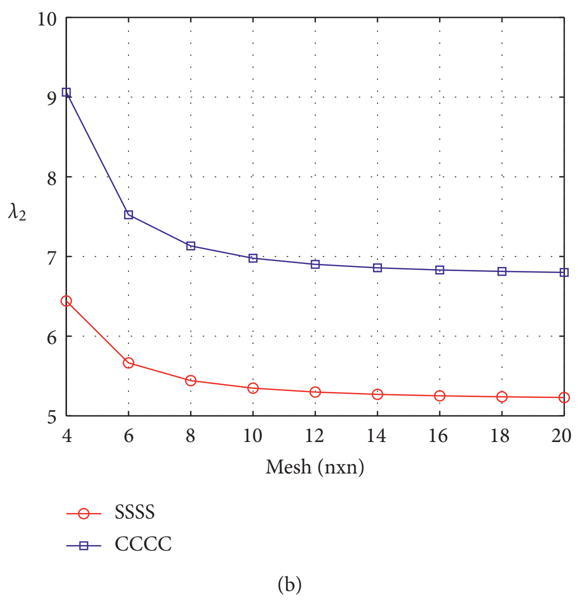


FIGURE 5: The convergence of element mesh to nondimensional frequency of plate. (a)λ_1 and (b)λ_2.

TABLE 1: Nondimensional frequencies of plates.

Plates	K_1	K_2	Author	λ_1	Δ (%)	λ_2	Δ (%)	λ_3	Δ (%)
SSSS $v = 0.3$ $h/a = 0.01$	100	10	Ferreira et al. [17]	2.6559		5.5718		8.5384	
			Zhou et al. [14]	2.6551	0.03	5.5717	0.00	8.5406	0.03
			Xiang et al. [5]	2.6551	0.03	5.5718	0.00	8.5405	0.02
			MITC3	2.6604	0.17	5.6103	0.70	8.6296	1.07
			Present	2.6590	0.12	5.5920	0.37	8.6017	0.74
	500	10	Ferreira [17]	3.3406		5.9285		8.7754	
			Zhou et al. [14]	3.3398	0.02	5.9285	0.00	8.7775	0.02
			Xiang et al. [5]	3.3400	0.02	5.9287	0.00	8.7775	0.02
			MITC3	3.3441	0.10	5.9649	0.61	8.8642	1.01
			Present	3.3430	0.07	5.9477	0.32	8.8370	0.70
SSSS $v = 0.3$ $h/a = 0.1$	200	10	Ferreira et al. [17]	2.7902		5.3452		7.8255	
			Zhou et al. [14]	2.7756	0.52	5.2954	0.93	7.7279	1.25
			Xiang et al. [5]	2.7842	0.22	5.3043	0.77	7.7287	1.24
			MITC3	2.7874	0.10	5.3258	0.36	7.7719	0.68
			Present	2.7887	0.05	5.3362	0.17	7.7971	0.36
	1000	10	Ferreira et al. [17]	3.9844		6.0430		8.3112	
			Zhou et al. [14]	3.9566	0.70	5.9757	1.11	8.1954	1.39
			Xiang et al. [5]	3.9805	0.10	6.0078	0.58	8.2214	1.08
			MITC3	3.9827	0.04	6.0266	0.27	8.2619	0.59
			Present	3.9836	0.02	6.0358	0.12	8.2856	0.31
CCCC $v = 0.15$ $h/a = 0.015$	1390.2	166.83	Ferreira et al. [17]	8.1669		12.821		16.842	
			Zhou et al. [14]	8.1675	0.01	12.823	0.02	16.833	0.05
			Omurtag et al. [6]	8.1375	0.36	12.898	0.60	16.932	0.53
			MITC3	8.1842	0.21	12.909	0.69	17.010	1.00
			Present	8.1729	0.07	12.872	0.40	16.939	0.58

Next, we consider an SSSS FGP plate (Al/Al_2O_3) with its material properties as follows: metal (Al) $E_b = 70\,\text{GPa}$, $\rho_b = 2702\,\text{kg/m}^3$ and ceramic (Al_2O_3) $E_t = 380\,\text{GPa}$, $\rho_t = 3800\,\text{kg/m}^3$. Poisson's ratio is fixed at $v = 0.3$. The FGP plate with even porosities is expressed as in [18]:

$$P(z) = P_b + (P_t - P_b)\left(\frac{z}{h} + 0.5\right)^k - \frac{\xi}{2}(P_t + P_b), \quad (61)$$

where $\xi\,(\xi \le 1)$ presents the porosity volume fraction. The stiffness factor and nondimensional frequencies of the plates are shown in equation (58) with $H_b = (E_b h^3/12(1 - v^2))$ and

Table 2: The first nondimensional frequencies of FGP plate according to the Winkler–Pasternak foundation stiffness ($k = 1$).

(K_1, K_2)	h/a	$\xi = 0$			$\xi = 0.2$		
		Present	[18]	Δ (%)	Present	[18]	Δ (%)
(0, 0)	0.05	9.010	9.020	0.11	8.485	8.370	1.37
	0.10	8.823	8.818	0.06	8.319	8.203	1.41
	0.15	8.541	8.516	0.29	8.069	7.950	1.50
	0.20	8.196	8.151	0.55	7.762	7.641	1.58
(100, 0)	0.05	9.389	9.430	0.43	9.020	8.917	1.16
	0.10	9.207	9.231	0.26	8.858	8.753	1.20
	0.15	8.933	8.934	0.01	8.614	8.505	1.28
	0.20	8.599	8.577	0.26	8.315	8.203	1.37
(100, 100)	0.05	15.383	15.439	0.36	16.338	16.320	0.11
	0.10	15.213	15.245	0.21	16.175	16.148	0.17
	0.15	14.962	14.966	0.03	15.932	15.895	0.23
	0.20	14.664	14.640	0.16	15.639	15.595	0.28

Table 3: Nondimensional frequencies of the FGP plate on EFTIM.

λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
0.8583	1.8118	1.8157	2.8015	3.3898	3.4248

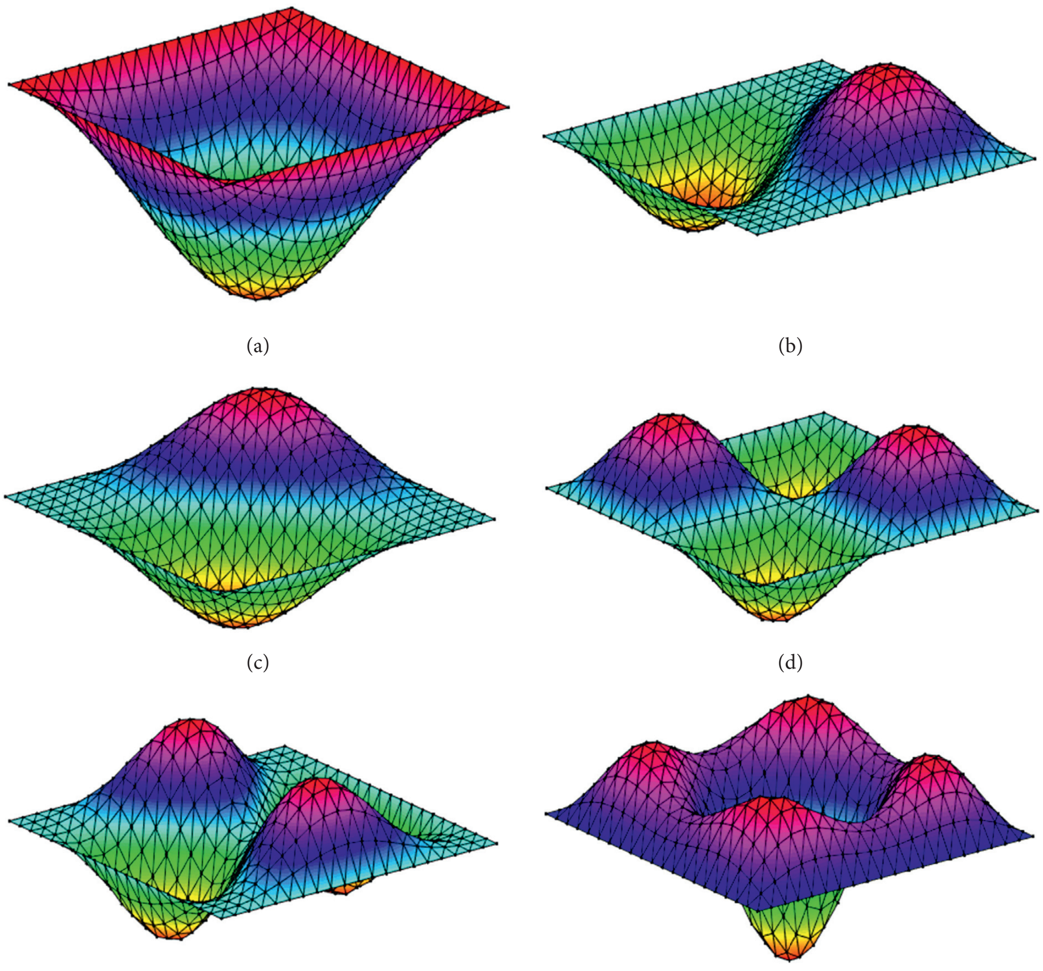
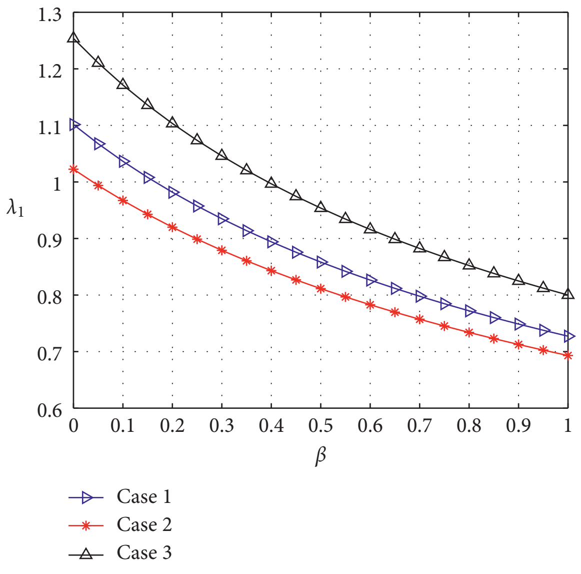
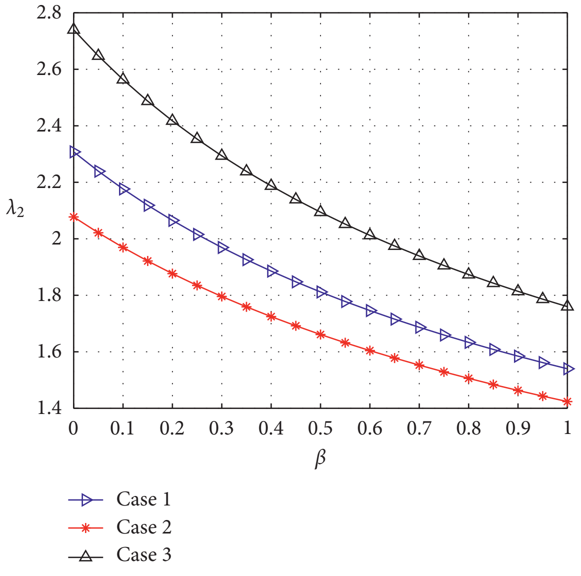


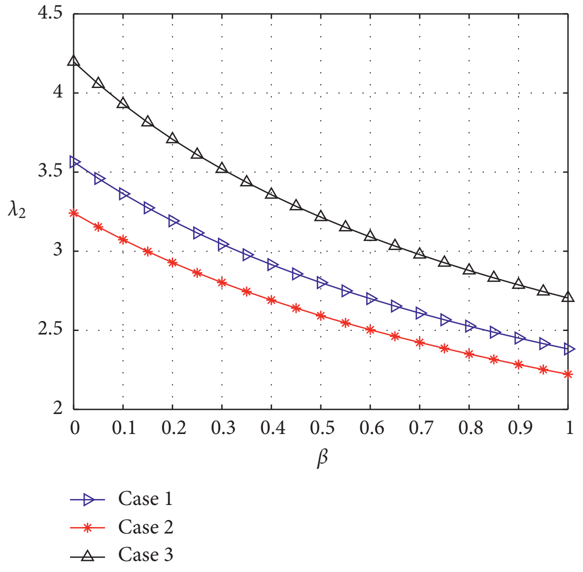
Figure 6: The first six mode sharps the FGP plate on EFTIM.



(a)



(b)



(c)

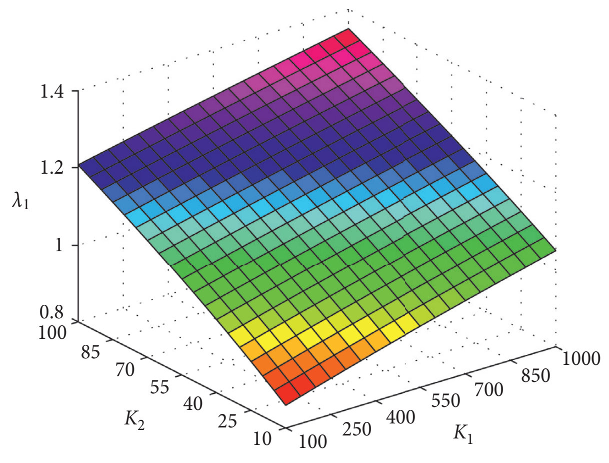
FIGURE 7: Nondimensional frequencies of the eFGP plate with difference of featured-index β: (a) nondimensional frequency λ_1; (b) nondimensional frequency λ_2; and (c) nondimensional frequency λ_3.

TABLE 4: The first three nondimensional frequencies of FGP on EFTIM.

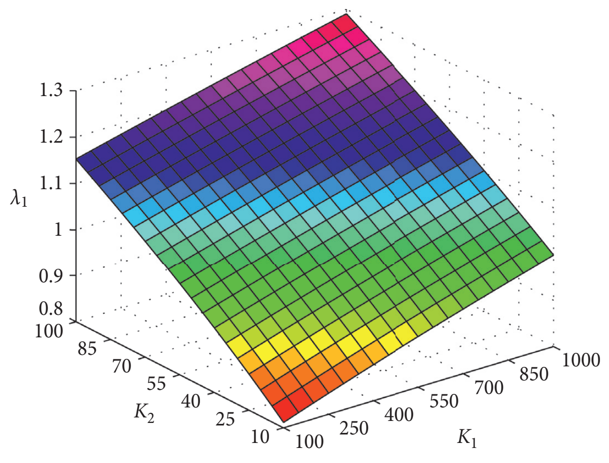
Parameter of plate	β	0	0.25	0.5	0.75	1
Case 1	λ_1	1.1015	0.9575	0.8583	0.7847	0.7273
($K_1 = 100$, $K_2 = 10$)	λ_2	2.3078	2.0156	1.8118	1.6595	1.5400
(SSSS)	λ_3	3.5651	3.1154	2.8015	2.5665	2.3822
Case 2	λ_1	1.0225	0.8988	0.8113	0.7453	0.6931
($K_1 = 100$, $K_2 = 10$)	λ_2	2.0774	1.8350	1.6612	1.5289	1.4239
(SSSS)	λ_3	3.2410	2.8630	2.5921	2.3859	2.2221
Case 3	λ_1	1.2537	1.0735	0.9538	0.8669	0.8001
($K_1 = 100$, $K_2 = 10$)	λ_2	2.7399	2.3531	2.0942	1.9054	1.7599
(SSSS)	λ_3	4.1972	3.6097	3.2152	2.9269	2.7044

Table 5: Nondimensional frequencies of the FGP plate with different K_1 and K_2.

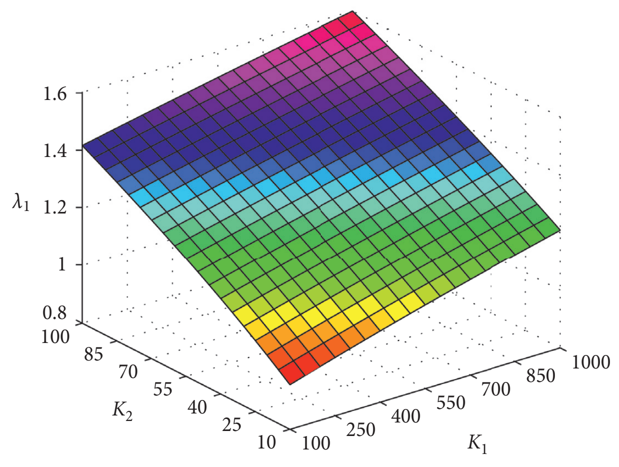
Case of Porosity distribution	K_2	K_1 100	250	500	750	1000
Case 1 (SSSS)	10	0.8583	0.8929	0.9477	0.9995	1.0488
	25	0.9258	0.9579	1.0092	1.0580	1.1047
	50	1.0285	1.0575	1.1042	1.1490	1.1921
	75	1.1218	1.1485	1.1916	1.2332	1.2734
	100	1.2079	1.2327	1.2730	1.3120	1.3499
Case 2 (SSSS)	10	0.8113	0.8452	0.8988	0.9493	0.9973
	25	0.8775	0.9089	0.9589	1.0064	1.0518
	50	0.9778	1.0061	1.0515	1.0950	1.1369
	75	1.0687	1.0947	1.1365	1.1769	1.2160
	100	1.1525	1.1765	1.2156	1.2534	1.2902
Case 3 (SSSS)	10	0.9538	1.0009	1.0747	1.1438	1.2090
	25	1.0452	1.0883	1.1566	1.2211	1.2823
	50	1.1819	1.2202	1.2815	1.3400	1.3960
	75	1.3044	1.3392	1.3953	1.4492	1.5011
	100	1.4163	1.4484	1.5004	1.5507	1.5993



(a)



(b)

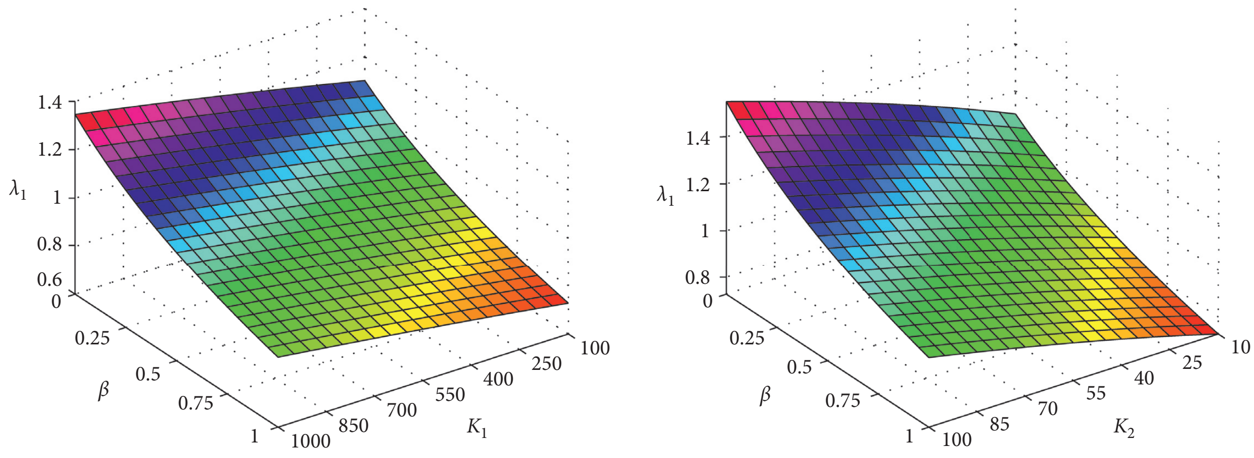


(c)

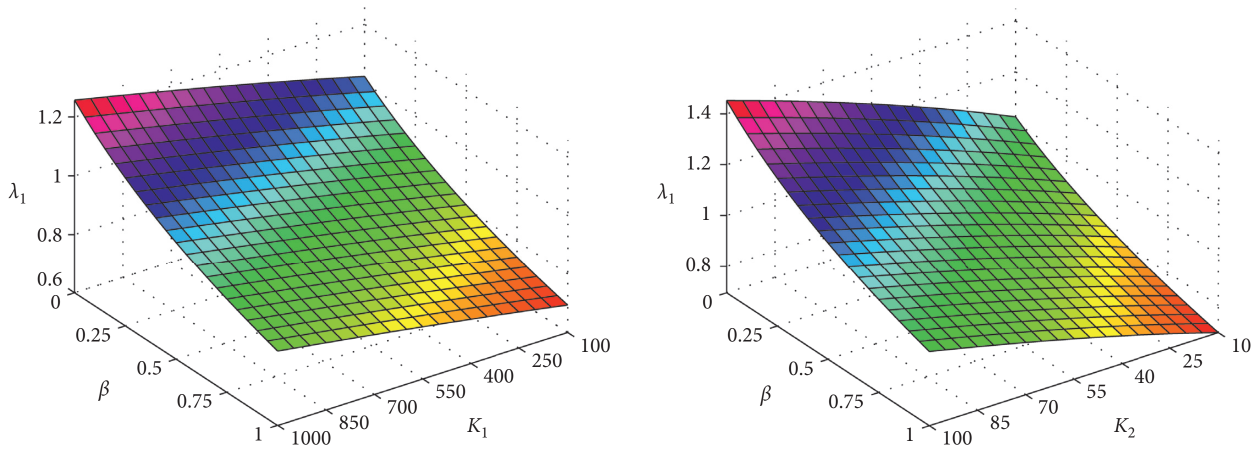
Figure 8: Nondimensional frequencies of FGP plate with different K_1 and K_2. (a) Case 1; (b) Case 2; and (c) Case 3.

Table 6: Nondimensional frequencies of FGP plate with different K_1, K_2, and β.

β	K_1 $(K_2 = 10)$					K_2 $(K_1 = 100)$				
Case 1										
	100	250	500	750	1000	10	25	50	75	100
0	1.1015	1.1459	1.2163	1.2828	1.3460	1.1015	1.1882	1.3199	1.4397	1.5502
0.25	0.9575	0.9961	1.0572	1.1150	1.1700	0.9575	1.0328	1.1473	1.2514	1.3475
0.5	0.8583	0.8929	0.9477	0.9995	1.0488	0.8583	0.9258	1.0285	1.1218	1.2079
0.75	0.7847	0.8163	0.8664	0.9138	0.9588	0.7847	0.8464	0.9403	1.0256	1.1043
1	0.7273	0.7566	0.8030	0.8469	0.8887	0.7273	0.7844	0.8715	0.9505	1.0235
Case 2										
0	1.0225	1.0652	1.1327	1.1964	1.2569	1.0225	1.1059	1.2323	1.3469	1.4525
0.25	0.8988	0.9363	0.9957	1.0517	1.1049	0.8988	0.9721	1.0833	1.1840	1.2768
0.5	0.8113	0.8452	0.8988	0.9493	0.9973	0.8113	0.8775	0.9778	1.0687	1.1525
0.75	0.7453	0.7763	0.8256	0.8720	0.9161	0.7453	0.8060	0.8982	0.9817	1.0586
1	0.6931	0.7220	0.7677	0.8109	0.8519	0.6931	0.7496	0.8353	0.9129	0.9845
Case 3										
0	1.2537	1.3156	1.4127	1.5035	1.5891	1.2537	1.3739	1.5536	1.7146	1.8617
0.25	1.0735	1.1265	1.2096	1.2874	1.3607	1.0735	1.1764	1.3303	1.4681	1.5941
0.5	0.9538	1.0009	1.0747	1.1438	1.2090	0.9538	1.0452	1.1819	1.3044	1.4163
0.75	0.8669	0.9097	0.9768	1.0396	1.0988	0.8669	0.9500	1.0742	1.1855	1.2872
1	0.8001	0.8395	0.9015	0.9595	1.0141	0.8001	0.8767	0.9914	1.0942	1.1880

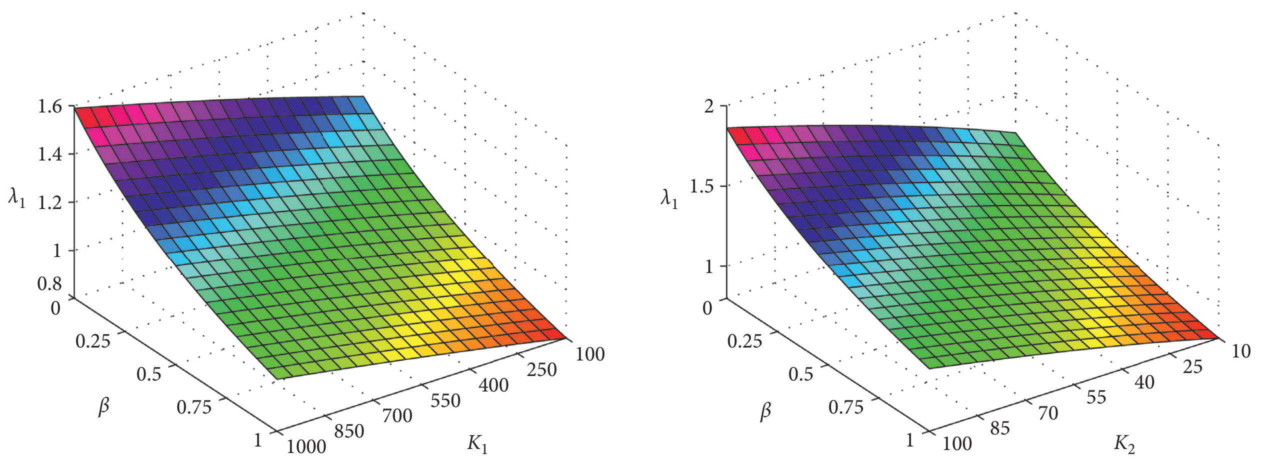


(a)



(b)

Figure 9: Continued.



(c)

FIGURE 9: Nondimensional frequencies of FGP plate with different K_1, K_2, and β. (a) Case 1; (b) Case 2; and (c) Case 3.

TABLE 7: Nondimensional frequencies of the FGP plate as a function of k and Ω.

Ω	SSSS					CCCC				
	k					k				
	0	2.5	5	7.5	10	0	2.5	5	7.5	10
Case 1										
0	0.6146	0.8460	0.9468	1.0184	1.0706	1.0275	1.2153	1.3432	1.4571	1.5444
0.25	0.6584	0.8830	0.9777	1.0469	1.0984	1.0935	1.2518	1.3694	1.4783	1.5633
0.5	0.7126	0.9238	1.0105	1.0767	1.1273	1.1732	1.2869	1.3930	1.4958	1.5780
0.75	0.7820	0.9677	1.0446	1.1070	1.1564	1.2711	1.3154	1.4102	1.5061	1.5848
1	0.8745	1.0105	1.0776	1.1357	1.1836	1.3924	1.3226	1.4130	1.5021	1.5766
Case 2										
0	0.6146	0.8460	0.9468	1.0184	1.0706	1.0275	1.2153	1.3432	1.4571	1.5444
0.25	0.6609	0.8604	0.9563	1.0288	1.0833	1.0977	1.2011	1.3156	1.4273	1.5162
0.5	0.7162	0.8685	0.9569	1.0299	1.0871	1.1707	1.1651	1.2611	1.3675	1.4564
0.75	0.7777	0.8600	0.9357	1.0073	1.0671	1.2179	1.0868	1.1556	1.2496	1.3342
1	0.7852	0.7972	0.8420	0.9009	0.9580	1.0347	0.9106	0.9310	0.9904	1.0547
Case 3										
0	0.6146	0.8460	0.9468	1.0184	1.0706	1.0275	1.2153	1.3432	1.4571	1.5444
0.25	0.6609	0.9413	1.0475	1.1196	1.1714	1.0977	1.3661	1.5012	1.6147	1.6994
0.5	0.7162	1.0785	1.1908	1.2621	1.3122	1.1707	1.5836	1.7243	1.8317	1.9084
0.75	0.7777	1.3066	1.4256	1.4917	1.5363	1.2179	1.9297	2.0656	2.1484	2.2018
1	0.7852	1.8589	1.9631	1.9772	1.9807	1.0347	2.4083	2.4138	2.3588	2.3246

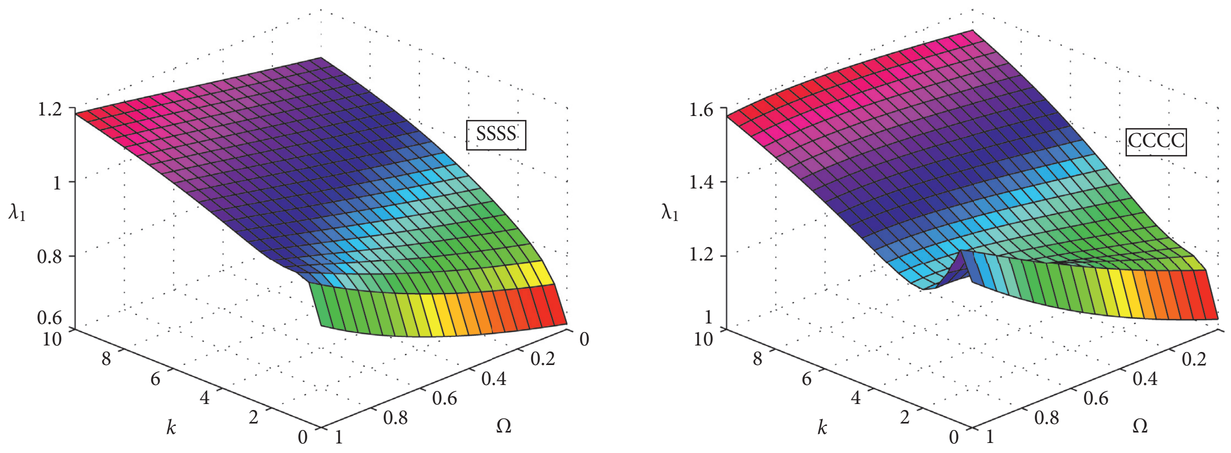
$\omega^* = (\omega a^2/h)\sqrt{(\rho_b/E_b)}$. The first nondimensional frequencies of present work compared with [18] are shown in Table 2.

6. Numerical Results and Discussions

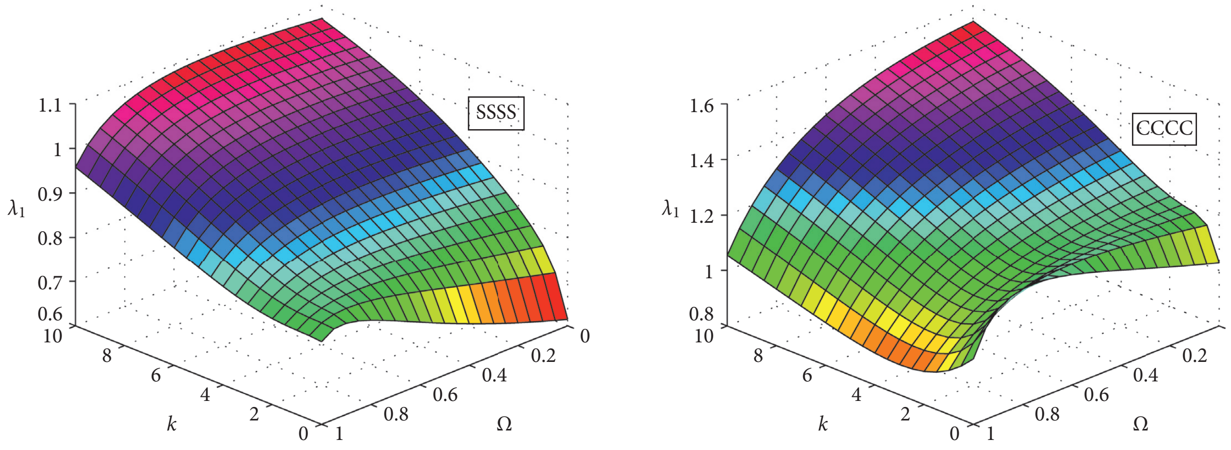
For free vibration problems, a fully simple support (SSSS) FGP plate is considered, wherein $a = b$, $h = a/10$, $\Omega = 0.5$, $k = 1$, $E_t = 10E_b$, $\rho_t = 10\rho_b$, $v = 0.38$, and $\mu_F = 0.5$. The first six nondimensional frequencies of the FGP plate with porosity distribution of Case 1 and stiffener of foundation $K_1 = 100$, $K_2 = 10$ are shown in Table 3; and the first six mode shapes are presented in Figure 6. The stiffness factors and nondimensional frequencies of FGP plate are shown in equation (58) with $H_b = (E_b h^3/12(1 - v^2))$.

6.1. Influence of the Parameters of the EFTIM to the Free Vibration for the FGP Plate

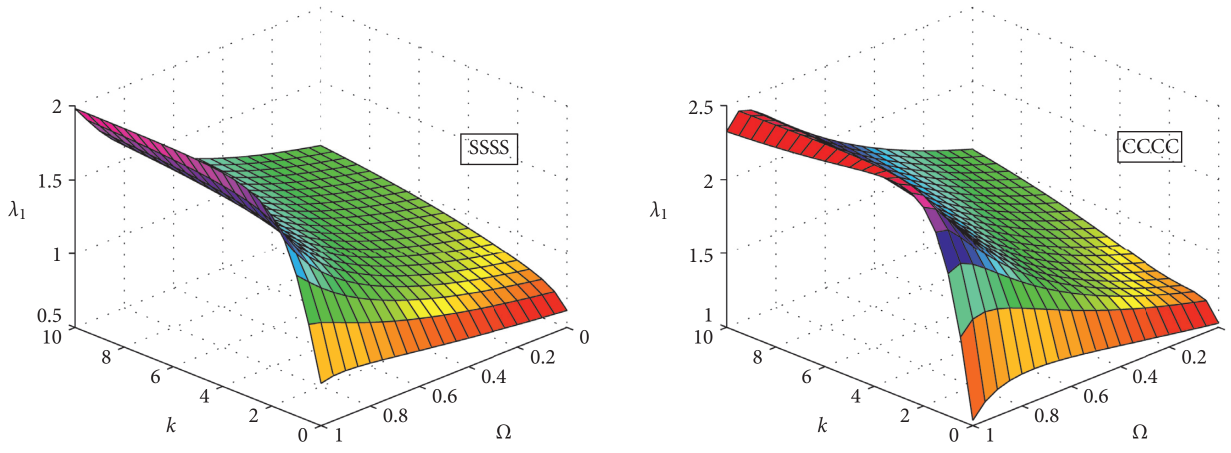
6.1. Influence of the Parameters of the EFTIM to the Free Vibration for the FGP Plate. Firstly, in order to investigate the effect of the featured-index of the mass of foundation β to free vibration of the FGP plate, the featured-index of mass is changed from 0 to 1. In Figure 7 and Table 4, it is seen that all cases of porosity distribution featured-index of the mass of foundation β significantly influence to free vibration of the FGP plate. As β increases, the mass of the plate increases, and the frequencies of the plate decrease. For all cases of porosity distribution of the FGP plate, the porosity distribution of



(a)



(b)



(c)

Figure 10: Nondimensional frequencies vibration of the FGP plate as a function of k. (a) Case 1; (b) Case 2; and (c) Case 3.

Case 3 leads to the maximum values of frequencies of the plate, while the porosity distribution of Case 2 leads to the minimum values. It can be observed that the FGP plate with porosity distribution Case 3 is stiffer than plates with other porosity distributions.

Next, the influence of nondimensional parameters of foundation stiffness K_1 and K_2 is investigated. We change K_1 from 100 to 1000 and K_2 from 10 to 100 with respect to $\beta = 0.5$ and $\mu_F = 0.5$. The first nondimensional frequency of the FGP plate with three cases of porosity distribution is presented in Table 5 and shown in Figure 8. As shown in these figure and table, when K_1 and K_2 increase, the nondimensional frequency of plate also increases. We also examine the effect of (β, K_1) and (β, K_2) parameters to nondimensional frequency. The numerical result is presented in Table 6 and Figure 9. Consequently, Winkler foundation stiffness K_1 and shear layer stiffness of Pasternak foundation K_2 make stiffness of plate become greater and the mass of the EF involved in the plate's vibration makes reduce frequencies.

6.2. Influence of the Parameters-FGP to Free Vibration of the Plate on EFTIM. Let us consider the effect of materials property to free vibration of the FGP plate. The power-law index k is changed from 0 to 10, and maximum porosity distributions Ω has the value from 0 to 1. We examine the SSSS FGP plate and fully clamped (CCCC) plate resting on EFTIM. The parameters of EFTIM are given by $\beta = 0.5$, $\mu_F = 0.5$, $K_1 = 100$, and $K_1 = 10$. The first nondimensional frequencies of plate with three cases of porosity distribution is shown in Table 7 and Figure 10.

As shown in these figures and tables, when k and Ω change, the values of nondimensional frequency change with no rule. It is understandable because with each case of change in porosity distributions k and Ω, the stiffness and the weight of the plate changes. From Figure 10, in the case of the CCCC plate, nondimensional frequency depending on k and Ω value varies more complex than the case of the SSSS plate. If k and Ω values are same, the frequency of the CCCC plate is larger than that of the SSSS plate. The results are quite reasonable because the SSSS boundary condition inherently offers more flexible boundary conditions than the CCCC boundary condition.

7. Conclusions

In this paper, new numerical results of free vibration of the FGP plate resting on EFTIM are studied. We used the ES-MITC3 to establish the fundamental equation of the FGP plate. The computed results obtained by this approach are in excellent agreement with others published. Our work has the following advantages.

The novel ES-MITC3 which computes the free vibration of the plate on EF takes into account the mass of foundation.

The numerical results obtained by ES-MITC3 show good agreement with the reference solutions and are more accurate than those obtained by the original MITC3.

The elastic foundation of Pasternak with three-parameters is developed by adding the featured-index of mass β to accurately describe the actual elastic foundation.

The mass of the elastic foundation involved in the vibration of the plate reduces the frequency of vibration, while two parameters K_1 and K_2 effect the stiffness of the plate.

The material parameters k, Ω and the case of porosity distribution effect of vibration of the plate. Numerical results are useful for calculation, design, and testing of material parameters in engineering and technologies.

This study suggests some further works on the dynamic response and heat transfer problems of the FGP plate resting on EF using different plate theories.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was funded by the Vietnam National Foundation for Science and Technology Development (NAFOSTED) under Grant no. 107.02-2019.330.

References

[1] E. Winkler, *Die Lehre von der Elasticitaet und Festigkeit*, Prag, Dominicus, 1867.

[2] P. L. Pasternak, *On a New Method of Analysis of an Elastic Foundation by Means of Two Foundation Constants*, Gosudarstvennoe Izdatelstvo Literaturi po Stroitelstvu i Arkhitekture, Moscow, Russia, in Russian, 1954.

[3] F. A. Fazzolari, "Generalized exponential, polynomial and trigonometric theories for vibration and stability analysis of porous FG sandwich beams resting on elastic foundations," *Composites Part B: Engineering*, vol. 136, pp. 254–271, 2018.

[4] A. W. Leissa, "The free vibration of rectangular plates," *Journal of Sound and Vibration*, vol. 31, no. 3, pp. 257–293, 1973.

[5] Y. Xiang, C. M. Wang, and S. Kitipornchai, "Exact vibration solution for initially stressed mindlin plates on Pasternak foundations," *International Journal of Mechanical Sciences*, vol. 36, no. 4, pp. 311–316, 1994.

[6] M. H. Omurtag, A. Özütok, A. Y. Aköz, and Y. Özçelikörs, "Free vibration analysis of Kirchhoff plates resting on elastic foundation by mixed finite element formulation based on Gâteaux differential," *International Journal for Numerical Methods in Engineering*, vol. 40, no. 2, pp. 295–317, 1997.

[7] K. Y. Özçelikörs, C. M. Wang, and X. Q. He, "Canonical exact solutions for Levy-plates on two-parameter foundation using Green's functions," *Engineering Structures*, vol. 22, no. 4, pp. 364–378, 2000.

[8] H. Matsunaga, "Vibration and stability of thick plates on elastic foundations," *Journal of Engineering Mechanics*, vol. 126, no. 1, pp. 27–34, 2000.

[9] Y. Ayvaz, A. Daloglu, and A. Dogangün, "Application of a modified Vlasov model to earthquake analysis of plates resting on elastic foundations," *Journal of Sound and Vibration*, vol. 212, no. 3, pp. 499–509, 1998.

[10] H.-S. Shen, J. Yang, and L. Zhang, "Free and forced vibration of Reissner-mindlin plates with free edges resting on elastic foundations," *Journal of Sound and Vibration*, vol. 244, no. 2, pp. 299–320, 2001.

[11] K. M. Liew, J.-B. Han, Z. M. Xiao, and H. Du, "Differential quadrature method for mindlin plates on Winkler foundations," *International Journal of Mechanical Sciences*, vol. 38, no. 4, pp. 405–421, 1996.

[12] K. M. Liew and T. M. Teo, “Differential cubature method for analysis of shear deformable rectangular plates on Pasternak foundations,” *International Journal of Mechanical Sciences*, vol. 44, no. 6, pp. 1179–1194, 2002.

[13] J.-B. Han and K. M. Liew, “Numerical differential quadrature method for Reissner/mindlin plates on two-parameter foundations,” *International Journal of Mechanical Sciences*, vol. 39, no. 9, pp. 977–989, 1997.

[14] D. Zhou, Y. K. Cheung, S. H. Lo, and F. T. K. Au, “Three-dimensional vibration analysis of rectangular thick plates on pasternak foundation,” *International Journal for Numerical Methods in Engineering*, vol. 59, no. 10, pp. 1313–1334, 2004.

[15] S. Chucheepsakul and B. Chinnaboon, “An alternative domain/boundary element technique for analyzing plates on two-parameter elastic foundations,” *Engineering Analysis with Boundary Elements*, vol. 26, no. 6, pp. 547–555, 2002.

[16] Ö. Civalek and M. H. Acar, “Discrete singular convolution method for the analysis of mindlin plates on elastic foundations,” *International Journal of Pressure Vessels and Piping*, vol. 84, no. 9, pp. 527–535, 2007.

[17] A. J. M. Ferreira, C. M. C. Roque, A. M. A. Neves, R. M. N. Jorge, and C. M. M. Soares, “Analysis of plates on Pasternak foundations by radial basis functions,” *Computational Mechanics*, vol. 46, no. 6, pp. 791–803, 2010.

[18] D. Shahsavari, M. Shahsavari, Li Li, and B. Karami, “A novel quasi-3D hyperbolic theory for free vibration of FG plates with porosities resting on Winkler/Pasternak/Kerr foundation,” *Aerospace Science and Technology*, vol. 72, pp. 134–149, 2018.

[19] A. Zenkour and A. Radwan, “Free vibration analysis of multilayered composite and soft core sandwich plates resting on Winkler-Pasternak foundations,” *Journal of Sandwich Structures & Materials*, vol. 20, no. 2, pp. 169–190, 2018.

[20] N. D. Duc, D. H. Bich, and P. H. Cong, “Nonlinear thermal dynamic response of shear deformable FGM plates on elastic foundations,” *Journal of Thermal Stresses*, vol. 39, no. 3, pp. 278–297, 2016.

[21] A. Mahmoudi, S. Benyoucef, A. Tounsi, A. Benachour, E. A. Adda Bedia, and S. Mahmoud, “A refined quasi-3D shear deformation theory for thermo-mechanical behavior of functionally graded sandwich plates on elastic foundations,” *Journal of Sandwich Structures & Materials*, vol. 21, no. 6, pp. 1906–1929, 2019.

[22] N. D. Duc, J. Lee, T. Nguyen-Thoi, and P. T. Thang, “Static response and free vibration of functionally graded carbon nanotube-reinforced composite rectangular plates resting on Winkler-Pasternak elastic foundations,” *Aerospace Science and Technology*, vol. 68, pp. 391–402, 2017.

[23] P.-T. Thang, T. Nguyen-Thoi, and J. Lee, “Closed-form expression for nonlinear analysis of imperfect sigmoid-FGM plates with variable thickness resting on elastic medium,” *Composite Structures*, vol. 143, pp. 143–150, 2016.

[24] T. Banh-Thien, H. Dang-Trung, L. Le-Anh, V. Ho-Huu, and T. Nguyen-Thoi, “Buckling analysis of non-uniform thickness nanoplates in an elastic medium using the isogeometric analysis,” *Composite Structures*, vol. 162, pp. 182–193, 2017.

[25] J. Kim, K. K. Żur, and J. N. Reddy, “Bending, free vibration, and buckling of modified couples stress-based functionally graded porous micro-plates,” *Composite Structures*, vol. 209, pp. 879–888, 2019.

[26] S. Coskun, J. Kim, and H. Toutanji, “Bending, free vibration, and buckling analysis of functionally graded porous micro-plates using a general third-order plate theory,” *Journal of Composites Science*, vol. 3, no. 1, p. 15, 2019.

[27] D. Chen, J. Yang, and S. Kitipornchai, “Elastic buckling and static bending of shear deformable functionally graded porous beam,” *Composite Structures*, vol. 133, pp. 54–61, 2015.

[28] A. S. Rezaei and A. R. Saidi, “Application of Carrera Unified Formulation to study the effect of porosity on natural frequencies of thick porous-cellular plates,” *Composites Part B: Engineering*, vol. 91, pp. 361–370, 2016.

[29] A. S. Rezaei and A. R. Saidi, “Exact solution for free vibration of thick rectangular plates made of porous materials,” *Composite Structures*, vol. 134, pp. 1051–1060, 2015.

[30] J. Zhao, F. Xie, A. Wang, C. Shuai, J. Tang, and Q. Wang, “A unified solution for the vibration analysis of functionally graded porous (FGP) shallow shells with general boundary conditions,” *Composites Part B: Engineering*, vol. 156, pp. 406–424, 2019.

[31] J. Zhao, F. Xie, A. Wang, C. Shuai, J. Tang, and Q. Wang, “Vibration behavior of the functionally graded porous (FGP) doubly-curved panels and shells of revolution by using a semi-analytical method,” *Composites Part B: Engineering*, vol. 157, pp. 219–238, 2019.

[32] Q. Li, D. Wu, X. Chen, L. Liu, Y. Yu, and W. Gao, “Nonlinear vibration and dynamic buckling analyses of sandwich functionally graded porous plate with graphene platelet reinforcement resting on Winkler-Pasternak elastic foundation,” *International Journal of Mechanical Sciences*, vol. 148, pp. 596–610, 2018.

[33] S. Sahmani, M. M. Aghdam, and T. Rabczuk, “Nonlocal strain gradient plate model for nonlinear large-amplitude vibrations of functionally graded porous micro/nano-plates reinforced with GPLs,” *Composite Structures*, vol. 198, pp. 51–62, 2018.

[34] D. Wu, A. Liu, Y. Huang, Y. Huang, Y. Pi, and W. Gao, “Dynamic analysis of functionally graded porous structures through finite element analysis,” *Engineering Structures*, vol. 165, pp. 287–301, 2018.

[35] P. T. Thang, T. Nguyen-Thoi, D. Lee, J. Kang, and J. Lee, “Elastic buckling and free vibration analyses of porous-cellular plates with uniform and non-uniform porosity distributions,” *Aerospace Science and Technology*, vol. 79, pp. 278–287, 2018.

[36] G. R. Liu, T. Nguyen-Thoi, and K. Y. Lam, “An edge-based smoothed finite element method (ES-FEM) for static, free and forced vibration analyses of solids,” *Journal of Sound and Vibration*, vol. 320, no. 4-5, pp. 1100–1130, 2009.

[37] T. Nguyen-Thoi, G. R. Liu, and H. Nguyen-Xuan, “An *n*-sided polygonal edge-based smoothed finite element method (nES-FEM) for solid mechanics,” *International Journal for Numerical Methods in Biomedical Engineering*, vol. 27, no. 9, pp. 1446–1472, 2011.

[38] T. Nguyen-Thoi, G. R. Liu, H. C. Vu-Do, and H. Nguyen-Xuan, “An edge-based smoothed finite element method for visco-elastoplastic analyses of 2D solids using triangular mesh,” *Computational Mechanics*, vol. 45, no. 1, pp. 23–44, 2009.

[39] H. Nguyen-Xuan, G. R. Liu, T. Nguyen-Thoi, and C. Nguyen-Tran, “An edge-based smoothed finite element method (ES-FEM) for analysis of two-dimensional piezoelectric structures,” *Smart Materials and Structures*, vol. 18, p. 12, 2009.

[40] T. N. Thanh, G. R. Liu, H. Nguyen-Xuan, and T. Nguyen-Thoi, “An edge-based smoothed finite element method for primal-dual shakedown analysis of structures,” *International Journal for Numerical Methods in Engineering*, vol. 82, no. 7, pp. 917–938, 2010.

[41] T. Nguyen-Thoi, P. Phung-Van, T. Rabczuk, H. Nguyen-Xuan, and C. Le-Van, “An application of the ES-FEM in solid

domain for dynamic analysis of 2D fluid-solid interaction problems," *International Journal of Computational Methods*, vol. 10, no. 1, Article ID 1340003, 2013.

[42] T. Nguyen-Thoi, P. Phung-Van, V. Ho-Huu, and L. Le-Anh, "An edge-based smoothed finite element method (ES-FEM) for dynamic analysis of 2D Fluid-Solid interaction problems," *KSCE Journal of Civil Engineering*, vol. 19, no. 3, pp. 641–650, 2015.

[43] C. V. Le, H. Nguyen-Xuan, H. Askes, T. Rabczuk, and T. Nguyen-Thoi, "Computation of limit load using edge-based smoothed finite element method and second-order cone programming," *International Journal of Computational Methods*, vol. 10, no. 1, Article ID 1340004, 2013.

[44] H. Nguyen-Xuan, L. V. Tran, T. Nguyen-Thoi, and H. C. Vu-Do, "Analysis of functionally graded plates using an edge-based smoothed finite element method," *Composite Structures*, vol. 93, no. 11, pp. 3019–3039, 2011.

[45] H. H. Phan-Dao, H. Nguyen-Xuan, C. Thai-Hoang, T. Nguyen-Thoi, and T. Rabczuk, "An edge-based smoothed finite element method for analysis of laminated composite plates," *International Journal of Computational Methods*, vol. 10, no. 1, Article ID 1340005, 2013.

[46] P.-S. Lee and K.-J. Bathe, "Development of MITC isotropic triangular shell finite elements," *Computers & Structures*, vol. 82, no. 11-12, pp. 945–962, 2004.

[47] T. Chau-Dinh, Q. Nguyen-Duy, and H. Nguyen-Xuan, "Improvement on MITC3 plate finite element using edge-based strain smoothing enhancement for plate analysis," *Acta Mechanica*, vol. 228, no. 6, pp. 2141–2163, 2017.

[48] T.-K. Nguyen, V.-H. Nguyen, T. Chau-Dinh, T. P. Vo, and H. Nguyen-Xuan, "Static and vibration analysis of isotropic and functionally graded sandwich plates using an edge-based MITC3 finite elements," *Composites Part B: Engineering*, vol. 107, pp. 162–173, 2016.

[49] Q.-H. Pham, T.-V. Tran, T.-D. Pham, and D.-H. Phan, "An edge-based smoothed MITC3 (ES-MITC3) shell finite element in laminated composite shell structures analysis," *International Journal of Computational Methods*, vol. 15, no. 7, Article ID 1850060, 2018.

[50] Q.-H. Pham, T.-D. Pham, V. T. Quoc, and D.-H. Phan, "Geometrically nonlinear analysis of functionally graded shells using an edge-based smoothed MITC3 (ES-MITC3) finite elements," *Engineering with Computers*, vol. 33, pp. 1–14, 2019.

[51] D. Pham-Tien, H. Pham-Quoc, T. Vu-Khac, and N. Nguyen-Van, "Transient analysis of laminated composite shells using an edge-based smoothed finite element method," in *Proceedings of the International Conference on Advances in Computational Mechanics 2017*, pp. 1075–1094, Springer, Berlin, Germany, 2018.

[52] K.-U. Bletzinger, M. Bischoff, and E. Ramm, "A unified approach for shear-locking-free triangular and rectangular shell finite elements," *Computers & Structures*, vol. 75, no. 3, pp. 321–334, 2000.

[53] T. Nguyen-Thoi, P. Phung-Van, H. Nguyen-Xuan, and C. Thai-Hoang, "A cell-based smoothed discrete shear gap method using triangular elements for static and free vibration analyses of Reissner-Mindlin plates," *International Journal for Numerical Methods in Engineering*, vol. 91, no. 7, pp. 705–741, 2012.

[54] K.-J. Bathe and E. N. Dvorkin, "A formulation of general shell elements-the use of mixed interpolation of tensorial components," *International Journal for Numerical Methods in Engineering*, vol. 22, no. 3, pp. 697–722, 1986.

Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 9232561, 10 pages
https://doi.org/10.1155/2020/9232561

Research Article

PTS-FNN-Based Health Prediction Method for Flexible Photoelectric Film Material Processing Equipment

Yaohua Deng,[1] **Kexing Yao**,[1] **Tuo Jin**,[2] **Zhaoxi Feng**,[1] **and Xiali Liu**[1]

[1]*School of Electro-mechanical Engineering, Guangdong University of Technology, Guangzhou, Guangdong, China*
[2]*Zhejiang Tobacco Company Wenzhou Company, Wenzhou, Zhejiang, China*

Correspondence should be addressed to Kexing Yao; 2111701138@mail2.gdut.edu.cn

Received 9 December 2019; Accepted 16 March 2020; Published 23 April 2020

Academic Editor: Stylianos Georgantzinos

Roll-to-Roll (R2R) processing is a common processing method for flexible photoelectric film materials. Due to the physical properties of the materials, the change in the performance of the R2R processing equipment can easily cause deformation of the flexible film material, it is particularly important to predict the performance degradation of the processing equipment. Based on the accuracy and real-time requirements of performance degradation prediction, a PTS-FNN model for performance degradation prediction was proposed in this paper, which combines the Possibilistic C-Means (PCM) fuzzy clustering and Takagi–Sugeno Fuzzy Neural Network (TS-FNN). We also studied the PCM classification algorithm of input data of PTS-FNN model, the predecessor network of TS-FNN prediction model and the construction method of post-component network. Finally, the implementation process of PCM classification algorithm and TS-FNN prediction model were given. The R2R processing equipment health prediction experiment system was built and the PTS-FNN model experiment was carried out. The experimental results showed that the training time of PTS-FNN model was 50.37% less than the standard TS-FNN prediction model. The prediction accuracy increased by 5.48%, and the PTS-FNN had no error in the judgment of state 1 and state 4.

1. Introduction

Currently, Roll-to-Roll (R2R) processing technique is the most widespread processing method for a series of flexible thin film materials internationally, for it can maintain the enhancement its productivity while automatizing the mechanical equipment to the greatest extent. R2R processing is a common processing method for flexible photoelectric thin film materials, and the performance deterioration of the R2R processing device is the primal problem the mass manufacturing of flexible photoelectric thin film materials faces. In recent years, the technology of equipment health prognosis has become a research hotspot [1, 2]. In 2017, Lee et al. [3] proposed a prognosis algorithm that boosts material removal rate (MRR) based on integrated models and data-driven approach. The method proposed combines the influences of physical mechanism and nearest neighbors, extracts the relating characteristics. These characteristics are inputted to build multiple regression models which will be integrated so as to obtain the final prognosis. Rapid development and extensive applications in deep learning have been achieved in the last few years. In the field of industry, the researcher applies deep learning to the analysis of industrial data. University of Michigan's Ni Xia [4] developed an operating load based real-time rolling Grey forecasting technique, so as to provide efficient accurate machine health prognosis, as well as analyzed the influencing factors. Deng et al. [5] (2018) combined the quality control chart and SoV, proposing a fault diagnosis approach of the flexible material R2R manufacturing system. Based on the relativity between the source of fault and product quality during manufacturing as well as statistical distribution pattern of the feature vector of processing quality and the source of fault, this approach utilized SoV model under controlled or uncontrolled state and mathematical model of probability distribution to deduce the statistical measurement of quality characteristic variables. The consequent quality control chart can pinpoint the fault location in the system quickly and easily. Zhao and Ouyang [6] (2016) applied Multi-Agent Genetic Algorithms (MAGA) to the prognosis of the state of avionic devices, which overcomes the flaw of the Baum–Welch algorithm easily falling into locally optimal solutions, greatly enhancing the prognosis accuracy, speed, and stability.

In application, the health prognosis model for R2R processing device has advanced instantaneity requirement; therefore, it is particularly vital to study how to enhance the instantaneity of the prognosis model. Because the Fuzzy C-Means (FCM) clustering algorithm is based on fuzzy theory to describe the uncertainty of sample generics, the fuzzy membership value of each classification point is obtained by optimizing the objective function. Although FCM is widely used in clustering algorithms, it has the problem of being sensitive to noise data. The PCM clustering algorithm can make up for the shortcomings of the above-mentioned FCM algorithm by relaxing the constraint of membership degree without normalizing the data. Because the neural network modeling has the characteristics to deal with the nonlinear approximation problem well, the PCM clustering method can well divide the data with irregular boundaries. For that reasons, the following content will be binding the PCM clustering analyzing method and the Takagi–Sugeno Fuzzy Neural Network (TS-FNN) to discuss the health state prognosis modeling of the R2R processing device.

2. Neural Network Model for Health Prediction of Roll-To-Roll Processing Equipment

The standard TS-FNN [7] was first proposed by Japanese scholars Takagi and Sugeno; it is a specific method for identifying fuzzy models of nonlinear dynamic systems. TS-FNN can be divided into antecedent networks and consequent network according to different structural functions (Figure 1). The antecedent network is used to match the fuzzy rules, that is, to calculate the adaptability of each fuzzy rule. The antecedent network is composed of four layers of neural networks: the first layer is the input layer, the second layer is the membership layer (Gaussian function), the third layer is to calculate the adaptability ($a_j = \min\{u_{11}, \ldots u_{1j}, \ldots u_{nj}\}$, a_j is the $j^{\text{-th}}$ fuzzy rule adaptability, n represents the number of inputs), the fourth layer is to perform normalized processing, the adaptability is normalized ($\overline{a_j} = (a_j / \sum_{b=1}^{m} a_b)$, m represents the number of nodes in the fourth layer of antecedent network), and $\{\overline{a_1}, \overline{a_2}, \ldots \overline{a_m}\}$ is the output to the antecedent network. The consequent network is to generate fuzzy rules, it consists of three layers of networks. The first layer is the input layer, and the second layer is the rule computing layer ($y_{gj} = \sum_{i=0}^{n} p_{ji}^{r} x_i$, $j = 1, 2, \ldots m$; $g = 1, 2, \ldots r$; p_{ji}^{r} is the linear parameter of the consequent network). The third layer is the system output layer. The system output $\{\underline{y_1}, \underline{y_2}, \ldots \underline{y_r}\}$ is obtained by adding the weights of $\{\overline{a_1}, \overline{a_2}, \ldots \overline{a_m}\}$ as the connection weights, where $y_r = \sum_{j=1}^{m} \overline{a_j} y_{rj}$, $y_r \in Y$, y_{rj} is the output of the $j^{\text{-th}}$ fuzzy rule.

The standard T-S fuzzy neural network model requires to linearize the input data, which will make the input space and fuzzy rules difficult to optimize and extract, and this will lead to more problems, for instance, the system structure should be planned in advance. Therefore, when dealing with the multi-input nonlinear space, the PCM fuzzy clustering is used as the antecedent model extraction method, to realize the expected purpose of improving the training efficiency and recognition effect of the model. The PCM algorithm can adjust the clustering center, radius, and number of clusters of the input space; rationally divide the ambiguity of the input data; and determine the membership function of the data points and the rule adaptability; therefore, this PCM algorithm achieves basic elimination of the low quality samples to participate in the antecedent networks calculation, and model training speed and accuracy can be greatly improved. According to the above analysis and reference [8], the performance regression evaluation model PTS-FNN of R2R processing equipment based on PCM and TS-FNN shown in Figure 2 is proposed. The model combines the advantages of the PCM method, the T-S fuzzy inference model, and the fuzzy neural network modeling method. The PTS-FNN model includes an antecedent network and a consequent network. The antecedent network matches the standard T-S fuzzy rules, and the antecedent network concludes the fuzzy rules. After the weighted calculation of the antecedent and consequent network, the PCA parameter of each station roller axis performance $X = \{x_1, x_2, \ldots x_n\}$ and the nonlinear modeling of equipment health status $S = \{s_1, s_2, \ldots s_r\}$ are established [9].

3. PCM Classification Method for Input Data of Health State Prediction Model of Roll-To-Roll Processing Equipment

Instead of normalizing the input data, Possibilistic C-Means Clustering Algorithm relaxes the constraints of the membership function during the division of data, which is an advantage of Possibilistic C-Means Clustering Algorithm, thus enhancing the sensitivity to noisy data [10, 11].

Here are the assumptions that x_h of the sample set $X = \{x_1, x_2, x_3, \ldots, x_h, \ldots x_n\}$ possesses p amount of characteristics $x_h = \{x_{h1}, x_{h2}, x_{h3}, \ldots, x_{hp}\}$, the sample set X possesses c amount of clustering center $V = \{v_1, v_2, v_3, \ldots, v_c\}$, and using d_{ik} to represent the Euclidean distance of sample x_k and clustering center v_i, gives

$$d_{ik} = \|x_k - v_{\text{i}}\| = \sqrt{\sum_{j=1}^{p} \left(x_{kj} - v_{ij}\right)^2}. \tag{1}$$

In equation (1), $i = 1, 2, \cdots, c$; $j = 1, 2, \cdots, p$; $k = 1, 2, \cdots, n$. Then the target function of PCM:

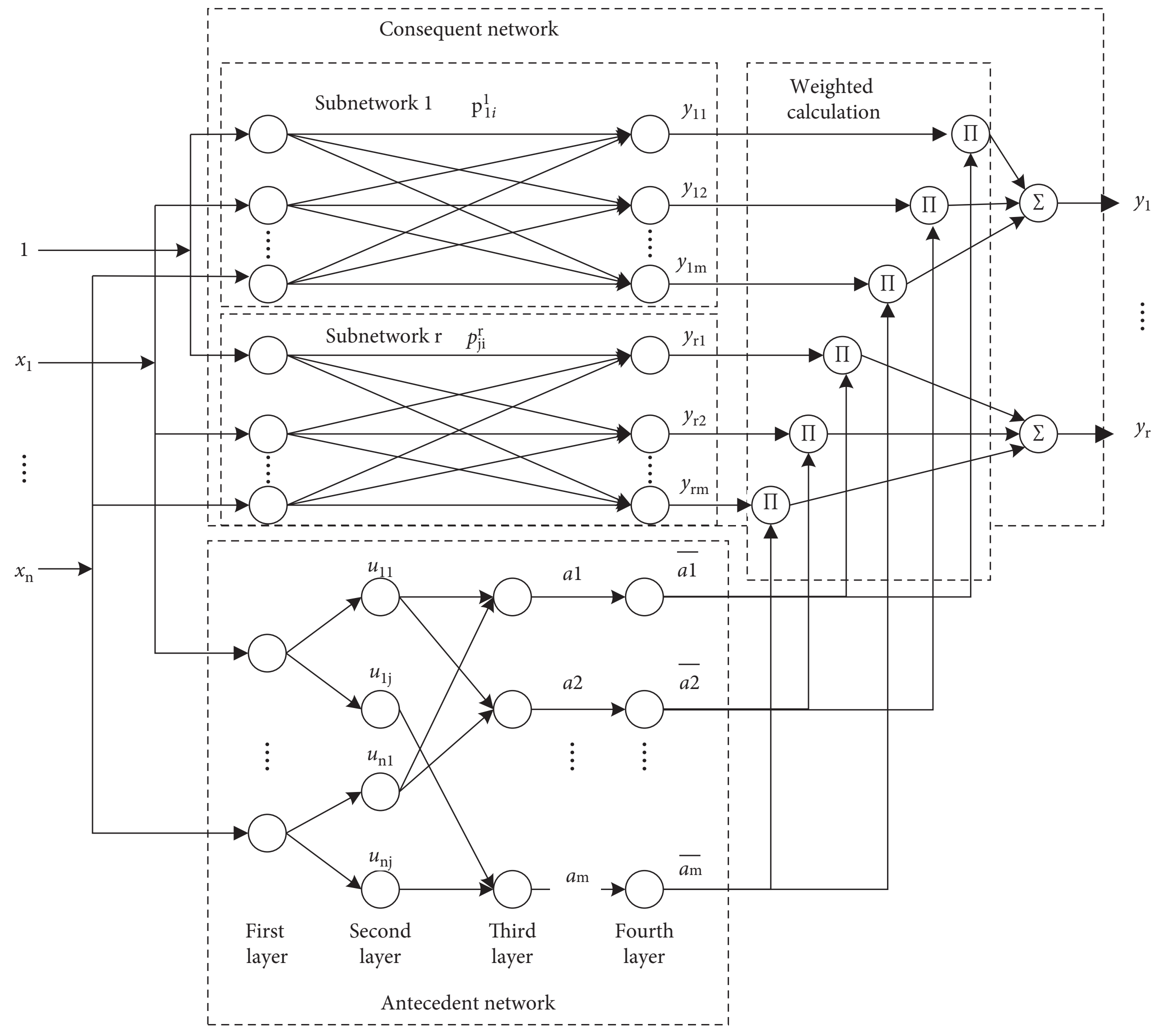


Figure 1: T-S fuzzy neural network model.

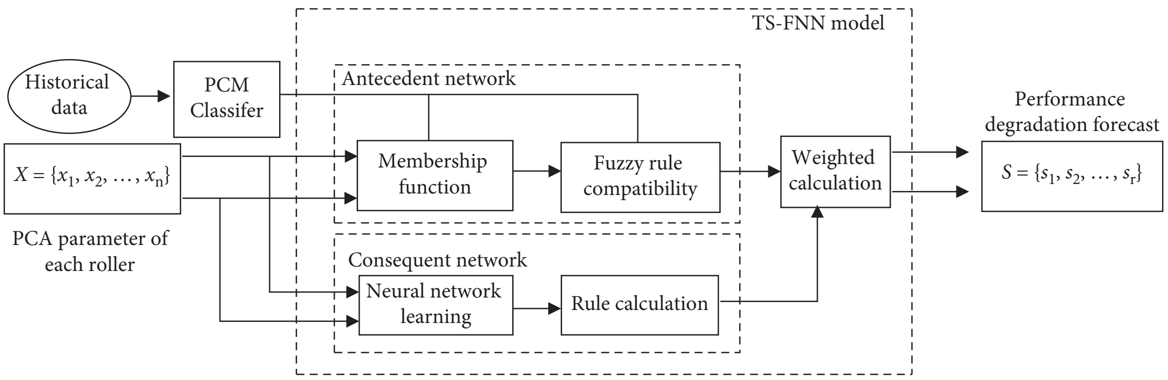


Figure 2: PTS-FNN-based R2R processing equipment health prediction model framework.

$$\min J_{\mathrm{PCM}}(X,U,V) = \sum_{i=1}^{c}\sum_{k=1}^{n}(u_{ik})^{\alpha} d_{ik}^2 + \sum_{i=1}^{c}\eta_i\sum_{k=1}^{n}(1-u_{ik})^{\alpha}. \tag{2}$$

In equation (2) $U = [u_{ik}]_{c\times n}$ represents the possibility partition matrix; α represents the weighting factor of the clustering's ambiguity and $\alpha \in [1,\infty]$(normally $1.5 \le \alpha \le 2.5$); u_{ik} represents membership under category No. i sample No. k ($u_{ik} \in [0,1]$); η_i represents the penalty parameter, its expression is shown as equation (3) [12].

$$\eta_i = K\frac{\sum_{k=1}^{n}(u_{ik})^{\alpha} d_{ik}^2}{\sum_{k=1}^{n}(u_{ik})^{\alpha}}. \tag{3}$$

In order to avoid null solution in the target function, introducing the regularization term $f(U)$ of the possibility partition matrix (shown in equation (4)), provides a larger penalty value when $u_{ik} \longrightarrow 0^+$.

$$f(U) = \sum_{i=1}^{c} \eta_i \sum_{k=1}^{n} (1 - u_{ik})^{\alpha}. \tag{4}$$

As mentioned in the previous chapters and sections, for the sake of dividing the fuzzy level of the input data of the health prognosis artificial neural network model for R2R processing device precisely, the front end of the PTS-FNN model is classified by PCM first. Combining equations (1)–(4), the calculating formulas of the possibility partition matrix U and the clustering center V is shown as equations (5) and (6), respectively.

$$u_{ik} = \left[1 + \left(\frac{d_{ik}^2}{\eta_i}\right)^{(1/\alpha - 1)}\right]^{-1}, \tag{5}$$

$$v_i = \frac{\sum_{k=1}^{n} (u_{ik})^{\alpha} x_k}{\sum_{k=1}^{n} (u_{ik})^{\alpha}}. \tag{6}$$

Figure 3 provides the process of the PCM algorithm:

① Set clustering category to c, set the fuzzy weight to α (normally $\alpha = 2$), set the iteration thresholding to q (normally between 0.001 and 0.01), set the iteration count to t, initialize the possibility partition matrix $U^{(s)}$ and the clustering prototype $V^{(s)}$ $(s = 0)$

② Estimate η_i base on $U^{(s)}$ and $V^{(s)}$ as well as equation (3)

③ Update the division of the matrix $U^{(s+1)}$ and the clustering prototype $V^{(s+1)}$ in accordance with equations (5) and (6)

④ Reestimate η_i, repeat ③

⑤ Determining threshold: according to the given threshold q, if $\|V^{(s+1)} - V^{(s)}\| \leq q$, stop the iteration; otherwise, $t = t + 1$ and Jump to ②

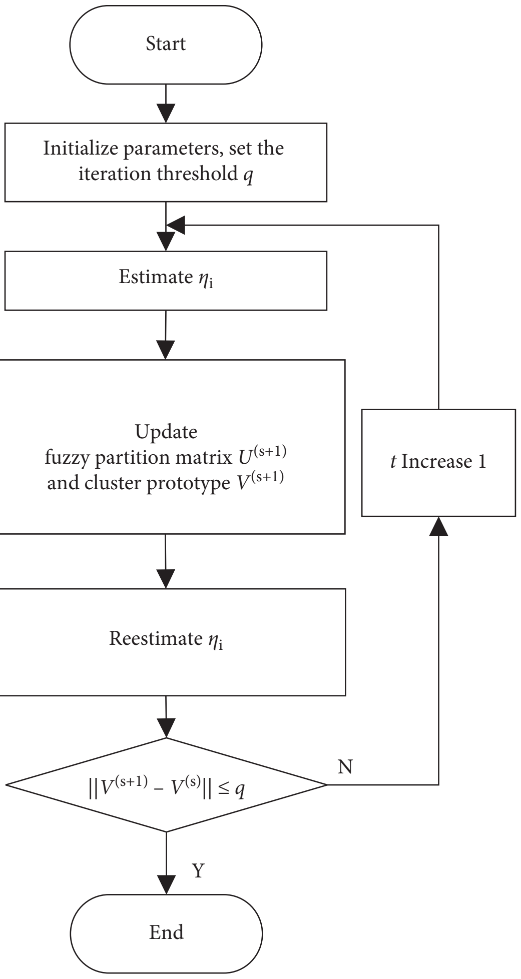

Start
Initialize parameters, set the iteration threshold q
Estimate η_i
Update fuzzy partition matrix $U^{(s+1)}$ and cluster prototype $V^{(s+1)}$
Reestimate η_i
$\|V^{(s+1)} - V^{(s)}\| \leq q$
N
Y
t Increase 1
End

Figure 3: PCM algorithm calculation process.

4. Construction of TS-FNN Model for Health State Prediction of Roll-To-Roll Processing Equipment

Figure 2 shows that the health status prognosis TS-FNN model for R2R processing device includes determining the structural parameter of the antecedent network and the consequent network. The following content focus on introducing the deduction of the antecedent network's the membership function and the rule compatibility, as well as the weight between each node of the consequent network can be obtained using the Error Back Propagation Algorithm [8].

Here are the assumptions that matrix U = $[u_{ik}]$ is a fuzzy matrix after PCM data division, G_i $(1 \leq i \leq c)$ represents fuzzy category group of c after division, β represents clustering fuzzy degree weighted index and the category center v_{iq} and the corresponding variance σ_{iq}^2 of G_i can be expressed as [7, 8]

$$v_{iq} = \frac{\sum_{k=1}^{n} (u_{ik})^{\beta} q_k}{\sum_{k=1}^{n} (u_{ik})^{\beta}},$$
$$\sigma_{iq}^2 = \frac{\sum_{k=1}^{n} (u_{ik})^{\beta} (q_k - v_{iq})^2}{\sum_{k=1}^{n} (u_{ik})^{\beta}}, \tag{7}$$
$$i = 1, 2 \ldots c,$$

Considering the critical demand of the fuzzy clustering after dividing the input data space, the category data q_k of G_i and the corresponding variance σ_{iq}^2 of the category component v_{in+1}, $i = 1, 2, \ldots c$ under the clustering center $V_i = [v_{i1}, v_{i2} \ldots, v_{in+1}]^T$ approximate 0, thus using the minimal value (namely the shortest distance to the category center v_{iq}) of the category amount q_k as the decision function $d_F(G_i)$ of the fuzzy clustering $d_F(G_i)$,

$$d_F(G_i) = \left\{q_k \mid \min\left(\left|q_k - v_{iq}\right|\right)\right\}, \quad k = 1, 2 \ldots l, i = 1, 2 \ldots c. \tag{8}$$

Correspondingly, the membership function $\mathrm{Gu}(x_k)$ of G_i is expressed as

$$\mathrm{Gu}_{ji}(x_{kj}) = \mathrm{Exp}\left(-\frac{|x_{kj}-v_{iq}|}{|v_{iq}-v_{iq}^{'}|}\times\gamma\right), \quad i=1,2,\ldots,c;\ j=1,2,\ldots,n. \tag{9}$$

In the equation $|v_{iq}-v_{jq}^{'}|$ represents the width of the input data's division area, v_{iq} represents the nearest cluster center value of cluster center No. i, factor γ sets the rate of the membership decrease when the input samples fleeing the cluster center, normally $2\le\gamma\le 4$.

The membership function $\mathrm{Gu}_{ji}(x_{kj})$ represents the similarity between the input sample x_k and a certain fuzzy class G_i. There is a relationship between the two: when x_k is far away from the prototype, then $\mathrm{Gu}_{ji}(x_{kj})$ is close to 0; when x_k is close to the prototype, then $\mathrm{Gu}_{ji}(x_{kj})$ is close to 1.

R_i represents a decision rule of certain clustering category G_i ($R_i:\ if\ x_k\in G_i,\ \text{then}\ \mathrm{d}(x_k)=\mathrm{d}_F(G_i),\ i=1,2\ldots c$), when the multidimensional fuzzy set G_i projected onto the entire input data space, the decision rule R_i' can be expressed as

$$\begin{aligned}R_i':\ &If\ x_1\in G_{i1}\ \text{and}\ x_2\in G_{i2}\ \text{and}\ \ldots\ \text{and}\ x_n\in G_{in},\\ &\text{Then}\ \mathrm{d}(x_k)=\mathrm{d}_F(G_i),\ i=1,2\ldots c,\end{aligned} \tag{10}$$

The compatibility $\mathrm{G}\alpha_i$ of the corresponding x_k for rule R_i' is the product of each component's membership $\mathrm{Gu}_{ji}(x_{kj})$, $j=1,2..n$:

$$G\alpha_i = \prod_{j=1}^{n}\mathrm{Gu}_{ji}(x_{kj}),\ i=1,2\ldots c. \tag{11}$$

As shown above, the membership function of the model is calculated by equation (9), the rule compatibility is calculated by equation (11).

σ_{iq}^2 manifests the quality of fuzzy division. The larger σ_{iq}^2 is, the worse quality the division is; thus, enter $E\sigma$ as the quality index of the fuzzy division. If $\sigma_{iq}^2 > E\sigma$, $c=c+1$, a second fuzzy division is warranted. The relevant fuzzy clustering will not be extracted as the TS-FNN antecedent network until all the categories meet the condition $\sigma_{iq}^2\le E\sigma$ [8].

After the building of the antecedent network is completed, estimates of the consequent network's linear parameter p_{ji}^r are required. In this paper, the linear least squares recursive estimation method is used to estimate the parameters of the postnetwork. In this part of the training learning phase, the parameters of the antecedent network need to be fixed first.

Constructing error cost function is as

$$E=\frac{1}{2}(y_r'-y_r)^2, \tag{12}$$

where y_r' represents the expected output, y_r represents the actual output.

Optimized estimate algorithm of p_{ji}^r.

$$\begin{aligned}\frac{\partial E}{\partial p_{ji}^r} &= \frac{\partial E}{\partial y_r}\frac{\partial y_r}{\partial y_{gj}}\frac{\partial y_{gj}}{\partial p_{ji}^r} = -\overline{a_j}(y_r'-y_r)x_i,\\ p_{ji}^r(j+1) &= p_{ji}^r(j) - f\frac{\partial E}{\partial p_{ji}^r} = p_{ji}^r(j)+f\overline{a_j}(y_r'-y_r)x_i.\end{aligned} \tag{13}$$

Among them, a_j is the j-th fuzzy rule adaptability, $\overline{a_j}=(a_j/\sum_{b=1}^{m}a_b)$; $f\in(0,1)$, f is algorithm learning rate; $g=1,2,\ldots,r$; $j=1,2,\ldots,m$.

5. Model Verification Experiment and Test

5.1. Construction of the Health Status Prognosis Model. In order to verify the efficiency of the health prognosis PTS-FNN model for R2R processing device this paper proposed, the R2R continuous manufacturing device shown in Figure 4 is introduced as the experiment platform, its principle of operation is shown as Figure 5. Use the PCA parameter x_1, x_2, x_3, and x_4 of the roller performance corresponding to workstation 1, 2, 3, and 5 of the R2R manufacturing system as the input of the PTS-FNN model. Divide the thin film processing quality and the health status of the device in accordance with Table 1 and quantify to obtain the health status indicator function y={1, 2, 3, 4} of the device as the output of PTS-FNN.

As shown in Figure 4, the R2R flexible material processing system adopted is mainly composed of a mechanical structure and drive control unit. For the mechanical structure, standard enforced aluminum alloy extrusions was selected as the supporting frame, including the unwinding module, the drive module, the wind-in module, etc. The experiment chose blue PET polyester film with thickness of 0.05 mm, width of 50 mm, elastic modulus of 3495 MPa, density of 1450 kg/m^3, and Poisson's ratio of 0.3. The vibration data is collected by the AWA5936 vibration measuring instrument. The AWA5936 vibration measuring instrument has good repeatability and can ensure the accuracy of the measurement. The built-in acceleration sensor sensitivity is 1pC/m/$s^2\pm 5\%$, the measurement range of acceleration is 0.03 m/s^2 to 1000 m/s^2, the piezoelectric sensor frequency response range is 10 Hz to 8 kHz, the measurement error is 5% to 10% (LO : 10 Hz to 1kH; HI: 1 kHz to 8 kHz), and the sampling frequency is 48 kHz.

Principle of operation is shown as Figure 5. Based on different functions of each roll shaft, the R2R processing system is divided into different workstations. As mentioned in the previous content, the vibration workstation 1, 2, 3, 4, and 5 is measured, respectively, and the data is collected and documented every 30 minutes.

Roll shafts of four workstations used to evaluate the performance deterioration of the R2R device are of 4 different stages: excellent condition, prerecession, developing recession, and near expiry. Roll shaft of each stage demonstrates different vibration signals. The vibration signals of workstation1's unwinding roller at different performance stages is shown in Figure 6 (g = 9.8 m/s^2).

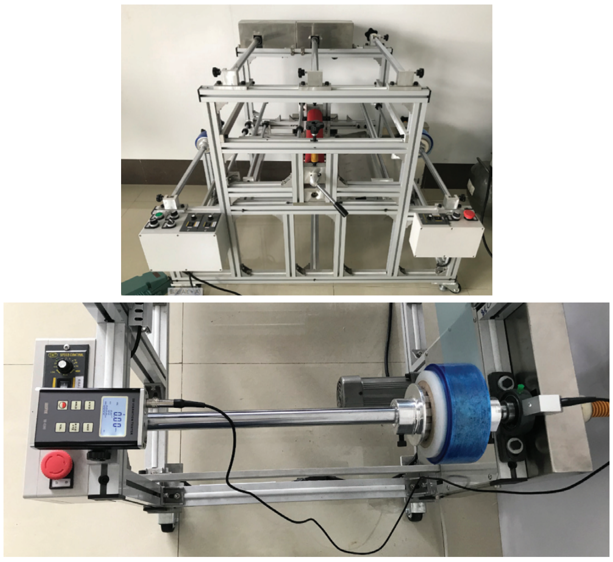


Figure 4: R2R continuous processing equipment.

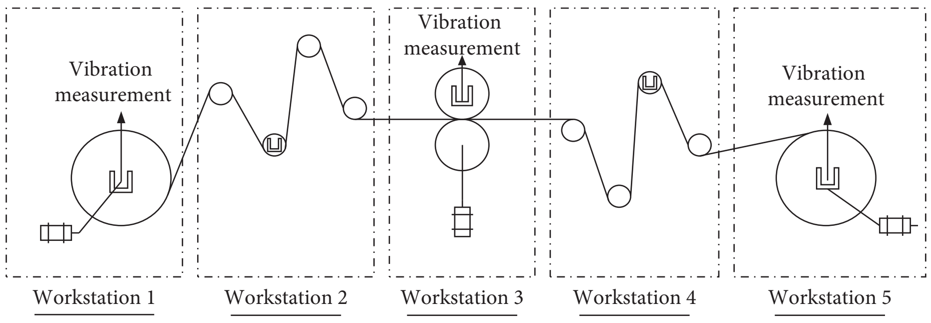


Figure 5: R2R system working principle diagram.

Table 1: Film processing quality and equipment health status.

Serial number	Film processing quality	Equipment health status	Device health status quantified value
1	The number of pleats is 0 and the maximum offset of the winding ≤ 1 mm	Excellent condition	1
2	The number of pleats is 1 and the maximum offset of the winding ≤ 1.5 mm	Prerecession	2
3	The number of pleats is 1–3 and the maximum offset of the winding ≤ 2 mm	Developing recession	3
4	More than 3 pleats or maximum offset > 2mm	Near expiry	4

Collect 20 sets of data from each stage of each of the 4 workstations' roll shafts, 320 in total, as the experiment data of the performance deterioration prognosis PTS-FNN model for the R2R processing device. Based on the building method mentioned in Section 3, make $c = 2$, $E\sigma = 1.25$, and start the fuzzy division of the input data. When $c = 5$ and all variance of the fuzzy division categories meet the condition of $\sigma_{iq}^2 \le E\sigma$, the fuzzy division of the input data ends. Table 2 shows the 5 category centers after fuzzy division, which is expressed as V_i, $i = 1, 2, \ldots, 5$, and the according width of divides data regions R_{iq} ($R_{iq} = |v_{iq} - v_{iq}^{'}|$, v_{iq} and $v_{iq}^{'}$ indicates the category component of V_i, v_{iq} is the clustering center value nearest category center No. i).

After obtaining each of the category center and the width of the divided regions, according to equation (9), make $\gamma = 2$, and the membership functions of input $(x)_1 \sim (x)_4$ are shown in Table 3.

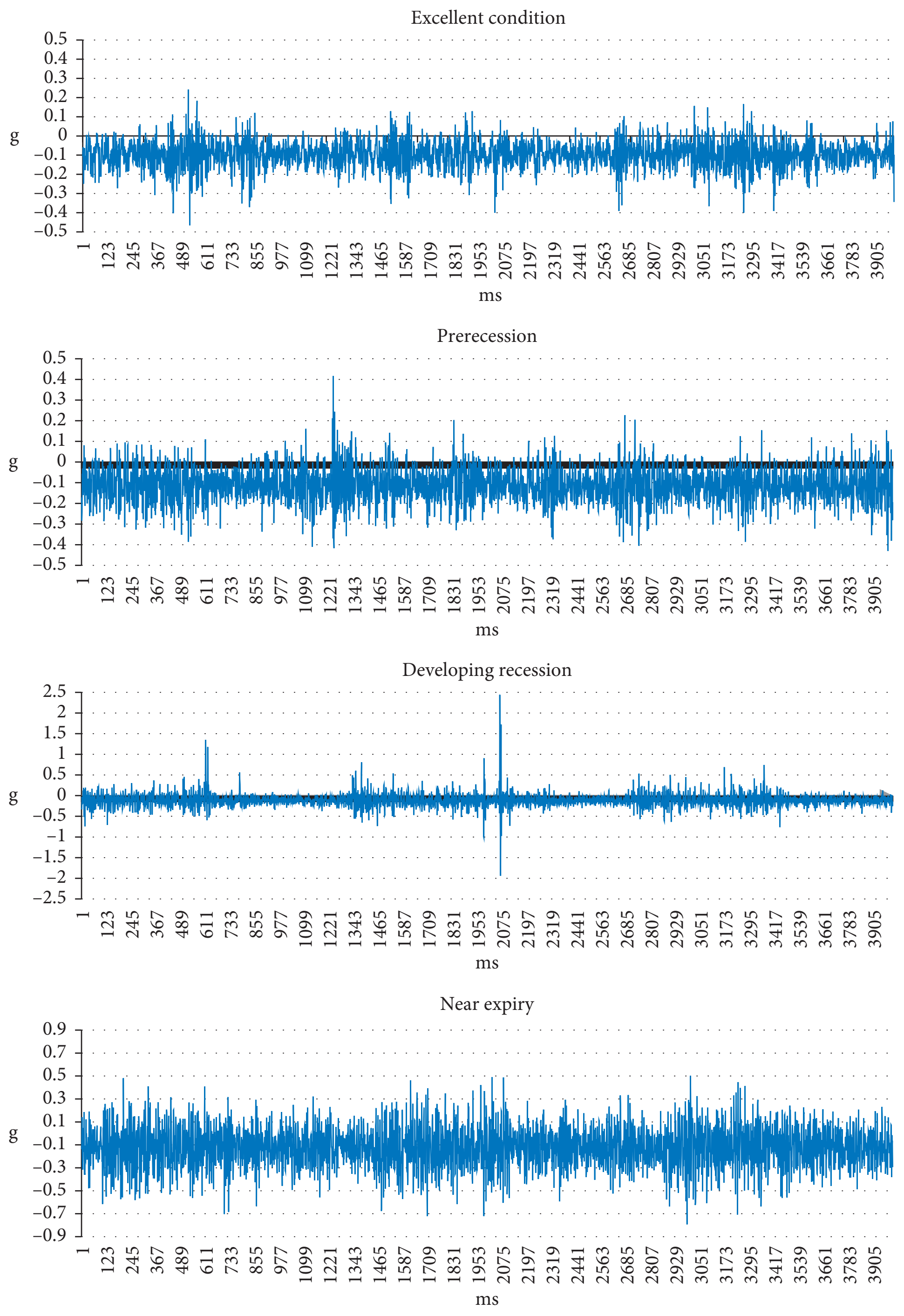


Figure 6: Vibration signal diagram of different performance conditions of the unwinding roller.

Table 2: Center coordinates of each category、the width of the division area.

G_i	Center position coordinates				The width of the division area			
	v_{ix_1}	v_{ix_2}	v_{ix3}	v_{ix_4}	R_{ix_1}	R_{ix_2}	R_{ix3}	R_{ix_4}
G_1	2.0863	2.5407	0.0824	1.7858	1.0760	0.2052	0.1001	0.1263
G_2	1.4892	2.2923	0.0823	1.3848	1.0041	0.7030	0.0010	0.1213
G_3	2.8620	1.5765	0.0912	2.0093	0.7403	0.1750	0.5743	0.1261
G_4	1.8905	1.3779	0.0634	1.1342	1.5080	0.0003	0.1050	0.1264
G_5	2.7644	3.0037	0.0735	1.6871	1.5760	0.0510	0.3847	0.1175

$$
\begin{aligned}
G\alpha_1 &= Gu_{11}(x_1)\times Gu_{21}(x_2)Gu_{31}(x_3)Gu_{41}(x_4) = \exp\left(-\frac{|x_1-2.0863|}{|1.0760|}\times 2\right)\times \exp\left(-\frac{|x_2-2.5407|}{|0.2052|}\times 2\right)\\
&\quad\times \exp\left(-\frac{|x_3-0.0824|}{|0.1001|}\times 2\right)\times \exp\left(-\frac{|x_4-1.7858|}{|0.1263|}\times 2\right),\\
G\alpha_2 &= Gu_{12}(x_1)\times Gu_{22}(x_2)Gu_{32}(x_3)Gu_{42}(x_4) = \exp\left(-\frac{|x_1-1.4892|}{|1.0041|}\times 2\right)\times \exp\left(-\frac{|x_2-2.2923|}{|0.7030|}\times 2\right)\\
&\quad\times \exp\left(-\frac{|x_3-0.0823|}{|0.0010|}\times 2\right)\times \exp\left(-\frac{|x_4-1.3848|}{|0.1213|}\times 2\right),\\
G\alpha_3 &= Gu_{13}(x_1)\times Gu_{23}(x_2)Gu_{33}(x_3)Gu_{43}(x_4) = \exp\left(-\frac{|x_1-2.8620|}{|0.7403|}\times 2\right)\times \exp\left(-\frac{|x_2-1.5765|}{|0.1750|}\times 2\right)\\
&\quad\times \exp\left(-\frac{|x_3-0.0912|}{|0.5743|}\times 2\right)\times \exp\left(-\frac{|x_4-2.0093|}{|0.1261|}\times 2\right),\\
G\alpha_4 &= Gu_{14}(x_1)\times Gu_{24}(x_2)Gu_{34}(x_3)Gu_{44}(x_4) = \exp\left(-\frac{|x_1-1.8905|}{|1.5080|}\times 2\right)\times \exp\left(-\frac{|x_2-1.3779|}{|0.0003|}\times 2\right)\\
&\quad\times \exp\left(-\frac{|x_3-0.0634|}{|0.1050|}\times 2\right)\times \exp\left(-\frac{|x_4-1.1342|}{|0.1246|}\times 2\right),\\
G\alpha_5 &= Gu_{15}(x_1)\times Gu_{25}(x_2)Gu_{35}(x_3)Gu_{45}(x_4) = \exp\left(-\frac{|x_1-2.7644|}{|1.5760|}\times 2\right)\times \exp\left(-\frac{|x_2-3.0037|}{|0.0510|}\times 2\right)\\
&\quad\times \exp\left(-\frac{|x_3-0.0735|}{|0.3847|}\times 2\right)\times \exp\left(-\frac{|x_4-1.6871|}{|0.1175|}\times 2\right).
\end{aligned}
\tag{14}
$$

Set neural nodes of the TS-FNN's antecedent network and consequent network for each layer referring to paper [6, 7]; see Table 4 for details.

Select 240 sets of the 320 sample sets to train the consequent network, the remaining 80 sets of samples are used to inspect the accuracy of the model. Set the initial learning efficiency to 0.55 which decreases successively by 0.05, the minimal expected learning error to 0.005, the amount of training steps to 500. Then the linear parameter of the consequent network p_{ji}^{r} is calculated (shown in Table 5).

5.2. Model Validation. In order to verify the validity of the PTS-FNN model proposed, select 240 sets of the 320 sample sets to build the PTS-FNN model and the standard TS-FNN [7] prognosis model, respectively, and repeat training both of the models 20 times each to establish the average time of model building and training. Training time of the former is 47.85 seconds while the latter is 96.43 seconds, which means the former model reduces the training time by 50.37% relatively.

The remaining 80 sets of samples are used to inspect the accuracy of the model. Table 6 shows a part of samples and testing data; the comparison of the accuracy of both models is shown in Table 7.

Comparing the two prediction models of standard TS-FNN and PTS-FNN neural network, it can be concluded that in the training time, the latter is shorter than the training time of the former, which improves the convergence speed of the neural network and reduces the training time; In terms of accuracy, the prediction accuracy of the standard TS-FNN is 91.25%, and the prediction accuracy of the PTS-FNN neural network is 96.25%. The latter is more accurate than the former, and there is no error in the judgment of state 1 and state 4.

6. Conclusion

Based on the analysis of the health prediction status of flexible photoelectric film processing equipment, this paper proposed a PTS-FNN for flexible material processing equipment health prediction combined with PCM and TS-FNN. The model combined the advantages of PCM method,

TABLE 3: Membership function of input quantity $(x)_1$~$(x)_4$. And according to equation (11) and Table 3, $G\alpha_i$ can be expressed as the equation given in the table.

Input quantity x_1,x_2		Input quantity x_3,x_4	
Serial number	Membership function	Serial number	Membership function
1	$\mathrm{Gu}_{11}(x_1) = \exp(-(\lvert x_1 - 2.0863\rvert/\lvert 1.0760\rvert) \times 2)$	11	$\mathrm{Gu}_{31}(x_3) = \exp(-(\lvert x_3 - 0.0824\rvert/\lvert 0.1001\rvert) \times 2)$
2	$\mathrm{Gu}_{12}(x_1) = \exp(-(\lvert x_1 - 1.4892\rvert/\lvert 1.0041\rvert) \times 2)$	12	$\mathrm{Gu}_{32}(x_3) = \exp(-(\lvert x_3 - 0.0823\rvert/\lvert 0.0010\rvert) \times 2)$
3	$\mathrm{Gu}_{13}(x_1) = \exp(-(\lvert x_1 - 2.8620\rvert/\lvert 0.7403\rvert) \times 2)$	13	$\mathrm{Gu}_{33}(x_3) = \exp(-(\lvert x_3 - 0.0912\rvert/\lvert 0.5743\rvert) \times 2)$
4	$\mathrm{Gu}_{14}(x_1) = \exp(-(\lvert x_1 - 1.8905\rvert/\lvert 1.5080\rvert) \times 2)$	14	$\mathrm{Gu}_{34}(x_3) = \exp(-(\lvert x_3 - 0.0634\rvert/\lvert 0.1050\rvert) \times 2)$
5	$\mathrm{Gu}_{15}(x_1) = \exp(-(\lvert x_1 - 2.7644\rvert/\lvert 1.5760\rvert) \times 2)$	15	$\mathrm{Gu}_{35}(x_3) = \exp(-(\lvert x_3 - 0.0735\rvert/\lvert 0.3847\rvert) \times 2)$
6	$\mathrm{Gu}_{21}(x_2) = \exp(-(\lvert x_2 - 2.5407\rvert/\lvert 0.2052\rvert) \times 2)$	16	$\mathrm{Gu}_{41}(x_4) = \exp(-(\lvert x_4 - 1.7858\rvert/\lvert 0.1263\rvert) \times 2)$
7	$\mathrm{Gu}_{22}(x_2) = \exp(-(\lvert x_2 - 2.2923\rvert/\lvert 0.7030\rvert) \times 2)$	17	$\mathrm{Gu}_{42}(x_4) = \exp(-(\lvert x_4 - 1.3848\rvert/\lvert 0.1213\rvert) \times 2)$
8	$\mathrm{Gu}_{23}(x_2) = \exp(-(\lvert x_2 - 1.5765\rvert/\lvert 0.1750\rvert) \times 2)$	18	$\mathrm{Gu}_{43}(x_4) = \exp(-(\lvert x_4 - 2.0093\rvert/\lvert 0.1261\rvert) \times 2)$
9	$\mathrm{Gu}_{24}(x_2) = \exp(-(\lvert x_2 - 1.3779\rvert/\lvert 0.0003\rvert) \times 2)$	19	$\mathrm{Gu}_{44}(x_4) = \exp(-(\lvert x_4 - 1.1342\rvert/\lvert 0.1246\rvert) \times 2)$
10	$\mathrm{Gu}_{25}(x_2) = \exp(-(\lvert x_2 - 3.0037\rvert/\lvert 0.0510\rvert) \times 2)$	20	$\mathrm{Gu}_{45}(x_4) = \exp(-(\lvert x_4 - 1.6871\rvert/\lvert 0.1175\rvert) \times 2)$

TABLE 4: Number of neurons in each layer of TS-FNN.

	First layer	Second layer	Third layer	Fourth layer
Antecedent network	4	20	5	5
Consequent network	5	11	5	1

TABLE 5: Consequent network parameter training results.

p_{1i}^1					p_{2i}^2				
0.4773	0.0749	0.6731	0.0389	0.5961	0.3543	0.1951	0.3349	−0.2759	0.6936
−0.0612	0.4454	0.3194	0.0080	0.6034	−0.3855	−0.0272	−0.1542	0.2631	−0.1080
0.5493	−0.0567	−0.2279	0.2641	−0.3716	−0.0477	0.0297	0.5030	0.1217	0.3906
0.6018	0.4309	−0.1189	0.6352	0.2714	0.0866	−0.2002	0.3345	−0.2826	0.4592
0.4378	−0.1727	0.0229	0.4259	−0.1466	0.1848	0.6407	0.2744	−0.1019	0.3758
−0.2025	0.3485	−0.1682	0.0041	0.0330	0.4103	0.5250	0.6120	−0.2094	0.0872
−0.1528	0.0966	0.3042	0.4979	−0.1313	0.5285	0.4929	0.0388	0.3712	−0.0690
−0.0172	0.1219	−0.1331	0.4291	−0.2181	0.5231	−0.2441	0.4129	0.4890	0.5448
−0.3210	0.6402	0.0462	−0.0245	−0.2632	0.4507	−0.0911	−0.2587	0.3511	0.4184
−0.1133	−0.1793	0.2354	0.2057	−0.0893	0.3479	0.5745	0.0471	0.4944	0.3721
0.0780	0.4552	0.0436	−0.1228	−0.0188	−0.3268	0.5609	−0.0953	−0.1683	−0.3001

TABLE 6: Part of the sample and test data.

Sample	x_1	x_2	x_3	x_4	PTS-FNN prediction result	TS-NN prediction result	Actual health status
1	0.0183	0.0181	0.0185	0.0181	1	1	1
2	0.0186	0.0188	0.0188	0.0187	1	1	1
3	0.0189	0.0187	0.1024	0.0943	3	2	3
4	0.0321	0.0452	0.0187	0.0182	2	2	2
5	0.0181	0.0190	0.0186	0.0182	1	1	1
6	0.0186	0.0184	0.0188	0.0189	1	1	1
7	0.0184	0.0816	0.0368	0.0988	3	2	3
8	0.0357	0.0132	0.0189	0.0179	1	1	1
...	...	...	...	...	...	...	...
78	0.2240	0.4577	0.0183	0.2570	4	3	4
79	0.0182	0.0187	0.0186	0.0189	1	1	1
80	0.0377	0.0391	0.0450	0.2490	2	2	2

TABLE 7: Comparison of model prediction accuracy.

		TS-FNN prediction result				PTS-FNN prediction result			
		State 1	State 2	State 3	State 4	State 1	State 2	State 3	State 4
Actual health status	State 1	42	1	0	0	43	0	0	0
	State 2	0	14	2	0	0	15	1	0
	State 3	0	3	11	0	0	2	12	0
	State 4	0	0	1	6	0	0	0	7

T-S fuzzy inference model, and fuzzy neural network modeling method. We also studied the PCM classification algorithm of input data of PTS-FNN model, the antecedent network of TS-FNN prediction model, and the construction method of postcomponent network. Finally, the implementation process of PCM classification algorithm and TS-FNN prediction model was given.

The R2R processing equipment health prediction experiment system was built and the PTS-FNN model experiment was carried out. The experimental results showed that the training time of PTS-FNN model was 50.37% less than the standard TS-FNN prediction model. The prediction accuracy increased by 5.48%, and the PTS-FNN had no in the judgment of state 1 and state 4.

The proposed PTS-FNN health state prediction model was used for health state prediction of continuous manufacturing systems with time-varying and multistation features such as R2R processing equipment, which improved the real-time and accuracy of the prediction model. The main problem which plagued the large-scale manufacturing of flexible photovoltaic film materials was solved.

Data Availability

The data that support the findings of this study are available from the corresponding author on request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Yaohua Deng and Kexing Yao contributed equally to this work.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under (Grant No. 51675109), Natural Science Foundation of Guangdong Province, China, under (Grant No. 2017A030313308), Provincial Science and Technology Plan Project of Guangdong Province, China, under (Grant No. 2016B010124002), Provincial Science and Technology Plan Project of Guangdong Province, China, under (Grant No. 2016B010112003), and Provincial Research and Development Project in Key Areas of Guangdong Province, China, under (Grant No. 2019B010154001).

References

[1] J. Lee, H. Davari, J. Singh, and V. Pandhare, "Industrial Artificial Intelligence for industry 4.0-based manufacturing systems," *Manufacturing Letters*, vol. 18, pp. 20–23, 2018.

[2] T. Xia, L. Xi, E. Pan, and J. Ni, "Reconfiguration-oriented opportunistic maintenance policy for reconfigurable manufacturing systems," *Reliability Engineering & System Safety*, vol. 166, pp. 87–98, 2017.

[3] J. Lee, Di Yuan, and X. Jia, "Etc. Enhanced virtual metrology on chemical mechanical planarization process using an integrated model and data-driven approach," *International Journal of Prognostics and Health Management*, vol. 8, pp. 1–8, 2017.

[4] J. Ni and T. Xia, "etc. Operating load based real-time rolling Grey forecasting for machine health prognosis in dynamic maintenance schedule," *Journal of Intelligent Manufacturing*, vol. 26, pp. 269–280, 2015.

[5] Y. Deng, K. Yao, Q. Lu, and Q. Lu, "Research on prediction method of performance degradation of flexible optoelectronic film material processing equipment based on adaptive fuzzy clustering," *Mathematical Problems in Engineering*, vol. 2018, pp. 1–9, 2018.

[6] J. Zhao and Z. Ouyang, "Etc. Combined prediction on avionics state optimized by MAGA," *Acta Armamentarii*, vol. 37, no. 4, pp. 727–734, 2016.

[7] Y. Deng and G. Liu, "ATS-FNN modeling and simulation of FWP processing deformation compensation prediction," *Journal of South China University of Technology (Natural Science Edition)*, vol. 40, no. 3, pp. 137–142, 2012.

[8] Y. Deng, Q. Lu, K. Yao, and N. Zhou, "Study on phosphor powder precipitation model in flexible material manufacturing process based on neuro-fuzzy networkflexible material manufacturing process based on neuro-fuzzy network," *Optik*, vol. 168, pp. 563–576, 2018.

[9] C. Lee, H. Lee, H. Seol, and Y. Park, "Evaluation of new service concepts using rough set theory and group analytic hierarchy process," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3404–3412, 2012.

[10] R. Krishnapuram, "A possibilistic approach to clusterin," *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 2, pp. 98–ll0, l993.

[11] N. R. Pal, K. Pal, J. M. Keller, and J. C. Bezdek, "A possibilistic fuzzy c-means clustering algorithm," *IEEE Transactions on Fuzzy Systems*, vol. 13, no. 4, pp. 517–530, 2005.

[12] A. Bhagat, N. Kshirsagar, P. Khodke, K. Dongre, and S. Ali, "Penalty parameter selection for hierarchical data stream clustering," *Procedia Computer Science*, vol. 79, pp. 24–31, 2016.

Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 6038619, 13 pages
https://doi.org/10.1155/2020/6038619

Research Article

An Approach to Semantic and Structural Features Learning for Software Defect Prediction

Shi Meilong,[1] Peng He ,[1,2] Haitao Xiao,[1] Huixin Li,[1] and Cheng Zeng[1]

[1]*School of Computer Science and Information Engineering, Hubei University, Wuhan 430062, China*
[2]*Hubei Key Laboratory of Applied Mathematics, Hubei University, Wuhan 430062, China*

Correspondence should be addressed to Peng He; penghe@hubu.edu.cn

Received 20 December 2019; Revised 1 February 2020; Accepted 24 February 2020; Published 6 April 2020

Guest Editor: Chunlai Chai

Research on software defect prediction has achieved great success at modeling predictors. To build more accurate predictors, a number of hand-crafted features are proposed, such as static code features, process features, and social network features. Few models, however, consider the semantic and structural features of programs. Understanding the context information of source code files could explain a lot about the cause of defects in software. In this paper, we leverage representation learning for semantic and structural features generation. Specifically, we first extract token vectors of code files based on the Abstract Syntax Trees (ASTs) and then feed the token vectors into Convolutional Neural Network (CNN) to automatically learn semantic features. Meanwhile, we also construct a complex network model based on the dependencies between code files, namely, software network (SN). After that, to learn the structural features, we apply the network embedding method to the resulting SN. Finally, we build a novel software defect prediction model based on the learned semantic and structural features (SDP-S2S). We evaluated our method on 6 projects collected from public PROMISE repositories. The results suggest that the contribution of structural features extracted from software network is prominent, and when combined with semantic features, the results seem to be better. In addition, compared with the traditional hand-crafted features, the *F*-measure values of SDP-S2S are generally increased, with a maximum growth rate of 99.5%. We also explore the parameter sensitivity in the learning process of semantic and structural features and provide guidance for the optimization of predictors.

1. Introduction

Software defect is an error in the code or incorrect behavior in software execution, also defined as failure to meet intended or specified requirements. Software reliability is regarded as one of the crucial problems in software engineering. Thus, the models used to ensure software quality are required, and the software defect prediction model is one of them. Defect prediction can estimate the most defect-prone software components precisely and help developers allocate limited resources to those bits of the systems that are most likely to contain defects in testing and maintenance phases [1].

As we all know, in software life cycle, the earlier you find the defect, the less it costs to fix [2]. Therefore, how to detect defects quickly and accurately is always an open challenge in the field of software engineering and has attracted extensive attention from industry and academia.

Typical defect prediction is composed of two parts: features extraction from source files and classifiers construction using various machine learning algorithms. Existing methods are dominated by traditional hand-crafted features, namely, source code metrics (e.g., CK, Halstead, MOOD, and McCabe's CC metrics). Unfortunately, these metrics generally overlook some important information implied in the code, such as semantic and structural information. Meanwhile, extensive machine learning algorithms have been adopted for software defect prediction, including Support Vector Machine (SVM), Naïve Bayes (NB), Decision Tree (DT), etc.

Programs have well-defined syntax and rich semantics hidden in the Abstract Syntax Trees (ASTs), which have been successfully used for programming patterns mining [3, 4], code completion [5, 6], and code plagiarism detection [7]. For example, Figure 1 shows two Java files, both of which contain an assignment statement, a while statement, a function call, and an increment statement. If we use traditional features to represent these two files, they are identical because of the same source code characteristics in terms of lines of code, function calls, raw programming tokens, etc. However, they are actually quite different according to semantic information. In other words, semantic information as new discriminative features should also be useful for characterizing defects for improving defect prediction.

At present, deep learning has emerged as a powerful technique for automated feature generation, since deep learning architecture can effectively capture highly complicated nonlinear features. To make use of its powerful feature generation ability, some researchers [8, 9] have already leveraged deep learning algorithms, such as Deep Belief Network (DBN) and Convolutional Neural Network (CNN) in learning semantic features from programs' ASTs, and verified that it outperforms traditional hand-crafted features in defect prediction.

As demonstrated by researchers [9], CNN is superior to DBN because of CNN's powerful efficiency to capture local patterns. Hence, CNN is capable of detecting local patterns and then conducting defect prediction. Since slight difference in local code structure, such as the code order difference illustrated in Figure 1, may trigger huge variance in the global program, we apply CNN instead of DBN to the construction of the defect prediction model.

However, the abovementioned studies still overlook the globally structural information among program files which can lead to more accurate defect prediction, although they consider the fine-grained semantic information in the program files. In order to better represent the global structure of software, previous studies [10–12] have successfully abstracted a software as a directed dependency network using complex network theory, usually termed as software network (SN), where software components such as files, classes, or packages are nodes and the dependency relationships between them are edges. Furthermore, using network analysis technologies, they have demonstrated the effectiveness of network structure information in improving the performance of defect prediction.

Unfortunately, network features the above authors used in defect prediction modeling, such as modularity, centrality, and node degree, still belong to the traditional hand-crafted features. As an emerging deep learning technology, network representation learning becomes a novel approach for automatically learning latent features of nodes in a network [13] and receives much attention. Therefore, using representation learning to extract the structural information from code files and further apply the learned features to defect prediction may effectively improve the performance of existing prediction models.

Unlike the existing studies, in our work, instead of using traditional hand-crafted metrics, we introduced deep learning technologies to automatically extract the semantic

```
int i = 6;
while (i < 10) {
    func ();
    i++;
}
```

File1. java

```
int i = 6;
func ();
while (i < 10)  {
    i++;
}
```

File2. java

Figure 1: A simple case.

(local fine-grained) and structural (global coarse-grained) features of code files for defect prediction modeling and seek empirical evidence that they can achieve acceptable performance compared with the benchmark models. Our contributions to the current state of research are summarized as follows:

(i) We further demonstrated that the automatically learned semantic features can significantly improve defect prediction compared to traditional features

(ii) In terms of improving the performance of defect prediction, we also validated that the contribution of structural features extracted from software network by representation learning is comparable to that of semantic features on the whole

(iii) Interestingly, we also found that the combination of semantic and structural features has greater impact on the improvement of prediction performance

The rest of this paper is organized as follows. Section 2 is a review of related work on this topic. Sections 3 and 4 describe the preliminary theories and the approach of our empirical study, respectively. Section 5 is the detailed experimental setups and the primary results. Some threats to validity that could affect our study are presented in Section 6. Finally, Section 7 concludes the work and presents the agenda for future work.

2. Related Studies

2.1. Software Defect Prediction. Software defect prediction technology has been widely used in software quality assurance and can effectively reduce the cost of software development. It uses the previous defect data to build a predictor and then employs the established model to predict whether a new code fragment is defective. At present, conventional software defect prediction can be roughly divided into two steps. The first stage is feature extraction, which makes the representation of defects more efficient by manually designing some features or combining existing features. The second is the classification by machine learning methods, specifically, by using the learning algorithm to establish an accurate model, so as to provide better prediction.

Most defect prediction techniques leverage features that are composed of the hand-crafted code metrics to train machine learning-based classifiers [14]. Commonly used code metrics include static code metrics and process metrics. The former include McCabe metrics [15], CK metrics [16], and MOOD metrics [17], which are widely examined and used for defect prediction. Compared to the above static code metrics, process metrics can reveal much about how programmers collaborate on tasks. Moser et al. [18] used the number of revisions, authors, past fixes, and ages of files as metrics to predict defects. Nagappan and Ball [19] proposed code churn metrics and showed that these features were effective for defect prediction. Hassan [20] used entropy of change features to predict defects. Other process metrics, including developer individual characteristics [21] and collaboration between developers [22, 23], were also useful for defect prediction.

Meanwhile, many machine learning algorithms have been adopted for defect prediction, including Support Vector Machine (SVM) [24], Bayesian Belief Network [25], Naive Bayes (NB) [26], Decision Table (DT) [1], neural network [27], and ensemble learning [28]. For instance, Kumar and Singh [24] evaluated the capability of SVM with combinations of different feature selection and extraction techniques in predicting defective software modules and tested on five NASA datasets. In [25], the authors predicted the quality of a software by using the Bayesian Belief Network. Arar and Ayan [26] proposed a Feature Dependent Naive Bayes (FDNB) classification method to software defect prediction and evaluated their approach on PROMISE datasets. He et al. [1] examined the performance of tree-based machine learning algorithms on defect prediction from the perspective of simplifying metric. Li et al. [28] proposed a novel Two-Stage Ensemble Learning (TSEL) approach to defect prediction using heterogeneous data. They experimented on 30 public projects and showed that the proposed TSEL approach outperforms a range of competing methods.

In addition, to overcome the lack of training data, a cross-project defect prediction (CPDP) model was proposed by some research studies. To improve the performance of CPDP, Turhan et al. [29] proposed to use a nearest-neighbor filter for target project to select training data. Nam et al. [30] proposed TCA+, which adopted a state-of-the-art technique called Transfer Component Analysis (TCA) and optimized normalization process. They evaluated TCA+ on eight open-source projects, and the results showed that TCA+ significantly improved CPDP. Nam et al. [21] also presented methods for defect prediction that match up different metrics in different projects to address the heterogeneous data problem in CPDP.

2.2. Deep Learning in Software Engineering. Representation learning has been widely applied to feature learning, which can capture the highly complex nonlinear information. Recently, deep learning algorithms have been adopted to improve research tasks in software engineering. Yang et al. [31] proposed an approach to generate features from existing features by using Deep Belief Network (DBN) and then used these new features to predict whether a commit is buggy or not. This work was motivated by the weaknesses of Logistic Regression (LR) that LR cannot combine features to generate new features. They used DBN to generate features from 14 traditional features and several developer experience-related features. Wang et al. [8] also leveraged DBN to automatically learn semantic features from token vectors extracted from programs' Abstract Syntax Trees (ASTs) and further validated that the learned semantic features significantly improve the performance of defect prediction. Similarly, Li et al. [9] used convolution neural network for feature generation based on the program's AST and proposed a framework of defect prediction. To explore program's semantics, Phan et al. [32] attempted to learn new defect features from program control flow graphs by convolution neural network.

However, these studies still ignore the structural features of programs, such as the dependencies between program files. Prior studies [12, 33–35] have demonstrated the effectiveness of network structure information in improving the performance of the defect prediction model. Nowadays, the node of a network can be represented as a low-dimensional vector by means of network embedding. A large number of network embedding algorithms have been successfully applied in network representational learning, including DeepWalk [36], Node2vec [37], and LINE [38]. Through the representational learning of software networks formed by various dependencies between code files, in this paper, we extract structural features of program files, so as to supplement the existing semantic features for defect prediction.

2.3. Software Network. In recent years, software networks (SN) have been widely utilized to characterize the problems in software engineering practices [39]. For example, some complexity metrics based on software networks are proposed to evaluate the software quality. Gu et al. [40] proposed a metric of cohesion based on SN for measuring connectivity of class members. From the perspective of social network analysis (SNA), Zhang et al. [41] put forward a suite of metrics for static structural complexity, which overcomes the limitations of traditional OO software metrics. Ma et al. [42] proposed a hierarchical set of metrics in terms of coupling and cohesion and analyzed a sample of 12 open-source OO software systems to empirically validate the set. Pan and Chai [43] leverage a meaningful metric based on SN to measure software stability.

In addition to complexity metrics, software network-based measures for stability and evolvability have also been presented by some researchers. Zhang et al. [41] analyzed the evolution of software networks from several kinds of object-oriented software systems and discovered some evolution rules such as distance among nodes increase and scale-free property. Gu and Chen [44] validated software evolution laws using network measures and discussed the feasibility of modeling software evolution. Peng et al. [11] constructed the software network model from a multigranularity perspective and further analyzed the evolutions of three open-source software systems in terms of network scale, quality, and structure control indicators, using complex network theory.

Besides, for software ranking task, Srinivasan et al. [45] proposed a software ranking model based on software core components. Pan et al. constructed [46] a novel model ElementRank based on SN, which leverages multilayer complex network to rank software. In addition, SN is also applied to analyze the structure of software structure [47]. Furthermore, a generalized k-core decomposition model [48] is leveraged to identify key class.

3. Preliminaries

3.1. Overview of Software Defect Prediction. Software defect prediction plays an important role in reducing the cost of software development and ensuring the quality of software. It can find the possible defective code blocks according to the features of historical data, thus allowing workers to focus their limited resources on the defect-prone code. Figure 2 presents a basic framework of software defect prediction and has been widely used in existing studies [1, 8, 12, 18, 19, 24, 25].

Most defect prediction models are based on machine learning; therefore, it is a first step to collect defect datasets. The defect datasets consist of various code features and labels. Commonly used features are various software code metrics mentioned above. Label indicates whether the code file is defective or not for binary classification. In the setting, predictor is trained using the labeled instances of project and then used to predict unlabeled ("?") instances as defective or clean. In the process of defect prediction, the instances used to learn classifier is called training set and the instances used to evaluate classifier are called test set.

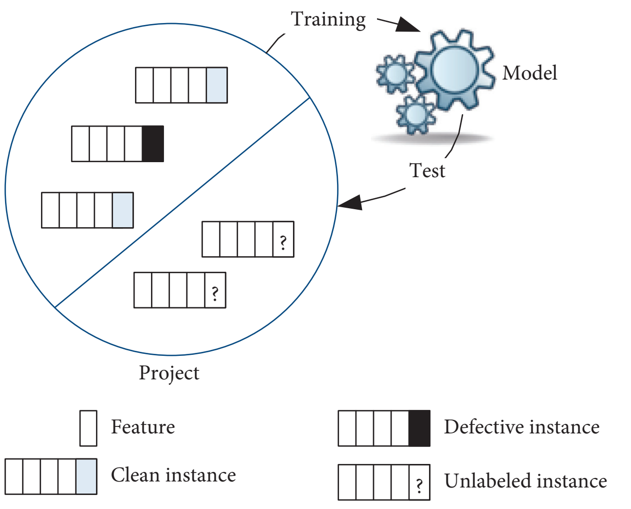


FIGURE 2: General framework of software defect prediction.

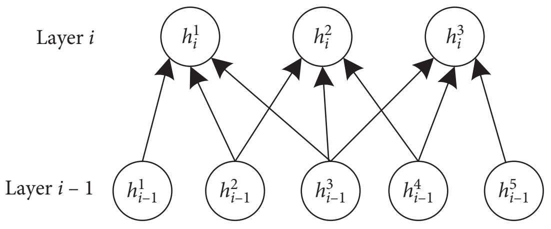


FIGURE 3: A convolutional layer architecture.

3.2. Convolutional Neural Network. Convolutional neural network (CNN) is one of the most popular algorithms for deep learning, a specialized kind of neural networks for processing data that have a known gridlike topology [49]. Compared with traditional artificial neural network, CNN has many advantages and has been successfully demonstrated in many fields, including NLP [50], image recognition [51], and speech recognition [52]. Here, we will use CNN for learning semantic features from software source code through multilayer nonlinear transformation, so as to replace the manual features of code complexity. Moreover, the deep structure enables CNN to have strong representation and learning ability. CNN has two significant characteristics: local connectivity and shared weight, which are helpful to extract features for our software defect prediction modeling.

Compared with the full connection in feedforward neural network, the neurons in the convolutional layer are only connected to some neurons of adjacent layer and generate spatially local connection. As shown in Figure 3, each unit h_i in the hidden layer i is only connected with 3 adjacent neurons in the layer $i-1$, rather than with all the neurons. Each subset acts as a local filter over the input vectors, which can produce strong responses to a spatially local input pattern. Each local filter applies a nonlinear transformation: multiplying the input with a linear filter, adding a bias term, and then applying a nonlinear function. In Figure 3, if we denote the k-th hidden unit in layer i as h_i^k, then the local filter in layer $i-1$ acts as follows:

$$h_i^k = f\left(\sum_{j=1}^{3} W_{i-1}^j * h_{i-1}^j + b_{i-1}\right), \quad (1)$$

where W_{i-1} and b_{i-1} denote the weights and bias of the local filter, respectively.

In addition, sparse connectivity has regularization effect, which improves the stability and generalization ability of network structure and can effectively avoid overfitting. At the same time, it reduces the number of weight parameters, is beneficial to accelerate the learning of neural network, and reduces the memory cost in calculation.

Parameter sharing refers to using the same parameters (W and b) for each local filter. In previous neural networks, when calculating the output of a layer, the parameters of each unit are different. However, in CNN, the same filter should share the same weight W and bias b. The reason is that a repeating unit can identify feature regardless of its position in the receptive field. On the other hand, weight sharing enables us to conduct feature extraction more effectively.

3.3. Construction of Software Network. In software engineering, researchers in the field of complex systems used complex networks theory to represent software systems by taking software components (such as package, file, class, and method) as nodes and their dependency relationships as edges, named as software network. The role of SN in software defect prediction, evolution analysis, and complexity measurement has been confirmed in the literature [11, 12, 33–35].

Files are key software components in the software system, and they are gathered up by interactions. SN at file level can be defined as in Figure 4: Every file is viewed as a single node in SN, and the dependency and association relationships between files are represented by edges (directed or undirected). Let $SN = (V, E)$ represents the software network, where each file can be treated as node $n_i\,(n_i \in V)$. The relationships between every pair of files, if exist, form a directed edge $e_i\,(e_i \in E)$.

3.4. Network Embedding. Network embedding (EM) is to map information networks into low-dimensional spaces, in which every vertex is represented as a low-dimensional vector. Such a low-dimensional embedding is very useful in a variety of applications such as node classification [3], link prediction [10], and personalized recommendation [23]. So far, various network embedding methods have been proposed successively in the field of machine learning. In this paper, we adopt Node2vec algorithm to embedding learning of the token vector.

Node2vec performs a random walk on neighbor nodes and then sends the generated random walk sequences to the Skip-gram model for training. In Node2vec, a 2nd-order random walk with two parameters p and q are used to flexibly sample neighborhood nodes between BFS (breadth-first search) and DFS (depth-first search).

Formally, given a source node u, we simulate a random walk of fixed length l. Let c_i denote the ith node in the walk, starting with $c_0 = u$. Node c_i is generated by the following distribution:

$$P\left(c_i = x \,\middle|\, c_{i-1} = v\right) = \begin{cases} \dfrac{\pi_{vx}}{Z}, & \text{if } (v, x)\varepsilon E, \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where π_{vx} is the unnormalized transition probability between node v and x, and Z is the normalized constant.

As shown in Figure 5, the unnormalized transition probability π_{vx} sets to $\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$, where d_{tx} represents the shortest distance between node t and x:

$$\alpha_{pq}(t, x) = \begin{cases} \dfrac{1}{p}, & \text{if } d_{tx} = 0, \\ 1, & \text{if } d_{tx} = 1, \\ \dfrac{1}{q}, & \text{if } d_{tx} = 2. \end{cases} \tag{3}$$

4. Approach

In this section, we elaborate our proposed method of software defect prediction via semantic and structural features from source code files (SDP-S2S). The overall framework of SDP-S2S is presented in Figure 6. It mainly consists of three parts: the first part is the generation of semantic features from source codes and will be detailed in Section 4.1. The second part will be explained in Section 4.2, which focuses on the extraction of structural features from software network by network embedding learning. The last part refers to combining the semantic and structural features obtained in the first two steps into new features and used for software defect prediction.

4.1. Generation of Semantic Features. In order to achieve semantic features for each source code file, we should first map the source code files into ASTs and parse them as real-valued token vectors. After that, the token vectors are encoded and preprocessed, and then, the resulting token vectors are fed to CNN for feature learning to generate the semantic features. The generation process is described in detail in the following three steps.

4.1.1. Parsing AST. In order to represent the semantic features of source code files, we need to find the appropriate granularity as the representation of the source code. As previous study [8] has shown, AST can represent the semantic and structural information of source code with the most appropriate granularity. We first parse the source code files into ASTs by calling an open-source python package javalang. As treated in [9], we only select three types of nodes on ASTs as tokens: (1) nodes of method invocations and class instance creations, which are recorded as their corresponding names; (2) declaration nodes, i.e., method/type/enum declarations, whose values are extracted as tokens; and (3) control flow nodes, such as while, if, and throw, are recorded as their node types. Three types of selected nodes are listed in Table 1.

We call javalang's API to parse the source code into an AST. Given a path of the software source code, the token sequences of all files in the software will be output. As described in Algorithm 1, first traverse the source code files under path P, and each file is parsed into an AST via the PARSE-AST function. For each AST, we employ the preorder traversal strategy to retrieve the three types of nodes selected in Table 1 and receive the final token sequence.

4.1.2. Token Sequence Preprocessing. Since CNN only accepts inputs as numerical vectors, the token sequences generated from ASTs cannot be directly fed to CNN. Therefore, to get the numerical token vectors, it is necessary for the extracted token sequences to be converted into integer vectors. To do this, we give each token a unique integer ID, and the ID of the recurring token is identical. Note that, because the token sequences of all files are of unequal length, the converted integer token vectors may differ in their dimensions. Also, CNN requires input vectors to have the same length; hence, we append 0 to each integer vectors, making their lengths consistent with the longest vector. Additionally, during the encoding process, we filter out infrequent tokens which might be designed for a specific file and not generalized for other files. Specifically, we only

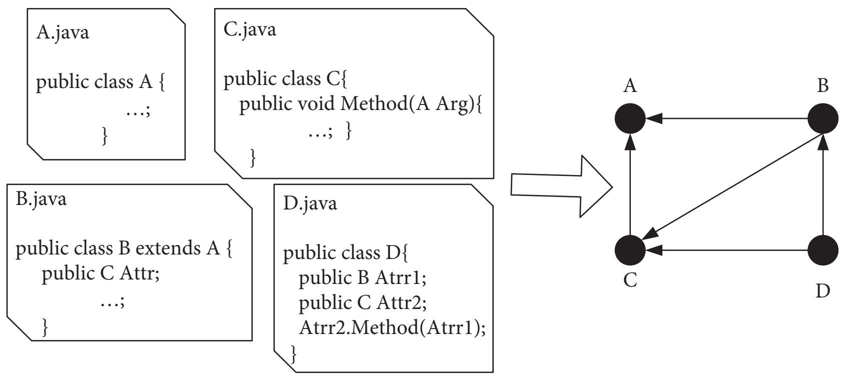


Figure 4: A single source code segment and its corresponding software network model at file level.

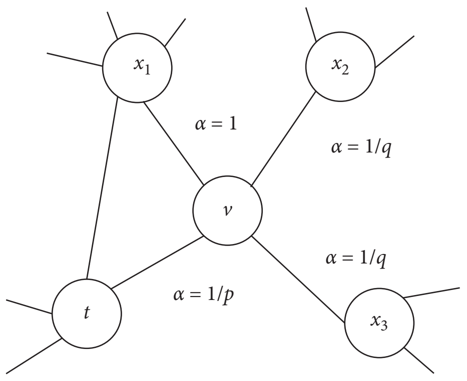


Figure 5: Random walk process in Node2vec.

encode tokens occurring three or more times, while denote the others as 0.

In addition, for software defect prediction, the class imbalance of defect dataset often exists. Specifically, the number of clean instances vastly outnumbers that of defective instances. Assuming that the number of clean instances is labeled as s_n and the number of defective samples is s_y, the imbalance rate (IR) [53] is used to measure the degree of imbalance:

$$\mathrm{IR} = \left\lfloor \frac{s_n}{s_y} \right\rfloor. \tag{4}$$

The larger the IR, the greater the imbalance, and vice versa. Imbalance data will degrade the performance of our model. To address this issue, we duplicate the defective files several times until the IR index is close to 1.

4.1.3. Building CNN. In this paper, we adopt classic architecture of CNN for feature learning. After encoding and preprocessing token vectors, exclude the input and output layers; we train the CNN model with four layers, including an embedding layer (turn integer token vectors into real-valued vectors of fixed size), a convolutional layer, a max-pooling layer, and a fully connected layer. The overall architecture is illustrated in Figure 7.

ReLu activation functions are used for training, and the implementation is based on Keras (http://keras.io). The output layers are activated by sigmoid function and used only for the parameters of the neural network weight matrix, to optimize the learning features. In addition, in this paper, Adam optimizer based on the improved stochastic gradient descent (SGD) algorithm is employed. Adam optimizer dynamically adjusts the learning rate for each parameter by calculating the first- and second-order moment estimations of the gradient. Compared with other optimization algorithms, Adam can ensure that the learning rate is distributed in an explicit range after each iteration, so that the parameter changes smoothly.

Given a project P, suppose it contains n source code files, all of which have been converted to integer token vectors $x \in R^l$ and of equal length l by the treatments described previously. Through the embedding layer, each token will be mapped to a d-dimension real-value vector. In other words, each file becomes a real-value matrix $X_{l\times d}$. As the input of convolutional layer, a filter $\mathscr{L} \in \mathbb{R}^{h\times d}$ is applied to a region of h tokens to produce a new feature. For example, a feature f_i is generated from a region of tokens $x_{i:i+h-1}$ by

$$f_i = \tanh\left(\mathscr{L} \cdot x_{i:i+h-1} + b\right). \tag{5}$$

Here, b is a bias term and tanh is a nonlinear hyperbolic tangent function. Each possible region of tokens in the description $\{x_{1:h}, x_{2:h+1}, \ldots, x_{i:i+h-1}, \ldots, x_{l-h+1:l}\}$ applies filter $\mathscr{L}$ to produce a feature map:

$$F = \left[f_1, f_2, \ldots, f_{l-h+1}\right], \tag{6}$$

where $f_i \in R^{l-h+1}$. Then, a 1-max-pooling operation is carried out over the mapped features and the maximum value $\widehat{F} = \max\{f\}$ is taken as the feature corresponding to this particular filter $\mathscr{L}$. Usually, multiple filters with different region sizes are used to get multiple features. Finally, a fully connected layer further generated the semantic features.

4.2. Generation of Structural Features. Before applying network embedding to represent structural features of source codes, it is necessary to build a software network model according to source files. As we did in the previous studies [11, 12], we use DependencyFinder API to parse the compiled source files (.zip or .jar extension) and extract their relationships using a tool developed by ourselves. With the directed software network, we further perform embedding

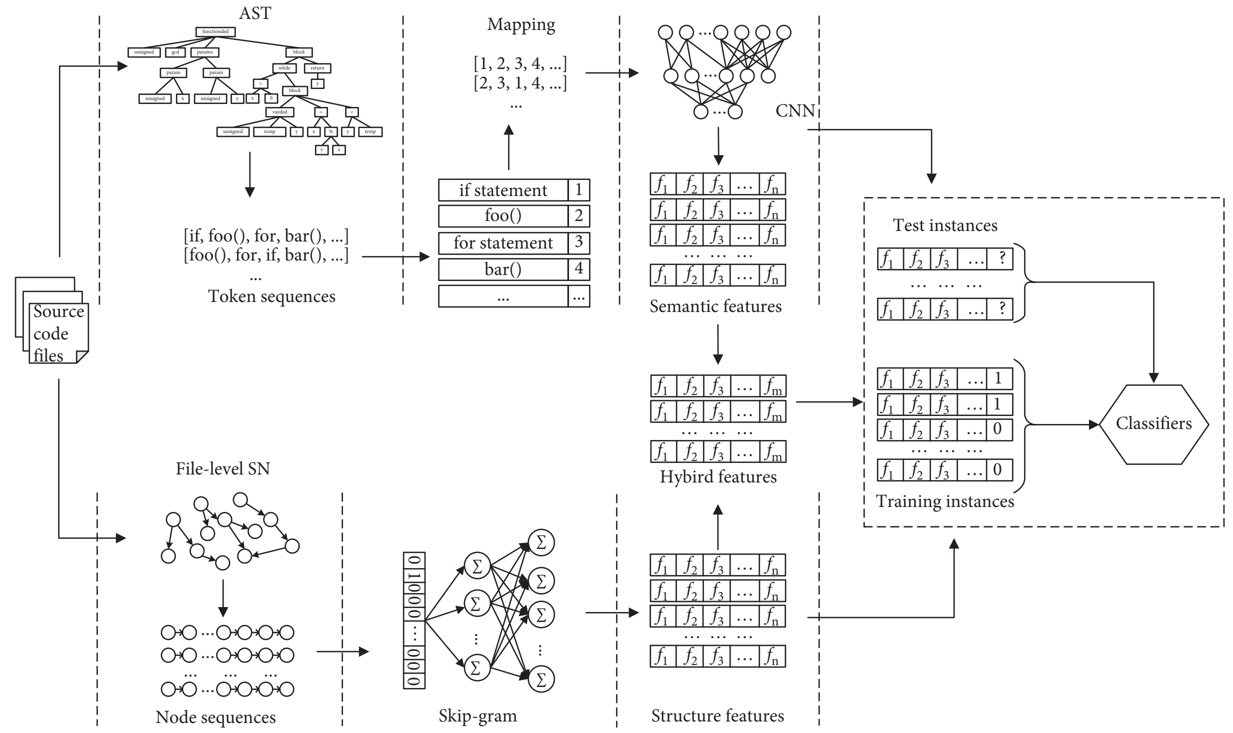


Figure 6: The overall framework of our approach.

Table 1: The selected AST nodes.

ID	Node types	AST node record
$Node_1$	Method invocations and class instance creations	MethodInvocation, SuperMethodInvocation, MemberReference, SuperMemberReference
$Node_2$	Declaration nodes	PackageDeclaration, InterfaceDeclaration, ClassDeclaration, MethodDeclaration, ConstructorDeclaration, VariableDeclarator, CatchClauseParameter, FormalParameter , TryResource, ReferenceType, BasicType
$Node_3$	Control flow nodes	IfStatement, WhileStatement, DoStatementForStatement, AssertStatement, BreakStatementContinueStatement, ReturnStatement, ThrowStatement, SynchronizedStatement, TryStatement, SwitchStatementBlockStatement, StatementExpression, CatchClauseSwitchStatementCase, ForControl, EnhancedForControl

learning using the Node2vec method. For more details on Node2vec, please refer to the literature [37].

4.3. Feature Concatenation. So far, we have got the semantic and structural features of source code files, respectively. Here, we label semantic feature as F_{semantic} and structural feature as $F_{\text{structural}}$. In order to verify the effectiveness of code semantic and structural features on software defect prediction, in addition to analyzing the impact of each type of generation feature on defect prediction, we also explore the impact of their combination. We directly concatenate the semantic feature vectors with structural feature vectors via *Merge* operator in Keras, and the resulting feature vectors presented as F_{hybrid}.

5. Experiment Setup

5.1. Dataset. In our study, 6 Apache open-source projects based on Java are selected (https://github.com/apache) and a total of 12 defect datasets available at the PROMISE repository (http://promise.site.uottawa.ca/SERepository/datasets-page.html) are picked for validation. Detailed information on the datasets is listed in Table 2, where #Avg. (files) and #Avg. (defect rate) are the average number of files and the average percentage of defective files, respectively. An instance in the defect dataset represents a class file and consists of two parts: independent variables including the learned features (e.g., the CNN-learned semantic features) and a dependent variable labeled as defective or not in this class file.

5.2. Evaluation Measures. The essence of defect prediction in this study is a binary classification problem. Note that a binary classifier can make two possible errors: false positives and false negatives. In addition, a correctly classified defective class file is a true positive and a correctly classified clean class file is a true negative. We evaluate the classification results in terms of Precision, Recall, and *F*-measure, which are described as follows:

```
     Input: path $p$ of the software source code
     Output: token sequences for all file
(1)  function EXTRACT (Path $p$)
(2)    $F$ = the set of source code files under path $p$;
(3)    for each $f \in F$ do
(4)      create sequence $s_{\text{file-token}}$;
(5)      $s_{\text{file-token}} \longleftarrow$ PARSE – AST($f$);
(6)      return $s_{\text{file-token}}$;
(7)    end for
(8)  end function
(9)  function PARSE-AST (File $f$)
(10)   create sequence $s_{\text{token}}$;
(11)   root $\longleftarrow$ javalang.parseFile2AST($f$);
(12)   for all ASTNode $k \in$ root do
(13)     if $k \in \text{Node}_1$ then
(14)       record its name and append to $s_{\text{token}}$;
(15)     else if $k \in \text{Node}_2$ then
(16)       record its declared value and append to $s_{\text{token}}$;
(17)     else if $k \in \text{Node}_3$ then
(18)       record its type and append to $s_{\text{token}}$;
(19)     end if
(20)   end for
(21)   return $s_{\text{token}}$
(22) end function
```

Algorithm 1: Parse-AST and return the token sequence of each file.

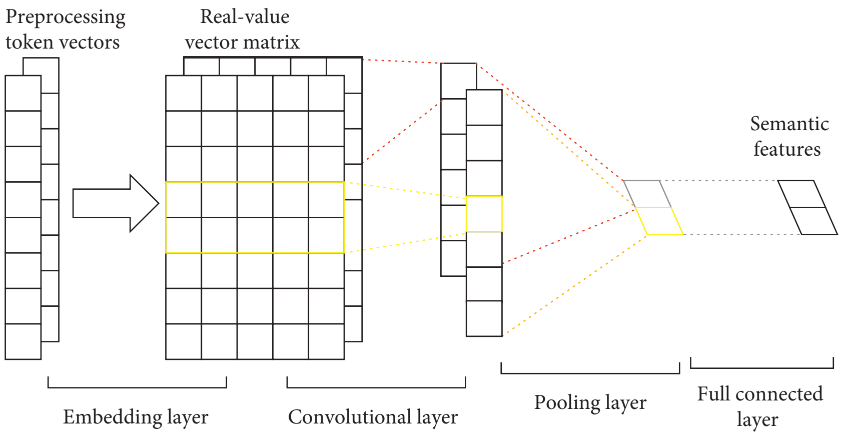


Figure 7: The process of CNN in our context.

$$\text{Precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}},$$
$$\text{Recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}, \qquad (7)$$
$$F-\text{measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}.$$

False positive refers to the predicted defective files that actually have no defects, and false negative refers to the actually defect-prone files predicted as clean. Precision and Recall are mutually exclusive in practice. Therefore, F-measure, as a weighted average of Precision and Recall, is more likely to be adopted. The value of F-measure ranges between 0 and 1, with values closer to 1 indicating better performance for classification results.

5.3. Experiment Design. First, to make a comparison between the traditional hand-crafted features and automatically learn features in our context, four scenarios will be considered in our experiments.

(i) SDP-base represents software defect prediction based on the traditional hand-crafted features

(ii) SDP-S1 represents software defect prediction based on the semantic features F_{semantic}

Table 2: Details of the datasets.

Project	Releases	Avg. (#files)	Avg. (defect rate) (%)	IR (imbalance rate)
Camel	1.4, 1.6	892	18.6	5.01
Lucene	2.0, 2.2	210	55.7	1.14
Poi	2.5, 3.0	409	64.7	0.55
Synapse	1.1, 1.2	239	30.5	2.70
Xalan	2.5, 2.6	815	48.5	1.07
Xerces	1.2, 1.3	441	15.5	5.20

(iii) SDP-S2 represents software defect prediction based on the structural features $F_{\text{structural}}$

(iv) SDP-S2S represents software defect prediction based on the semantic and structural features F_{hybrid}

Second, we will further explore prediction performance under different parameter settings for CNN and network embedding learning. For each project, note that we use the data from the older version to train the CNN model. Then, the trained CNN is used to generate semantic and structural features for both the older and newer versions. After that, we use the older version to build a defect prediction model and apply it to the newer version.

5.4. Experimental Results

5.4.1. Impact of Different Features. For each type of feature, Table 3 shows some interesting results: except for few cases (Poi), the *F*-measure values of SDP-S1, SDP-S2, and SDP-S2S are greater than those of the benchmark SPD-base, implying a significant improvement in accuracy. For example, for Camel, the growth rate of performance is more than 21.7%, when using the learned semantic and/or structural features. Especially when semantic and structural features are used comprehensively, the advantage is more obvious, indicated by the 99.5% performance growth. Additionally, note that for Xerces, although the growth rates of performance are slightly lower than that of Camel, it is still considerable, around 30%. For Lucene, Synapse, and Xalan, the corresponding maximum growth rates are 27.9% (0.7564), 22.6% (0.5204), and 10% (0.2406), respectively. The majority of positive growth rates suggest the feasibility of our proposed method of automatically learning features from source code files.

In Table 3, the results also show that SDP-S2S performs better than SDP-S1 and SDP-S2, indicated by more *F*-measure values in bold. Specifically, compared to the other two methods, SDP-S2S achieves the best performance on projects Camel, Lucene, and Xerces. In order to better distinguish their influences on defect prediction, we make further comparisons in terms of the Wilcoxon signed-rank test (*p*-value) and Cliff's effect size from a statistical perspective. In Table 4, the Wilcoxon signed-rank test highlights that there is no significant performance difference among the three predictors, indicated by the Sig. $p > 0.01$. However, when it comes to the Cliff's effect size delta, the negative values show that their effect size is different. Specifically, SDP-S2 outperforms SDP-S1, whereas SDP-S2S outperforms SDP-S2.

Table 3: *F*-measure values of defect prediction built with four types of features.

Projects	SDP-base	SDP-S1 (△%)	SDP-S2 (△%)	SDP-S2S (△%)
Camel	0.2531	0.5044 (99.3%)	0.3081 (21.7%)	**0.5049 (99.5%)**
Lucene	0.5912	0.6397 (8.2%)	0.6873 (16.3%)	**0.7564 (27.9%)**
Poi	0.7525	*0.7250 (−3.7%)*	**0.7892 (4.9%)**	*0.7340 (−2.5%)*
Synapse	0.4244	0.4444 (4.7%)	**0.5204 (22.6%)**	0.4390 (3.4%)
Xalan	0.6165	**0.6780 (10.0%)**	0.6229 (1.0%)	0.6623 (7.4%)
Xerces	0.1856	0.2406 (29.6%)	0.2432 (31.0%)	**0.2650 (42.8%)**

△% represents the growth rate of performance relative to SDP-base.

Table 4: Comparison of the distributions of three methods in terms of the Wilcoxon signed-rank test and Cliff's effect size.

	Sig. $p < 0.01$	Cliff's delta
SDP-S1 vs. SDP-S2	0.753	−0.056
SDP-S1 vs. SDP-S2S	0.345	−0.167
SDP-S2 vs. SDP-S2S	0.600	−0.056

With the evidences provided by the above activities, the approach of feature learning proposed in this paper is validated to be suitable for defect prediction.

5.4.2. Parameter Sensitivity Analysis

(1) Parameter Analysis of CNN. When using CNN to represent semantic features, the setting of some parameters of the network layer will affect the representation of semantic features and thus affect prediction performance. In this section, according to the key parameters of CNN, including the length of filter, the number of filters, and embedding dimensions, we tune the three parameters by conducting experiments with different values of the parameters. Note that, for other parameters, we directly present their values obtained from previous studies [9]: batch size is set as 32 and the training epoch is 15. By fixing other parameters, we analyze the influence of the three parameters on the results, respectively.

Figures 8–10, respectively, present the performance obtained under different filter lengths, different number of filters, and different embedding dimensions. It is not hard to find that all six curves reach the highest performance when the filter length is set to 10. The optimal number of filters is 20, where the performance generally reaches the peak. Interestingly, for project Xerces, when the number of filters is set as 100, the performance becomes particularly bad. With regard to the embedding dimensions, six curves on the whole are very stable, which means that the dimension of representation vector has a very limited impact on the prediction performance.

(2) Parameter Analysis of Software Network Embedding. For the generation of structural features, in Node2vec, a pair of parameters p and q controlling random walk will affect the learning. That is, different combinations of p and q determine the different ways of random walk in the process of network embedding and then generate different structural features. Therefore, we further analyze the two parameters.

Take Poi and Synapse, for example, we construct 25 groups of (p, q) and let $p, q \in [0.25, 0.5, 1, 2, 4]$. With different combinations (p, q), the results are as shown in Figure 11 and the effect of different combinations is different. For example, when the combination (p, q) is set as (4, 2) in Poi, the best performance 0.789 is achieved, and yet the suitable combinations (p, q) is (0.5, 0.25) in Synapse, and the F-measure value is 0.5204. Therefore, for each project in our context, we give out the optimal combination (p, q), shown in Table 5, so as to learn the defect structural information and generate corresponding structural features better.

6. Threats to Validity

To evaluate the feasibility of our method in defect prediction, we constructed four kinds of predictors according to different features and compared their performance. In this paper, although we do not explicitly compare with the state-of-the-art defect prediction techniques, SDP-S1 is actually equivalent to the method proposed in the literature [13]. Since the original implementation of CNN is not released, we have reproduced a new version of CNN via Keras. Throughout, we strictly followed the procedures and parameters settings described in the reference, such as the selection of AST nodes and the learning rate when training neural networks. Therefore, we are confident that our implementation is very close to the original model.

In this paper, our experiments were conducted with defect datasets of six open-source projects from the PROMISE repository, which might not be representative of all software projects. More projects that are not included in this paper or written in other programming languages are still to be considered. Besides, we only evaluated our approach in terms of different features and did not compare with other state-of-the-art prediction methods. To make our approach more generalizable, in the future, we will conduct experiments on a variety of projects and compare with more benchmark methods.

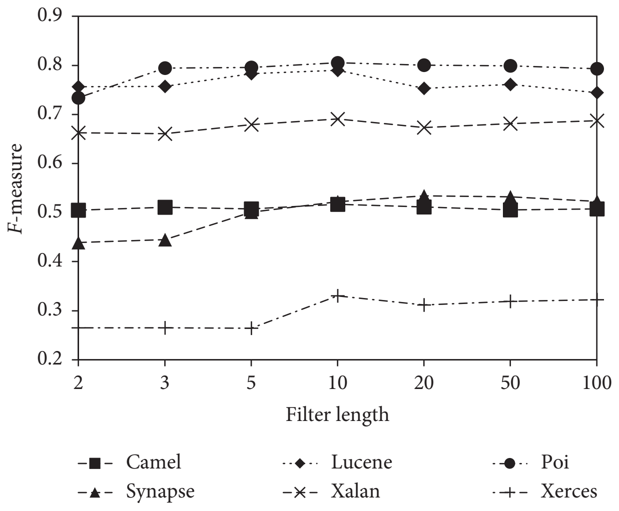


Figure 8: Different filter lengths.

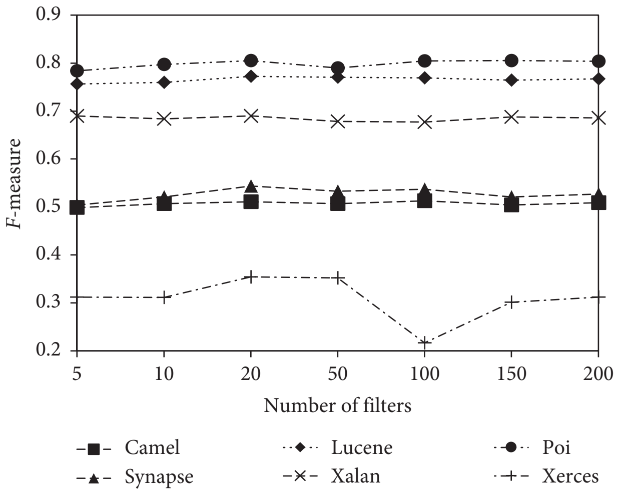


Figure 9: Different number of filters.

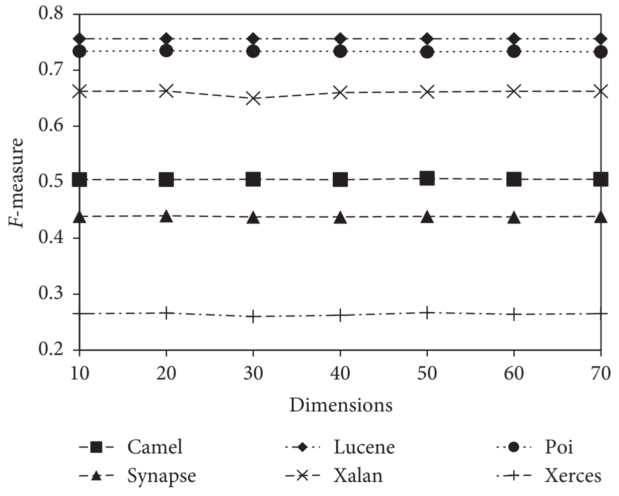
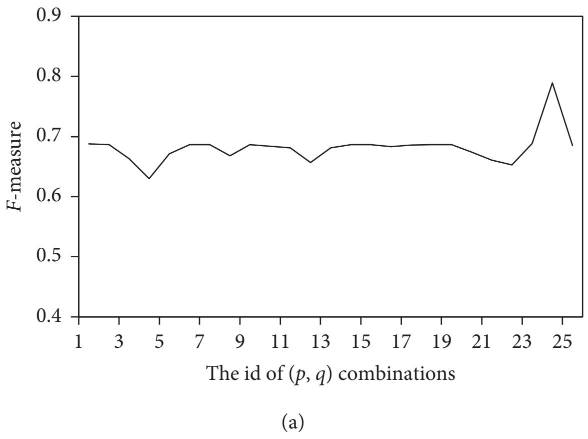


Figure 10: Different embedding dimensions.



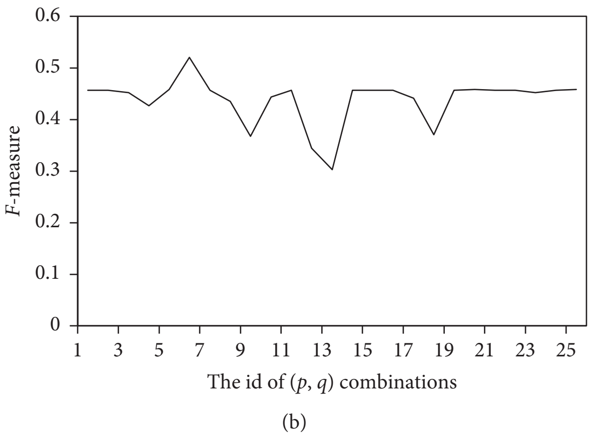


Figure 11: Results of different combinations of p and q. (a) Poi. (b) Synapse.

Table 5: The combination of p and q selected in this paper.

Project	p	q
Camel	2	1
Lucene	0.25	4
Poi	4	2
Synapse	0.5	0.25
Xalan	4	1
Xerces	0.5	4

7. Conclusion

This study aims to build better predictors by learning as much defect feature information as possible from source code files, to improve the performance of software defect predictions. In summary, this study has been conducted on 6 open-source projects and consists of (1) an empirical validation on the feasibility of the structural features that learned from software network at the file level, (2) an in-depth analysis of our method SDP-S2S combined with semantic features and structural features, and (3) a sensitivity analysis with regard to the parameters in CNN and network embedding.

Compared with the traditional hand-crafted features, the F-measure values are generally increased, the maximum is up to 99.5%, and the results indicate that the inclusion of structural features does improve the performance of SDP. Statistically, the advantages of SDP-S2S are particularly obvious from the perspective of Cliff's effect size. More specifically, the combination of semantic features and structural features is the preferred selection for SDP. In addition, our results also show that the filter length is preferably 10, the optimal number of filters is 20, and the dimension of the representation vector has a very limited impact on the prediction performance. Finally, we also analyzed the parameters p and q involved in the embedding learning process of software network.

Our future work mainly includes two aspects. On the one hand, we plan to validate the generalizability of our study with more projects written in different languages. On the other hand, we will focus on more effective strategies such as feature selection techniques. Last but not least, we also plan to discuss the possibility of considering not only CNN and Node2vec model but also RNN or LSTM for learning semantic features and graph neural networks for network embedding, respectively.

Data Availability

The experimental data used to support the findings of this study are available at https://pan.baidu.com/s/1H6Gw7UHb7vfBFFVfDBF6mQ.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (2018YFB1003801); the National Natural Science Foundation of China (61902114); Hubei Province Education Department Youth Talent Project (Q20171008); and Hubei Provincial Key Laboratory of Applied Mathematics (HBAM201901).

References

[1] P. He, B. Li, X. Liu, J. Chen, and Y. Ma, "An empirical study on software defect prediction with a simplified metric set," *Information and Software Technology*, vol. 59, pp. 170–190, 2015.

[2] E. N. Adams, "Minimizing cost impact of software defects," Report RC, IBM Research Division, New York, NY, USA, 1980.

[3] P. Yu, F. Yang, C. Cao et al., "API usage change rules mining based on fine-grained call dependency analysis," in *Proceedings of the 2017 Asia-Pacific Symposium on Internetware*, ACM, Shanghai, China, 2017.

[4] T. T. Nguyen, H. A. Nguyen, N. H. Pham et al., "Graph-based mining of multiple object usage patterns," in *Proceedings of the 2009 ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 24–28, Amsterdam, Nethelands, 2009.

[5] B. Cui, J. Li, T. Guo et al., "Code comparison system based on Abstract syntax tree," in *Proceedings of the 2010 IEEE*

International Conference on Broadband Network & Multimedia Technology, pp. 668–673, IEEE, Beijing, China, October 2010.

[6] S. Q. de Medeiros, G. D. A. Alvez Junior, and F. Mascarenhas, "Automatic syntax error reporting and recovery in parsing expression grammars," 2019, https://arxiv.org/abs/1905.02145.

[7] F. Deqiang, X. Yanyan, Y. Haoran, and B. Yang, "WASTK: a weighted Abstract syntax tree kernel method for source code plagiarism detection," *Scientific Programming*, vol. 2017, Article ID 7809047, 8 pages, 2017.

[8] S. Wang, T. Liu, and L. Tan, "Automatically learning semantic features for defect prediction," in *Proceedings of the 38th IEEE International Conference on Software Engineering*, pp. 297–308, IEEE, Austin, TX, USA, May 2016.

[9] J. Li, P. He, J. Zhu, and M. R. Lyu, "Software defect prediction via convolutional neural network," in *Proceedings of the 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pp. 318–328, IEEE, Prague, Czech Republic, July 2017.

[10] W. Pan, B. Li, Y. Ma, and Y. J. Liu, "Multi-granularity evolution analysis of software using complex network theory," *Journal of Systems Science and Complexity*, vol. 24, no. 6, pp. 1068–1082, 2011.

[11] H. E. Peng, P. Wang, L. I. Bing et al., "An evolution analysis of software system based on multi-granularity software network," *Acta Electronica Sinica*, vol. 46, no. 2, pp. 257–267, 2018.

[12] P. He, B. Li, Y. Ma, and L. He, "Using software dependency to bug prediction," *Mathematical Problems in Engineering*, vol. 2013, Article ID 869356, 12 pages, 2013.

[13] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: a survey," *IEEE Transactions on Big Data*, vol. 6, no. 1, pp. 3–28, 2018.

[14] D. Radjenović, M. Heričko, R. Torkar, and A. Živkovič, "Software fault prediction metrics: a systematic literature review," *Information & Software Technology*, vol. 55, no. 8, pp. 1397–1418, 2013.

[15] T. J. McCabe, "A complexity measure," *IEEE Transactions on Software Engineering*, vol. SE-2, no. 4, pp. 308–320, 1976.

[16] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476–493, 1994.

[17] J. Bansiya and C. G. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Transactions on Software Engineering*, vol. 28, no. 1, pp. 4–17, 2002.

[18] R. Moser, W. Pedrycz, and G. Succi, "A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction," in *Proceedings of the 2008 ACM/IEEE International Conference on Software Engineering*, pp. 181–190, IEEE, Leipzig, Germany, May 2008.

[19] N. Nagappan and T. Ball, "Using software dependencies and churn metrics to predict field failures: an empirical case study," in *Proceedings of the 2007 International Symposium on Empirical Software Engineering & Measurement*, pp. 364–373, IEEE, Madrid, Spain, September 2007.

[20] A. E. Hassan, "Predicting faults using the complexity of code changes," in *Proceedings of the 31st International Conference on Software Engineering*, pp. 78–88, IEEE, Vancouver, Canada, May 2009.

[21] J. Nam, W. Fu, S. Kim, T. Menzies, and L. Tan, "Heterogeneous defect prediction," *IEEE Transactions on Software Engineering*, vol. 44, no. 9, pp. 874–896, 2018.

[22] T. Wolf, A. Schroter, D. Damian, and T. Nguyen, "Predicting build failures using social network analysis on developer communication," in *Proceedings of the 2009 IEEE 31st International Conference on Software Engineering*, pp. 1–11, Vancouver, Canada, May 2009.

[23] M. Soto, Z. Coker, and C. L. Goues, "Analyzing the impact of social attributes on commit integration success," in *Proceedings of the 2017 IEEE/ACM 14th International Conference on Mining Software Repositories*, ACM, 2017, DOI: 10.1109/MSR.2017.34.

[24] R. Kumar and K. P. Singh, "SVM with feature selection and extraction techniques for defect-prone software module prediction," in *Proceedings of 6th International Conference on Soft Computing for Problem Solving*, pp. 279–289, Springer, Singapore, 2017.

[25] N. Fenton, M. Neil, W. Marsh, P. Hearty, Ł. Radliński, and P. Krause, "On the effectiveness of early life cycle defect prediction with Bayesian nets," *Empirical Software Engineering*, vol. 13, no. 5, pp. 499–537, 2008.

[26] Ö. F. Arar and K. Ayan, "A feature dependent naive Bayes approach and its application to the software defect prediction problem," *Applied Soft Computing*, vol. 59, pp. 197–209, 2017.

[27] Q. Cao, Q. Sun, Q. Cao, and H. Tan, "Software defect prediction via transfer learning based neural network," in *Proceedings of the 2015 1st International Conference on Reliability Systems Engineering*, pp. 1–10, Beijing, China, October 2015.

[28] Z. Li, X.-Y. Jing, X. Zhu, H. Zhang, B. Xu, and S. Ying, "Heterogeneous defect prediction with two-stage ensemble learning," *Automated Software Engineering*, vol. 26, no. 3, pp. 599–651, 2019.

[29] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Software Engineering*, vol. 14, no. 5, pp. 540–578, 2009.

[30] J. Nam, S. Pan, and S. Kim, "Transfer defect learning," in *Proceedings of the 35th International Conference on Software Engineering (ICSE)*, pp. 382–391, IEEE, San Francisco, CA, USA, May 2013.

[31] X. Yang, D. Lo, X. Xia, Y. Zhang, and J. Sun, "Deep learning for just-in-time defect prediction," in *Proceedings of the 2015 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, IEEE, 2015.

[32] A. V. Phan, M. L. Nguyen, and L. T. Bui, "Convolutional neural networks over control flow graphs for software defect prediction," 2018, https://arxiv.org/abs/1802.04986.

[33] T. Zimmermann and N. Nagappan, "Predicting defects using network analysis on dependency graphs," in *Proceedings of the 30th International Conference on Software Engineering*, pp. 531–540, IEEE, Leipzig, Germany, May 2008.

[34] H. Abandah and I. Alsmadi, "Dependency graph and metrics for defects prediction," *International Journal of ACM Jordan*, vol. 2, no. 3, pp. 115–119, 2013.

[35] P. Bhattacharya, M. Iliofotou, I. Neamtiu, and M. Faloutsos, "Graph-based analysis and prediction for software evolution," in *Proceedings of the International Conference on Software Engineering*, pp. 419–429, IEEE, Zurich, Switzerland, June 2012.

[36] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: online learning of social representations," 2014, https://arxiv.org/abs/1403.6652.

[37] A. Grover and J. Leskovec, "node2vec: scalable feature learning for networks," in *Proceedings of the 2016 ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, San Francisco, CA, USA, August 2016.

[38] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: large-scale information network embedding," in

Proceedings of the 24th International Conference on World Wide Web, pp. 1067–1077, Florence, Italy, May 2015.

[39] G. Concas, C. Monni, M. Orru et al., "Software systems through complex networks science: review, analysis and applications," in *Proceedings of the 4th International Workshop on Emerging Trends in Software Metrics*, pp. 14–20, San Francisco, CA, USA, May 2012.

[40] A. Gu, X. Zhou, Z. Li, Q. Li, and L. Li, "Measuring object-oriented class cohesion based on complex networks," *Arabian Journal for Science and Engineering*, vol. 42, no. 8, pp. 3551–3561, 2017.

[41] H. H. Zhang, W. J. Feng, and L. J. Wu, "The static structural complexity metrics for large-scale software system," *Applied Mechanics and Materials*, vol. 44–47, pp. 3548–3552, 2010.

[42] Y.-T. Ma, K.-Q. He, B. Li, J. Liu, and X.-Y. Zhou, "A hybrid set of complexity metrics for large-scale object-oriented software systems," *Journal of Computer Science and Technology*, vol. 25, no. 6, pp. 1184–1201, 2010.

[43] W. Pan and C. Chai, "Measuring software stability based on complex networks in software," *Cluster Computing*, vol. 22, no. S2, pp. 2589–2598, 2019.

[44] Q. Gu and D. Chen, "Validation and simulation of software system evolution rules using software networks," *Science China*, vol. 44, no. 1, pp. 20–36, 2014.

[45] S. M. Srinivasan, R. S. Sangwan, and C. J. Neill, "On the measures for ranking software components," *Innovations in Systems and Software Engineering*, vol. 13, no. 2-3, pp. 161–175, 2017.

[46] W. Pan, H. Ming, C. K. Chang, Z. Yang, and D.-K. Kim, "ElementRank: ranking Java software classes and packages using multilayer complex network-based approach," *IEEE Transactions on Software Engineering*, p. 1, 2019.

[47] W. Pan, B. Li, J. Liu, Y. Ma, and B. Hu, "Analyzing the structure of Java software systems by weighted k-core decomposition," *Future Generation Computer Systems*, vol. 83, pp. 431–444, 2018.

[48] W. Pan, B. Song, K. Li, and K. Zhang, "Identifying key classes in object-oriented software using generalized k-core decomposition," *Future Generation Computer Systems*, vol. 81, pp. 188–202, 2018.

[49] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, MA, USA, 2016, http://www.deeplearningbook.org.

[50] T. N. Sainath, B. Kingsbury, A. R. Mohamed et al., "Improvements to deep convolutional neural networks for LVCSR," in *Proceedings of the 2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 315–320, Olomouc, Czech Republic, December 2013.

[51] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, Springer, vol. 25, no. 2, , pp. 1–9, Berlin, Germany, 2012.

[52] D. Maturana and S. Scherer, "VoxNet: a 3D convolutional neural network for real-time object recognition," in *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928, IEEE, Hamburg, Germany, September 2015.

[53] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, 2012.

Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 5670215, 8 pages
https://doi.org/10.1155/2020/5670215

Research Article

Leverage Label and Word Embedding for Semantic Sparse Web Service Discovery

Chengai Sun,[1,2] Liangyu Lv,[1,2] Gang Tian,[1,2] Qibo Wang,[1,2] Xiaoning Zhang,[1,2] and Lantian Guo[3]

[1]*College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, China*
[2]*Key Laboratory for Wisdom Mine Information Technology of Shandong Province, Shandong University of Science and Technology, Qingdao, China*
[3]*School of Automation and Electronic Engineering, Qingdao University of Science and Technology, Qingdao, China*

Correspondence should be addressed to Gang Tian; tiangang@whu.edu.cn

Received 15 November 2019; Revised 30 January 2020; Accepted 28 February 2020; Published 24 March 2020

Guest Editor: Weifeng Pan

Information retrieval-based Web service discovery approach suffers from the semantic sparsity problem caused by lacking of statistical information when the Web services are described in short texts. To handle this problem, external information is often utilized to improve the discovery performance. Inspired by this, we propose a novel Web service discovery approach based on a neural topic model and leveraging Web service labels. More specifically, words in Web services are mapped into continuous embeddings, and labels are integrated by a neural topic model simultaneously for embodying external semantics of the Web service description. Based on the topic model, the services are interpreted into hierarchical models for building a service querying and ranking model. Extensive experiments on several datasets demonstrated that the proposed approach achieves improved performance in terms of F-measure. The results also suggest that leveraging external information is useful for semantic sparse Web service discovery.

1. Introduction

In the era of Big Data, a growing number of business enterprises worldwide are driven to deploy their business applications into Web services in both intranet and internet [1, 2]. A number of registry centers, such as, Programmableweb (http://www.programmableweb.com) and Mashape (https://www.mashape.com/) and business enterprises, have built their own service discovery mechanism to provide a convenient way to access these Web services. The search engine-based approaches are widely adopted among these registries. However, the discovery method-based searching engine technology which mainly focuses on keyword-based matching may result in the poor recall problem due to lacking of keywords in Web service descriptions, using of synonyms, or variations of keywords [3].

Two kinds of methods are often adopted to alleviate the poor recall problem in discovering nonsemantic Web services [4, 5]. The first one is to perform a broad searching and get a potentially large number of Web services which may not really be interest to users. The second one is to cluster services into similar functional clusters using the descriptions of Web services to enhance the capability of the search engine. Since this kind of method can effectively reduce the search space, it has attracted higher attentions from researchers [3–7].

There are some new issues emerged when using the aforementioned Web service discovery approaches in recent years. One is the semantic sparsity problem resulting from short text descriptions of Web services that there is no sufficient information to express the full semantics of the Web service. The current Web service marketplaces often briefly describe the main functions, the providers, and the types of a Web service using short sentences which do not contain enough statistic information so as to hinder effective similarity computing and pose challenges to traditional service retrieval approaches [8, 9]. Faced with this issue, transfer of external knowledge to enrich the semantic representation of short text documents has been proposed such as Tian et al. [5] transfer external knowledge by using Gaussian LDA and the word embedding model from auxiliary information to enhance the semantics of the Web services.

Inspirited by these excellent findings, we propose to introduce the word embeddings which have been shown to capture lexico-semantic regularities in the language. In the embedding space, words with similar syntactic and semantic properties are found to be close to each other [10]. Thus, this feature is particularly suitable to solve the problems of using synonyms/variations of keywords in the query. Furthermore, the context information such as the co-occurrence information in the word embeddings can be effectively used to enrich the semantics of a document. Inspired by this, we propose to introduce word embedding to handle the semantic sparsity problem in the discovery of Web service.

To enhance the clustering performance, extensive research has been carried out on category information [11]. Inspired by this, some topic models can directly integrate these information into the generative process of a topic model to improve topic quality and cluster accuracy. Some excellent work had been done to leverage external meta-information to enhance the topic model [12].

According to the above description, we propose a label-aided neural topic model (LNTM) derived from Gaussian LDA [13] which leverages word embeddings and external label information to improve Web service discovery.

Our main contributions are as follows:

(1) We presented an approach that leverages pretrained word embeddings to enrich the semantics of Web service descriptions

(2) We proposed a label-augmented neural topic model to retrieve the Web services based on word embeddings and categories of the Web services

(3) We experimentally illustrate that the proposed approach outperforms several other approaches with higher evaluation metrics

2. Related Work

Web service discovery provides a mechanism to discover relevant services from different service registries. Base on the description method of services, the Web service discovery can be generally divided into two categories: semantic-based and nonsemantic-based. For instance, the Ontology Web Language for Services- (OWL-S-) based service is a typically semantic description language. In contrast, WSDL, Web Application Description Language (WADL), and natural language are typical nonsemantic description languages. Semantic-based approaches mainly focus on high-level match-making [14, 15], whereas nonsemantic-based discovery methods utilize information retrieval techniques [3–7]. In the proposed approach, we concentrate on nonsemantic Web services.

The nonsemantic-based discovery approaches are fairly different due to different description languages. For example, Elgazzar et al. [3] preprocessed the WSDL document to extract content, types, messages, ports, and service name as main features for the discovery method and utilized information retrieve approach to enhance Web service discovery. The WSDL documents need be preprocessed to construct the features for representing the Web service. If the Web services use different description languages, the WSDL-based methods must be adjusted for the discovery process. In this paper, we focus on the discovery of Web services which have shorter description and may contain less features compared with the services with sufficient information files. Therefore, the above methods may fail to work since they lack ways to handle the semantic sparsity problem.

Several studies have found that it is helpful to leverage external information to handle the semantic sparse problems of information retrieval approach [4, 8, 16]. Chen et al. proposed an augment LDA model to utilize both WSDL and tags for Web service discovery so as to provide effective Web service clustering [16]. There are also different methods to handle with the semantic sparsity problem. For example, Hu et al. proposed to enhance the short text cluster by leveraging world knowledge [17]. Jin et al. utilized a transfer learning model to cluster short texts to embody auxiliary long texts [8]. These approaches can partially handle the semantic sparsity problem; however, they also have some limitations. For instance, Hu et al.'s work in [17] makes the implicit assumption that the auxiliary data are semantically related to the short texts, which may not be true in the real world. Similarly, work [8] makes the assumption that the topical structure of the two domains which is completely identical would not be wholly correct.

Some studies utilized the complex network to handle Web service clustering problems to introduce the capability of network-based software. Many approaches have been performed from a complex network perspective by representing software systems (or service-oriented software systems) as software networks (or software service networks). Ma et al. [18] and Pan et al. [19] analyzed the topological structure of software networks, revealing many shared properties such as small-world and scale-free. Şora and Chirila [20] and Pan et al. [21, 22] proposed approaches to identify key classes in Java systems. Ma et al. [18] and Pan et al. [23–25] proposed software metrics by using parameters in complex networks. Zhou and Wang [26] and Pan et al. [27, 28] proposed an approach to cluster services by using community detection approaches in complex networks. There works are helpful to utilize the capability of the complex network; however, it still has the problem of semantic sparsity.

Faced with above problems, we propose to introduce another solution that introduces external information by word embeddings which have been shown to be beneficial for the semantic sparsity [29].

Latent Dirichlet Allocation (LDA) and extensions have been proved as efficient methods for boosting the discovery performance of Web services [16, 30]. However, due to the base assumption that the words are discrete multinomial distribution, these probabilistic models cannot benefit from the word embeddings which are continuous vectors. Faced with this, we propose to use the neural topic model to leverage the advantages of both word embeddings and probabilistic models. Category labels can play an important role in the clustering procedures. Inspired by this, leveraging both label information and embeddings to enhance the discovery performance has attracted our attention. As a result, a label-aided neural topic model derived from Gaussian LDA which integrates both word embeddings and external label information is proposed.

3. The Discovery Process of the Proposed Approach

As is shown in Figure 1, the service discovery process of the proposed model consists of four major parts: service preprocessing, service modelling, query modelling, and service ranking. As shown in Figure 1, the Web service is firstly crawled and preprocessed. The description and the label of a Web service are extracted from the collected materials. Then, the service descriptions are taken as the input of the word2vec model to create the word embeddings. After getting the word embeddings, we map the words in the service description label into word embedding to produce one input and take the Web service label as the other input for the proposed model LNTM. The LNTM will convert each Web service into representations of latent factors. To model users' queries, the words in a query are looked up from embeddings and mapped into embeddings. In the service ranking phase, based on LNTM and users' queries, a probabilistic service ranking model is proposed to retrieve relevant services for the users.

In the proposed approach, training word embedding and modelling services are conducted offline, and the efficiency of the proposed discovery model can be guaranteed. Hence, the focus of the approach will be placed on the accuracy of discovery.

3.1. LNTM. For capturing semantic regularities in the language and handling the semantic sparse of Web services, an augmented topic model with word embeddings for Web service discovery is proposed in [31]. In the meanwhile, labels of documents can be used to guide topic learning so as to find more meaningful topics [12]. Therefore, in this paper, a label-augmented neural topic model is proposed to leverage label information and capture semantic regularities for enhancing discovery performance of the Web service in this paper.

In the proposed model LNTM, the word embedding v for each term in a document d at position i is written as $v_{di} \in R^W$, and W is the length of the word embedding. As a result, the words in a document are mapped into continuous vectors in the W-dimensional space. Therefore, each topic k is characterized as a multivariate Gaussian distribution with mean μ_k and covariance Σ_k. The Gaussian parameterization is determined by both analytic convenience and the semantic similarity of embeddings. To govern the mean and variance of each Gaussian, the Gaussian distribution centered at zero and an inverse Wishart distribution for the covariance are placed as the conjugate priors.

Similar to Gaussian LDA, Web service modeled by LNTM is represented as the mixtures over latent topics with proportions drawn from a Dirichlet prior.

To integrate labels, words are indicators for the presence of labels, and then l_d would include 1 in the positions for each label listed on document d and 0, otherwise. The graphical representation of LNTM is shown in Figure 2. Based on above notions, the generative process of LNTM for a document can be summarized as follows:

(1) For topic $k = 1, \ldots, K$,
   (a) For each label $l = 1, \ldots, L$, choose $\lambda_{l,k} \sim \mathrm{Ga}(s, s)$
   (b) Draw topic covariance $\Sigma_k \sim W^{-1}(\Psi, \nu)$
   (c) Draw topic mean $\mu_k \sim \mathrm{Normal}(\mu, (1/k)\Sigma_k)$
(2) For each document d in corpus D,
   (a) For each topic k, compute $\alpha_{d,k} = \prod_l^{L_{\mathrm{doc}}} \lambda_{l,k}^{f_{d,l}}$
   (b) Draw topic distribution $\theta_d \sim \mathrm{Dir}(\alpha)$
   (c) For each word index $i = 1, \ldots, N_d$,
      (i) Draw a topic $z_n \sim \mathrm{Cat}(\theta_d)$
      (ii) Condition on z_{di}, draw a word v_{di} with a probability $v_{di}/z_{di}, u_{1,\ldots,K}, \Sigma_{1,\ldots,K} \sim \mathrm{Normal}(\mu_{z_{di}}, \Sigma_{z_{di}})$

3.2. Web Service Modelling Using LNTM. The LNTM is a generative model in which each embedding v in a service description is associated with the latent variable topic z, and each topic z is associated with the service description d. With these two distributions, a Web service can be expressed as two layers: the service topics and the topic embeddings.

After using the LNTM, the service-topic distribution is achieved by the parameter θ ($\theta \in |\text{services}| \times |\text{topics}|$), and topic embedding is achieved by the multivariate Gaussians.

To infer the topic assignments of individual embeddings and the posterior distribution of services over the topics, a collapsed Gibbs sampling method is adopted to derive the topic assignments to each embedding by using the update rule shown in the following equation:

$$p(z_{di} = k, \lambda_{di} = l \mid z_{-di}, l_{-di}, V_d, \zeta, \alpha) \propto \left(n_{l_{di} z_{di}} + \alpha_{z_{di}}\right) \times t_{\nu_k - M + 1}\left(v_{di} \mid \mu_k, \frac{\kappa_k + 1}{\kappa_k}\Sigma_k\right), \quad (1)$$

where z_{-di} represents the topic assignments of all word embeddings, excluding the one at the i-th position of service$d$. λ_{-di} represents the label assignments. V_d is the sequence of vectors for service description d; M is the length of the word embedding; a tuple $\zeta = (\mu, \kappa, \Sigma, \nu)$ is the parameters of the prior distribution; and $t_{\nu_\prime}(x \mid \mu', \Sigma')$ is the multivariate t-distribution with freedom degree $\nu\prime$ and parameters μ' and Σ'.

Note that the first part of equation (1) which expresses the probability of topic k in service description d is derived as that of Gaussian LDA.

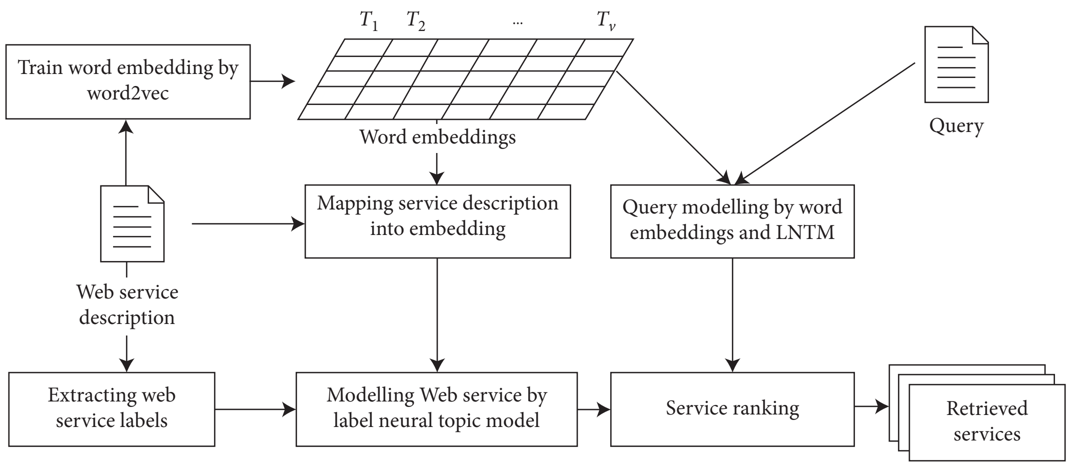

Train word embedding by word2vec
T_1 T_2 ... T_v
Word embeddings
Query
Mapping service description into embedding
Query modelling by word embeddings and LNTM
Web service description
Extracting web service labels
Modelling Web service by label neural topic model
Service ranking
Retrieved services

FIGURE 1: The discovery process.

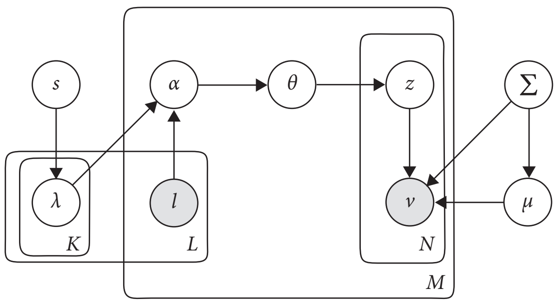


FIGURE 2: Graphical model of LNTM.

The second part of equation (1) expresses the probability of assignment of topic k to the vector v_{di} given the current topic assignments which represented by a multivariate t-distribution with parameters $(\mu_k, \kappa_k, \Sigma_k, v_k)$. These parameters for the posterior distribution are calculated by equation (2) as the Gaussian LDA:

$$\begin{aligned}
\kappa_k &= \kappa + N_k,\\
\mu_k &= \frac{\kappa\mu + N_k\overline{v}_k}{\kappa_k},\\
v_k &= v + N_k,\\
\sigma_k &= \frac{\Psi_k}{v_k - M + 1},\\
\Psi_k &= \Psi + C_k + \frac{\kappa N_k}{\kappa_k}\left(\overline{v}_k - \mu\right)\left(\overline{v}_k - \mu\right)^{\top}.
\end{aligned} \tag{2}$$

Here, the parameters $\overline{v}_k$ and C_k are calculated as

$$\begin{aligned}
\overline{v}_k &= \frac{\sum_d \sum_{i:z_{di}} v_{di}}{N_k},\\
C_k &= \sum_d \sum_{i:z_{di}=k} \left(v_{di} - \overline{v}_k\right)\left(v_{di} - \overline{v}_k\right)^{\top},
\end{aligned} \tag{3}$$

where N_k is the total counts of the words of the topic assignment of k across all descriptions. $\overline{v}_k$ and C_k are the sample mean and the scaled form of sample covariance assigned topic k, respectively. Intuitively, the parameters μ_k and Σ_k are the posterior mean and covariance. The parameters κ_k and v_k denote the strength of the priors for mean and covariance, respectively. After getting these parameters, we can simply achieve the topic-embedding distribution as discussed above.

3.3. Query Modelling and Ranking. To retrieve relevant services by the proposed model, we firstly translate the user query into embeddings. The words in a query are extracted and mapped into the embeddings by looking up the embedding features.

To rank the retrieved Web service, we use the generated probabilities to calculate the similarity between the user queries and the Web services as the work in [31]. The similarities are represented by $P(Q \mid s_i)$, where Q is the query and s_i is the i-th Web service. Thus, using the assumptions of the LNTM described above, it can be calculated by the following equation:

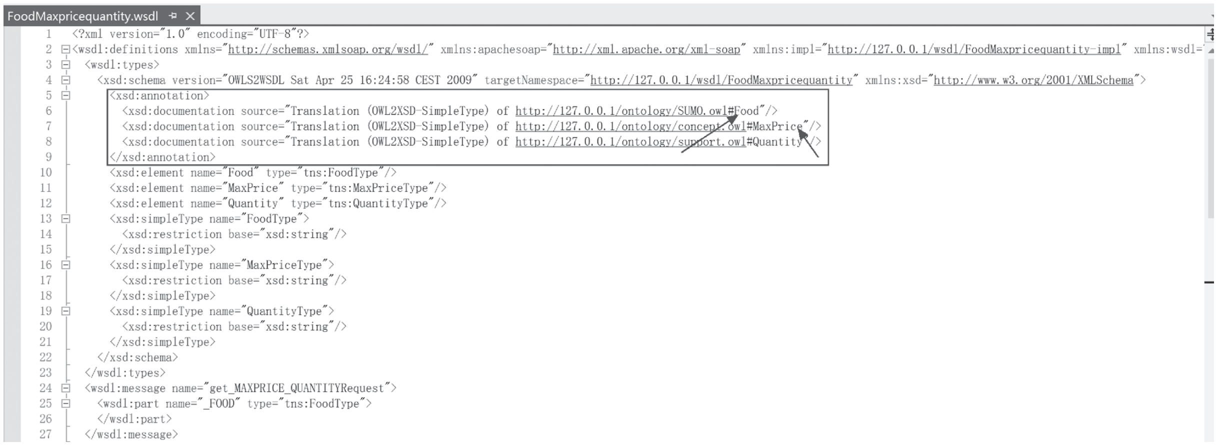
$$P\left(Q \mid s_i\right) = \prod_{e\in Q} P\left(e \mid s_i\right) = \prod_{e\in Q}\sum_{z=1}^{K} P\left(e \mid z\right)P\left(z \mid s_i\right). \tag{4}$$

Here, $P(e \mid z)$ and $P(z \mid s)$ are the posterior probabilities computed according to above equation (2) and the matrix θ, respectively. Finally, we can obtain a list of retrieved services towards a query according the value of $P(Q|s_i)$.

4. Experiment Setting

To evaluate the proposed approach, we conducted several experiments on the standard Web service test dataset SAWSDL-TC3 (TC3) (http://www.semwebcentral.org/projects/sawsdl-tc) as Tian et al. [31] did. To use TC3, we first parse the WSDL files into a plain text and then removed stop words and lemmatized the remaining words.

As is known to all, the Web services of TC3 do not have explicit category labels. However, there are some implicit categories in the WSDL files. As shown in Figure 3, the node "<xsd:annotation>" of service "FoodMaxpricequantity.wsdl"



```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://127.0.0.1/wsdl/FoodMaxpricequantity-impl" xmlns:wsdl="
  <wsdl:types>
    <xsd:schema version="OWLS2WSDL Sat Apr 25 16:24:58 CEST 2009" targetNamespace="http://127.0.0.1/wsdl/FoodMaxpricequantity" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:annotation>
        <xsd:documentation source="Translation (OWL2XSD-SimpleType) of http://127.0.0.1/ontology/SUMO.owl#Food"/>
        <xsd:documentation source="Translation (OWL2XSD-SimpleType) of http://127.0.0.1/ontology/concept.owl#MaxPrice"/>
        <xsd:documentation source="Translation (OWL2XSD-SimpleType) of http://127.0.0.1/ontology/support.owl#Quantity"/>
      </xsd:annotation>
      <xsd:element name="Food" type="tns:FoodType"/>
      <xsd:element name="MaxPrice" type="tns:MaxPriceType"/>
      <xsd:element name="Quantity" type="tns:QuantityType"/>
      <xsd:simpleType name="FoodType">
        <xsd:restriction base="xsd:string"/>
      </xsd:simpleType>
      <xsd:simpleType name="MaxPriceType">
        <xsd:restriction base="xsd:string"/>
      </xsd:simpleType>
      <xsd:simpleType name="QuantityType">
        <xsd:restriction base="xsd:string"/>
      </xsd:simpleType>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="get_MAXPRICE_QUANTITYRequest">
    <wsdl:part name="_FOOD" type="tns:FoodType">
    </wsdl:part>
  </wsdl:message>
```

Figure 3: The category labels for a service.

has values of "#Food," "#MaxPrice," and "Quantity." As a result, we extracted these values to generate the category labels for each Web service in our experiments. Since the Web services in the real-world service registry all belong to their certain categories, it is easily to collect the category label information for using the proposed approach.

In our experiments, we used precision p, recall r, and F-measure f as the evaluation criterion which is defined in equation (5) for the proposed approach. The larger the F-measure is, the better the performance of the discovery is.

$$p = \frac{|\text{relevant} \cap \text{predicted}|}{|\text{predicted}|},$$

$$r = \frac{|\text{relevant} \cap \text{predicted}|}{|\text{relevant}|}, \tag{5}$$

$$f = 2 \cdot \frac{p \cdot r}{p + r},$$

where relevant is the relevant class labels and predicted is the predicted results of the classification methods.

4.1. Performance of the Proposed Approach. To examine the performance of our approach, we compare the proposed method with three other Web service discovery approaches. These approaches are demonstrated as follows:

(1) LDA: when using LDA, the latent factors learnt from the Web service description are adopted to represent the Web service, and then a discovery approach is used to rank the services [30].

(2) Meta-LDA: in Meta-LDA [12], metainformation such as a category of a Web service is directly incorporated into the generative process. The external metainformation can improve the topic quality and modelling accuracy. We use Meta-LDA to group Web services into different clusters and then employ a probabilistic model to rank the services.

(3) Gaussian – LDA: a Gaussian LDA-based Web service discovery approach which makes use of embeddings for semantic sparsity Web service discovery is conducted based on the work done by Tian at al. [31].

(4) LNTM: for LNTM, we first train the word embeddings by word2vec from the prepared corpus. Then, we train the LNTM by incorporating embedding and service category labels into the generative process and organize the Web service into different clusters. Finally, we represent the query by embedding and utilize the probabilistic discovery model to rank the Web services as illustrated in Section 3.

For LDA and the Meta-LDA model, following the modelling process mentioned above, the topics are generated from the descriptions of the Web services. Then, we tuned the algorithms, respectively, to their best parameter settings by cross validation.

Table 1 shows the experimental data on TC3. According to these experiments, we have several observations: firstly, LNTM outperformed all the competitors in terms of F-measure on nearly all the settings, showing the benefit of using both word embeddings and service category labels which demonstrates the effectiveness of the proposed model.

Secondly, by looking at the approaches using the label information, we can see the significant improvement of these models over LDA, which indicates that document labels can play an important role in guiding topic modelling.

Thirdly, the LNTM and Gaussian – LDA have better performance than the LDA-based method. The results show that the embedding-based approach which takes continuous embeddings as the input may capture more semantically coherent topics compared to the traditional LDA-based method.

Finally, it is interesting to note that the Meta-LDA outperforms LDA and LNTM outperforms Gaussian – LDA, respectively, in this study. These findings are in agreement with the idea that utilizing the category label data of Web services improves the performance of Web service discovery. These results inspire the research work to integrate other external information for effective Web service discovery.

Table 1: Performance of the proposed approach.

Query	LDA			Meta-LDA			Gaussian LDA			LNTM		
	p	r	f	p	r	f	p	r	f	p	r	f
@10	0.64	0.30	0.40	0.78	0.41	0.54	0.76	0.37	0.50	0.91	0.46	0.61
@15	0.57	0.35	0.43	0.80	0.39	0.52	0.69	0.43	0.53	0.82	0.55	0.65
@20	0.50	0.38	0.44	0.74	0.42	0.53	0.61	0.47	0.53	0.75	0.58	0.65
@25	0.45	0.31	0.43	0.69	0.39	0.49	0.58	0.51	0.54	0.71	0.63	0.66
@30	0.41	0.44	0.43	0.63	0.49	0.55	0.55	0.54	0.54	0.69	0.67	0.67
@35	0.38	0.46	0.42	0.67	0.53	0.59	0.51	0.59	0.55	0.65	0.72	0.68
@40	0.36	0.49	0.42	0.61	0.59	0.60	0.49	0.61	0.54	0.63	0.73	0.67

4.2. Validation of Labels. To validate that incorporating category label information can significantly improve the generative topic accuracy, we varied the proportion of services used in training from 20% to 80% and used the remaining for testing. Here, we utilize normalised pointwise mutual information (NPMI) as shown in equation (6) to evaluate the topic quality of LDA, Meta-LDA, and LNTM:

$$\mathrm{NPMI}(k) = \Sigma_{j=2}^{T}\sigma_{i=1}^{j-1}\left(\log\frac{p(w_i, w_j)}{p(w_i)p(w_j)} - \log\big(p(w_i, w_j)\big)\right). \tag{6}$$

The NPMI score of each topic in the experiments is calculated with top 15 words ($T = 15$). As shown in Figure 4, the NPMI scores of both LNTM and Meta-LDA outperform LDA. The result indicates that the label information can enhance the LDA-based model to find more meaningful topics. The details of the two corpus are shown in Table 2.

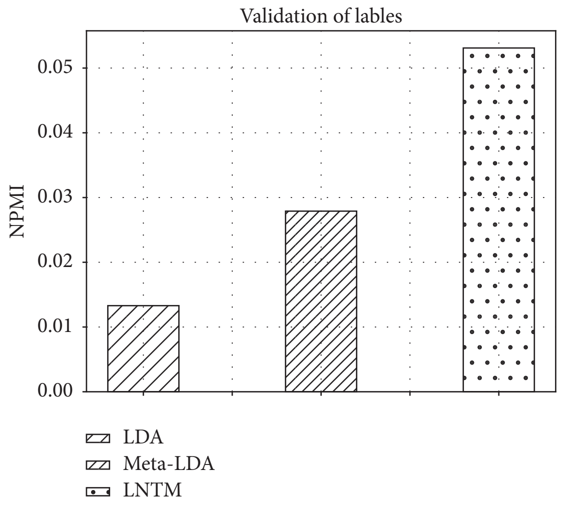


Figure 4: The influence of labels.

4.3. Validation of Embedding. As discussed above, embodying more semantic knowledge by changing the Bag of Words model into the continuous embedding space using LNTM can enhance the performance of the Web service discovery model. Several experiments are conducted so as to validate the result. Figure 5 shows the F-measure performance of the proposed approach with different word embeddings trained by the word2vec model using different corpus TC3 and Wikipedia.

As shown in Figure 5, the proposed approach using TC3 has better F-measure performance than using Wikipedia. The possible explanation for this may be that some words extracted from the WSDL files which do not have enough appearance counts in the Wikipedia corpus are removed when training the embeddings though they are very informative [31].

Table 2: Statistic of word embeddings.

	TC3	Wikipedia
Words	6,895	8,069,236
Documents	1,043	3,758,076
Embeddings	50	50

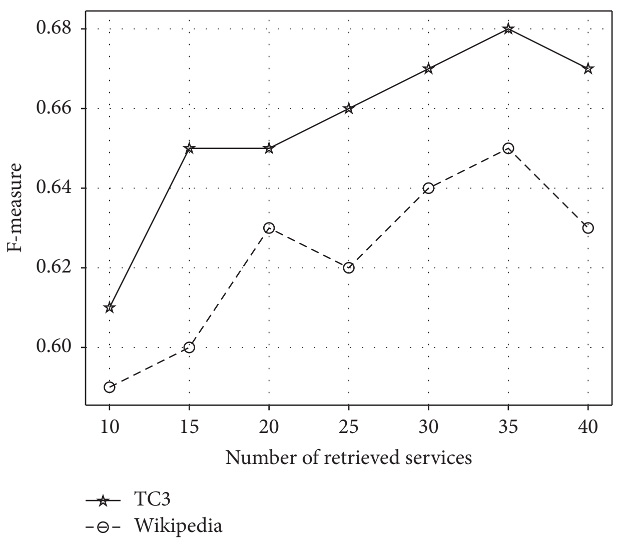


Figure 5: The influence of different embedding sets.

4.4. Influence of Hyperparameters. In LNTM, the parameter α illustrates the weight of language model contribution, μ and Σ control the document contribution, while s contributes to the label information. In our work, hyperparameters are empirically set as $\alpha = 1/K$, $s =$ zero, $\mu =$ zero mean, $\Sigma = 3 * I$, and 1,000 sampling iterations as in work [31]. Here, K is the number of topics, and I is the identity matrix. To check the influence of topic number k, we calculated $P(e \mid k)$ for different k. As shown in Figure 6, the

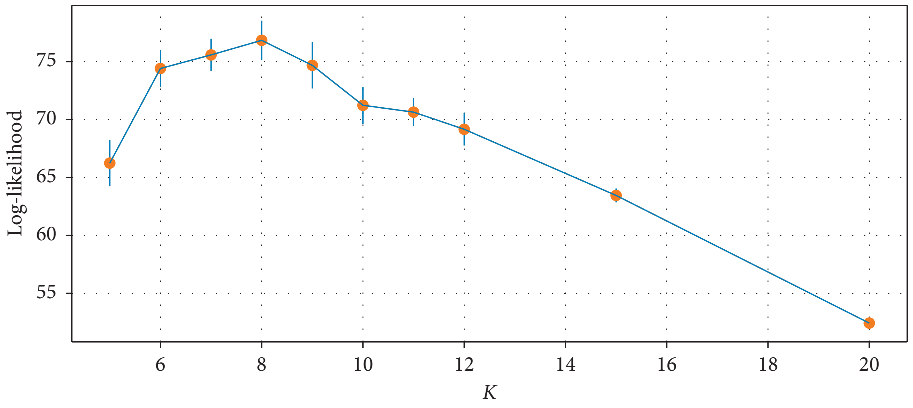


Figure 6: The number k of topics in LNTM.

result suggests that the data are best accounted for the proposed LNTM model incorporating 8 topics.

5. Conclusion

In this paper, we proposed a Web service discovery approach that combines word embeddings and category labels to deal with the poor recall problem in searching semantic sparse Web services. We used word embeddings to map the word into embedding so as to enrich the Web service semantics. We also introduced a label-augmented neural topic model LNTM which organizes the Web services into hierarchies for a probabilistic ranking approach.

Several experiments were conducted on a widely used dataset TC3 to validate the performance of our approach. Experimental results suggested that the proposed approach is feasible, and in particular, the word embeddings and label information both lead to enhanced performance in the Web service discovery process.

Since not all the Web services have their category labels, it is necessary here to clarify exactly how to conduct effective Web service discovery without labels. In the future, there is abundant room to further investigate the usefulness of various metainformation of Web service and propose different forms based on Gaussian – LDA to provide effective service discovery.

Data Availability

The experiments' data are available in http://www.semwebcentral.org/projects/sawsdltc.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under grant nos. 61702305, 11971270, and 61903089, the China Postdoctoral Science Foundation under grant no. 2017M622234 and Science and Technology Support Plan of Youth Innovation Team of Shandong higher School under grant no. 2019KJN2014.

References

[1] M. Xu, W. Tian, and R. Buyya, "A survey on load balancing algorithms for virtual machines placement in cloud computing," *Concurrency & Computation Practice & Experience*, vol. 29, no. 12, p. e4123, 2017.

[2] D. He, X. Yang, Z. Feng, S. Chen, and F. Fogelman-Soulie, "A probabilistic model for service clustering - jointly using service invocation and service characteristics," In Proceedings of the 2018 IEEE International Conference on Web Services (ICWS), San Francisco, CA, USA, July 2018.

[3] K. Elgazzar, E. H. Ahmed, and P. Martin, "Clustering wsdl documents to bootstrap the discovery of web services," in *Proceedings of the 2010 IEEE International Conference Web Services*, pp. 147–154, IEEE, Miami, FL, USA, July 2010.

[4] G. Tian, P. Liu, Y. Peng, and C. Sun, "Tagging augmented neural topic model for semantic sparse web service discovery," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 16, p. e4448, 2018.

[5] G. Tian, S. Zhao, J. Wang, Z. Zhao, J. Liu, and L. Guo, "Semantic sparse service discovery using word embedding and Gaussian lda," *IEEE Access*, vol. 7, pp. 88231–88242, 2019.

[6] W. Chen, I. Paik, and P. C. K. Hung, "Constructing a global social service network for better quality of web service discovery," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 284–298, 2015.

[7] L. De Jesus Silva, D. Barreiro Claro, and D. Cicero Pavão Lopes, "Semantic-based clustering of web services," *Journal of Web Engineering*, vol. 14, no. 3-4, pp. 325–345, 2015.

[8] O. Jin, N. N. Liu, K. Zhao, Y. Yu, and Q. Yang, "Transferring topical knowledge from auxiliary long texts for short text clustering," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pp. 775–784, ACM, Glasgow, Scotland, October 2011.

[9] S. Seifzadeh, K. F. Ahmed, M. S. Kamel, and F. Karray, "Short-text clustering using statistical semantics," in *Proceedings of the 24th International Conference on World Wide Web Companion*, pp. 805–810, Florence, Italy, May 2015.

[10] T. Mikolov, W.-T. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proceedings of the HLT-NAACL*, pp. 746–751, Atlanta, GA, USA, June 2013.

[11] E. Amine, J. Flocon-Cholet, S. Gosselin, and S. Vaton, "Bayesian mixture models for semi-supervised clustering,"

2019, https://hal.archives-ouvertes.fr/hal-02372337/file/Bayesian_Mixture_Models_For_SemiSupervised_Clustering.pdf.

[12] H. Zhao, D. Lan, W. Buntine, and G. Liu, "Metalda: a topic model that efficiently incorporates meta information," 2017, https://arxiv.org/abs/1709.06365.

[13] R. Das, M. Zaheer, and C. Dyer, *Gaussian LDA for Topic Models with Word Embeddings*, Vol. 1, Long Papers, Beijing, China, 2015.

[14] L. D. Ngan and R. Kanagasabai, "Semantic web service discovery: state-of-the-art and research challenges," *Personal and Ubiquitous Computing*, vol. 17, no. 8, pp. 1741–1752, 2013.

[15] P. Rodriguez Mier, C. Pedrinaci, M. Lama, and M. Mucientes, "An integrated semantic web service discovery and composition framework," 2015, https://arxiv.org/pdf/1502.02840.pdf.

[16] L. Chen, Y. Wang, Yu Qi, Z. Zheng, and J. Wu, "Wt-lda: user tagging augmented lda for web service clustering," in *Service-Oriented Computing*, pp. 162–176, Springer, Berlin, Germany, 2013.

[17] X. Hu, N. Sun, C. Zhang, and T.-S. Chua, "Exploiting internal and external semantics for the clustering of short texts using world knowledge," in *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pp. 919–928, ACM, Hong Kong, China, November 2009.

[18] Y.-T. Ma, K.-Q. He, B. Li, J. Liu, and X.-Y. Zhou, "A hybrid set of complexity metrics for large-scale object-oriented software systems," *Journal of Computer Science and Technology*, vol. 25, no. 6, pp. 1184–1201, 2010.

[19] W. Pan, B. Li, J. Liu, Y. Ma, and B. Hu, "Analyzing the structure of Java software systems by weightedK-core decomposition," *Future Generation Computer Systems*, vol. 83, no. 1, pp. 431–444, 2018.

[20] I. Şora and C.-B. Chirila, "Finding key classes in object-oriented software systems by techniques based on static analysis," *Information and Software Technology*, vol. 116, no. 1, pp. 75–89, 2019.

[21] W. Pan and C. Chai, "Structure-aware mashup service clustering for cloud-based internet of things using genetic algorithm based clustering algorithm," *Future Generation Computer Systems*, vol. 87, pp. 267–277, 2018.

[22] W. Pan and C. Chai, "Measuring software stability based on complex networks in software," *Cluster Computing*, vol. 22, no. 2, pp. 2589–2598, 2019.

[23] W. Pan, H. Jiang, H. Ming, C. Chai, B. Chen, and H. Li, "Characterizing software stability via change propagation simulation," *Complexity*, vol. 2019, Article ID 9414162, 17 pages, 2019.

[24] W. Pan, H. Ming, C. Chang, Z. Yang, and D.-K. Kim, "Elementrank: ranking java software classes and packages using multilayer complex network-based approach," *IEEE Transactions on Software Engineering*, 2019.

[25] Y. Xiang, W. Pan, H. Jiang, Y. Zhu, and H. Li, "Measuring software modularity based on software networks," *Entropy*, vol. 21, no. 4, p. 344, 2019.

[26] S. Zhou and Y. Wang, "Clustering services based on community detection in service networks," *Mathematical Problems in Engineering*, vol. 2019, Article ID 1495676, 11 pages, 2019.

[27] W. Pan, B. Song, K. Li, and K. Zhang, "Identifying key classes in object-oriented software using generalizedk-core decomposition," *Future Generation Computer Systems*, vol. 81, no. 1, pp. 188–202, 2018.

[28] W. Pan, J. Dong, K. Liu, and J. Wang, "Topology and topic-aware service clustering," *International Journal of Web Services Research*, vol. 15, no. 3, pp. 18–37, 2018.

[29] T. Kenter and M. de Rijke, "Short text similarity with word embeddings," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 1411–1420, ACM, Melbourne, Australia, October 2015.

[30] C. Gilbert, P. Barnaghi, and K. Moessner, "Probabilistic methods for service clustering," in *Proceeding of the 4th International Workshop on Semantic Web Service Matchmaking and Resource Retrieval, Organised in Conjonction the ISWC*, Citeseer, Guildford, UK, 2010.

[31] G. Tian, J. Wang, Z. Zhao, and J. Liu, "Gaussian LDA and word embedding for semantic sparse web service discovery," *Collaborate Computing: Networking, Applications and Worksharing*, Springer, Berlin, Germany, pp. 48–59, 2017.

Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 3428604, 20 pages
https://doi.org/10.1155/2020/3428604

Research Article

Method of Coupling Metrics for Object-Oriented Software System Based on CSBG Approach

Aihua Gu,[1] Lu Li,[1] Shujun Li,[1] Qifeng Xun,[1] Jian Dong,[1] and Jianhong Lin[2]

[1]*School of Information Engineering, Yancheng Teachers University, Yancheng 224002, China*
[2]*Zhejiang Ponshine Information Technology Co., Ltd., Hangzhou 310012, China*

Correspondence should be addressed to Aihua Gu; guaihua1978@163.com

Received 2 November 2019; Accepted 5 February 2020; Published 19 March 2020

Guest Editor: Weifeng Pan

Context. Coupling between classes is an important metric for software complexity in software systems. *Objective*. In order to overcome the shortcomings of the existing coupling methods and fully investigate the weighted coupling of classes in different cases in large-scale software systems, this study analyzed the relationship between classes at package level, class level, and method level. *Method*. The software system is considered as a set of special bipartite graphs in complex networks, and an effective method for coupling measurement is proposed as well. Furthermore, this method is theoretically proved to satisfy the mathematical properties of coupling measurement, leading to overcome the disadvantages of the majority of existing methods. In addition, it was revealed that the proposed method was efficient according to the analyses of existing methods for coupling measurement. Eventually, an algorithm was designed and a program was developed to calculate coupling between classes in three open-source software systems. *Results*. The results indicated the scale-free characteristic of complex networks in the statistical data. Additionally, the calculated power-law value was used as a metric for coupling measurement, so as to calculate coupling of the three open-source software. It indicated that coupling degrees of the open-source software systems contained a certain impact on evaluation of software complexity. *Conclusions*. It indicated that coupling degrees of the open-source software systems contained a certain impact on evaluation of software complexity. Moreover, statistical characteristics of some complex networks provided a reliable reference for further in-depth study of coupling. The empirical evidence showed that within a certain range, reducing the coupling was helpful to attenuate the complexity of the software, while excessively blindly pursuit of low coupling increases the complexity of software systems.

1. Introduction

Coupling refers to the degree of interdependence between software modules; a measure of how closely connected two routines or modules are [1]; and the strength of the relationships between modules. Structured design, including cohesion and coupling, was published in an article by Stevens et al. and a book by Stevens et al. [2, 3], and the latter subsequently became standard terms. Coupling is considered as a double-edged sword in object-oriented programming. On the one hand, object-oriented software development (OOSD) includes object-oriented requirement analysis, as well as object-oriented design. OOSD is a practical method of developing a software system which focuses on the objects of a problem throughout development. Interactions between objects reflect the interdependence between objects. If objects are isolated, then the software system can only achieve simple functions. However, objects are equivalent to cells in human body. If cells are completely isolated from human body, they basically do not play any significant role, reflecting that functions of a software system require a tight coupling between objects. On the other hand, tight coupling between objects would lead to a water-wave effect, meaning that changes in one object may result in further changes in other objects. The most terrible case is that there is a possibility of “avalanche” effect, which may affect the whole system, leading to a sharp decline in the testability, understandability, reliability, and maintainability of the system. Therefore, it is expected that classes are loosely coupled in terms of software design. A system can be tightly coupled in one aspect while being loosely coupled in another. However, software developers mainly prefer to develop those systems that are as loosely coupled as possible; thus, design, testing, and maintenance of the system would be relatively independent and more reasonable. Moreover, a decrease may be observed in the possibility that errors propagate between modules if there are few connections between modules [4]. Coupling has been widely used in evaluation of the degree of failure in classification [5–7], effective analysis [8, 9], and design pattern [10] of software systems.

The present article has the following organization: Section 2 summarizes the materials and methods. Section 3 shows the results. Section 4 provides a conclusion and suggests perspectives.

2. Materials and Methods

2.1. Methods. Currently, the methods for coupling measurement of object-oriented software systems are mainly structure-oriented measurement methods (Tables 1 and 2) [8, 11–15].

Comparative analysis of typical methods is shown in Tables 1–3, indicating that

(1) Methods for coupling measurement are mainly based on method invocation between classes.

(2) Calculation of coupling strength is defined as the degree of method invocation, which is weighted coupling.

(3) A small number of methods use fan-in and fan-out as metrics.

(4) Inheritance is dominant.

(5) Few methods use static method invocation, system measurement, and package-level metrics.

In addition to the abovementioned structural information methods, some scholars have recently used dynamic information methods [17, 18], semantic information methods [19–21], and logical information methods [22, 23]. Based on the results of previous studies, methods of coupling measurement cover the following cases:

(1) The DCMs are more accurate than that of structural information methods, while it seems to be difficult in the measurement of coupling metrics. However, structural information methods are more intuitive and easier to be perceived compared with semantic information and logical information methods.

(2) At present, the majority of the traditional structural information methods analyze coupling based on the degree of connecting edges between classes and mainly focus on complexity between classes and emphasize more on measurement from a local fine-grained aspect. Moreover, these degrees of connecting edges only consider a certain or a limited aspects of software engineering. Therefore, these methods contain some limitations, which cannot properly satisfy the requirement of an effective coupling measurement in complex software systems.

(3) Although a number of coupling measurement methods analyze network relationship from overall and macro perspectives based on graph theory, the majority of measurement metrics mainly use classes, packages, or methods as nodes to construct some undirected, directed, unweighted, or weighted network models. Moreover, they ignore a complex relationship of object-oriented characteristics between different classes. Some methods have not been theoretically verified for developing the mathematical characteristics of the measurement metrics.

The process of establishment of an effective method for coupling measurement between classes in a software system is determined by the following two aspects: reasonable measurement metrics and theoretical support of measurement metrics [24, 25]. Briand et al. mathematically analyzed measurement metrics of the software system and presented a robust theoretical support for the measurement metrics [4, 26, 27]. Many of complex networks have been shown to share the features such as "scale-free" and "small world" [28, 29]. Pan et al. revealed many physics-like laws in software systems from a complex network perspective recently [30, 31]. Studies on complex networks and software engineering revealed that class-level, method-level, and package-level diagrams of a software system could show the characteristics of "scale-free" and "small world," which provided a novel perspective for finding more reasonable measurement metrics [32–34]. Complex network theories were applied to measure software [35, 36], identify key software elements [37], and cluster Web services [38–39]. Researchers have found that many real networks have the bipartite graph characteristics of complex networks [40–45]. With combination of package level, class level, and method level, this study analyzed a complex relationship between classes in the same layer and all layers of a package and proposed a method for coupling metrics for object-oriented systems based on bipartite graph of complex networks, named here CSBG, and object-oriented software systems were expressed as a set of special bipartite graphs.

2.2. Problem Description. In this study, a statistical method for complex networks was used to analyze the degree of fan-out and the heterogeneity of classes at the same layer and all layers of a package, in addition to the calculation of coupling of software systems.

2.2.1. Relationship between Classes

Definition 1. ASS relationship (association and aggregation).

Association means which/how classes interact with each other, and association can be represented by a line between these classes with an arrow indicating the navigation

Table 1: The first part of existing methods for coupling measurement.

Method	Description
CBO [11]	$\mathrm{CBO}(c) = \lvert\{d \in C - \{c\} \mid \mathrm{uses}(c,d) \vee \mathrm{uses}(c,d)\}\rvert$; the metric is 1, if method in one class invokes other classes or is attributed to another class, otherwise it is 0
CBO′ [12]	$\mathrm{CBO}'(c) = \lvert\{d \in C - (\{c \cup \mathrm{Ancestors}(C)\}) \mid \mathrm{uses}(c,d) \vee \mathrm{uses}(c,d)\}\rvert$; this is similar to CBO method; however, that does not consider inheritance
RFC [13]	$\mathrm{RFC}(c) = \mathrm{RFC}_1(c)$, which is used for calculating the number of methods responding to an object's message
RFCα [13]	$R_0(c) = M(c)$, $R_{i+1}(c) = \cup_{m \in R_i(c)} \mathrm{PIM}(m)$, that is, a set of polymorphic methods invoked by functions in set $R_i(c)$; then $\mathrm{RFC}_\alpha(c) = \lvert\cup_{i=0}^{\alpha} R_i(c)\rvert$ with $\alpha = 1, 2, 3, \ldots,$
RFC′ [13]	$\mathrm{RFC}'(c) = \mathrm{RFC}_\infty(c)$
MPC [13]	$\mathrm{MPC}(c) = \sum_{m \in M_I(c)} \sum_{m' \in \mathrm{SIM}(m) - M_I(c)} \mathrm{NSI}(m, m')$; this calculates the number of static method invocation of classes
DAC [14]	$\mathrm{DAC}(c) = \lvert\{a \mid a \in A_I(c) \vee T(a) \in C\}\rvert$
DAC′ [14]	$\mathrm{DAC}'(c) = \lvert\{T(a) \mid a \in A_I(c) \vee T(a) \in C\}\rvert$; this formula is similar to DAC; however, if there is a relationship between classes, the metric is 1, otherwise the metric is 0

Table 2: The second part of existing methods for coupling measurement.

Method	Description
COF [14]	$\mathrm{COF}(C) = (\sum_{c \in C} \lvert\{d \mid d \in C - \{c\} \cup \mathrm{Ancestors}(c) \wedge \mathrm{uses}(c,d)\}\rvert) / \lvert C\rvert^2 - \lvert C\rvert - 2\sum_{c \in C} \mathrm{Descendent}(c)$
ICP [14]	This method calculates the parameters invoked by the method in a weighted class
IH-ICP [14]	This is similar to ICP, however, that only considers inheritance
NIH-ICP [14]	This is similar to ICP, however, that does not consider inheritance
SIMAS [8]	This method calculates the number of direct or indirect invocations between static methods of two different classes
PIM [8]	This method calculates the number of invocations in class C of methods in class D, and polymorphism is considered
PIMAS [8]	This method calculates the number of direct or indirect invocations between class methods, and polymorphism is taken into account
INAG [8]	The metric is 1 if there is an indirect aggregation between two classes; otherwise, the metric is 0
ACAIC [15]	$\mathrm{ACAIC}(c) = \sum_{d \in \mathrm{Ancestors}(c)} \mathrm{CA}(c,d)$; this calculates the number of out-degrees between one class and attributes of another classes in two classes with inheritance
OCAIC [15]	$\mathrm{OCAIC}(c) = \sum_{d \in \mathrm{Others}(c) \cup \mathrm{Friends}(c)} \mathrm{CA}(c,d)$; it calculates the number of out-degrees between one class and attributes of another class in two classes without inheritance
ACMIC [15]	$\mathrm{ACMIC}(c) = \sum_{d \in \mathrm{Ancestors}(c)} \mathrm{CA}(c,d)$; it calculates the number of out-degrees between one class and methods of another class in two classes with inheritance
OCMIC [15]	$\mathrm{OCMIC}(c) = \sum_{d \in \mathrm{Others}(c) \cup \mathrm{Friends}(c)} \mathrm{CA}(c,d)$; it calculates the number of out-degrees between one class and methods of another class in two classes without inheritance
AMMIC [15]	$\mathrm{AMMIC}(c) = \sum_{d \in \mathrm{Ancestors}(c)} \mathrm{MM}(c,d)$; it calculates the number of out-degrees for methods between two classes with inheritance
OMMIC [15]	$\mathrm{OMMIC}(c) = \sum_{d \in \mathrm{Others}(c) \cup \mathrm{Friends}(c)} \mathrm{MM}(c,d)$
ICF, FCF [16]	$\mathrm{ICF}_i = \sum_{k=1}^{n} I(k,i)\mathrm{ICF}_k$, $I(i,j) = e(i,j) / \sum_{k=1}^{n} e(i,k)$, $\mathrm{FCF}_i = \sum_{k=1}^{n} F(k,i)\mathrm{FCF}_k$, $I(i,j) = e(i,j) / \sum_{k=1}^{n} e(k,j)$

direction. Aggregation implies that one class exists in another class in the form of attribute.

Definition 2. DEP relationship (dependency)

DEP_*D*: dynamic dependency refers to an instance method in a class that invokes methods and attributes in another class.

DEP_*S*: static dependency refers to static methods in a class invokes methods and attributes in another class.

Definition 3. GEN (generalization)

One class inherits with another class, or one class implements interfaces with another class, or that of an abstract class.

2.2.2. Definition of Package Hierarchy. Packages of an object-oriented software system include classes and subpackages in the current hierarchy, and these subpackages contain classes in the current hierarchy and their subpackages. Software systems can actually be considered to be a tree hierarchical structure composed of packages.

t layer of a package is defined as $p^t \le E^{t+1}, R^{t+1} >$. E^{t+1} represents a set of classes in the t layer, while this layer does not contain subpackages of this layer. R^{t+1} represents class relationship in the t layer, that is, $R^{t+1} \subseteq E^{t+1} \times E^{t+1}$.

S_layer(i) is defined as a set of weighted fan-out of C_i at the t layer of the package, that is, $S\_\mathrm{layer}(i) \subseteq R^{t+1}$. S_all(i) is the set of weighted cross-package fan-out of C_i, that is, $S\_\mathrm{all}(i) \subseteq R^1 \cup R^2 \cdots R^t \cdots$.

2.3. CSBG for Coupling Measurement. Software stability and modularity could be measured based on complex network theories. In this study, software systems can be expressed as a set of bipartite graphs that use nodes as classes, and ASS as well as DEP are the edges constituted by attributes of the class with those of another class based on complex network theories. However, GEN is a direct connection between

Table 3: Comparative analysis for the typical methods of coupling measurement.

Method	Type	Strength	Fan-out/ fan-in	Indirect coupling	Inheritance	Weighted	Static invocation	System metric	Package level
CBO	Method invocation, attribute reference	#coupled classed	Both	No	Both	No	No		
CBO′					Non-inh.-based	No	No		
RFC				No		Yes			
RFCα	Method invocation	#methods invoked	Import	Yes	Both	Yes	No	No	
RFC′				Yes		Yes			
MPC	Method invocation	#methods invocations	Import	No	Both	Yes	Yes		
DAC	Type of attribute	#attributes	Import	No	Both	Yes	No		
DAC′		#distinct types				No	No		
COF	Method invocation, attribute reference	#coupled classed	Both	No	Non-inh.-based	Yes	No	Yes	
ICP					Both	Yes	No		
IH-ICP	Method invocation	#methods invocations, #parameters passed	Import	No	inh.-based	Yes	No		No
NIH-ICP					Non-inh.-based	Yes	No		
SIMAS	Method invocation	#methods invocations		Yes	Both	Yes	Yes		
PIM	Method invocation	#methods invocations	Import	No	Both	Yes	No		
PIMAS	Method invocation	#methods invocations		Yes	Both	Yes	No		
INAG	Type of attribute	#attributes		Yes	Both	No	No		
ACAIC					inh.-based	Yes		No	
OCAIC	Type of attribute	#attributes			Non-inh.-based	Yes			
ACMIC					inh.-based	Yes			
OCMIC	Type of parameter	#of parameters	Import	No	Non-inh.-based	Yes	No		
AMMC					inh.-based	Yes			
OMMC	Method invocation	#method invocations			Non-inh.-based	Yes			
ICF	Method invocation, attribute reference	#method invocations	Import	No	Both	Yes	No	No	No
FCF			Export			Yes			

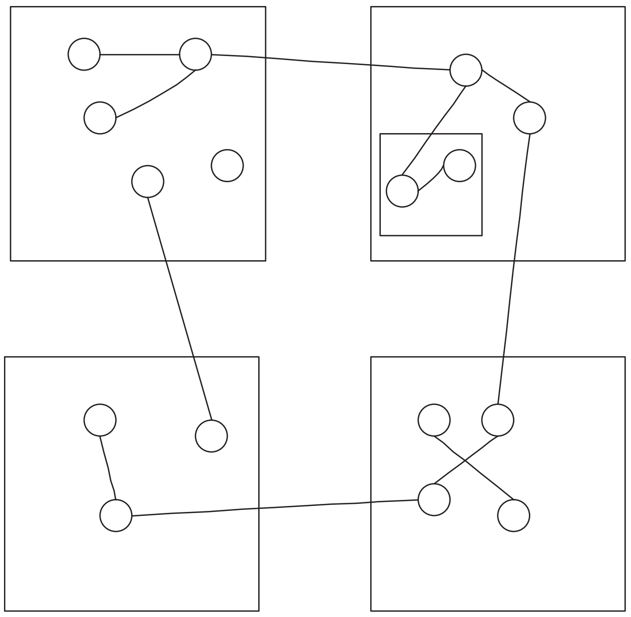


Figure 1: Illustration of a software system network composed of a set of special bipartite graphs (the large squares represent packages, and the 4 packages are at the same layer. The small squares show subpackages in the package. The circles denote classes in the package, and edges represent relationship between classes).

classes. Therefore, object-oriented software systems are taken into account as a set of special bipartite graphs constituted by classes in the package, as shown in Figure 1. There are defects in the coupling metrics containing the two metrics of fan-in and fan-out, because their total number is equal in a software system [46]. Therefore, this study only analyzed fan-out metric. The coupling strength between classes is correlated with the complexity of information exchange between modules. The more complex the information interaction (such as CBO), the tighter the coupling [47]. Coupling measurement metrics refer to the weighted fan-out of classes in special bipartite graphs. In these special bipartite graphs, classes associate with a class that may be at the same layer of the same package or at different layers of the package. Therefore, this study analyzed degrees of fan-out for classes in the same layer and all layers of the package. Moreover, heterogeneity of the abovementioned weighted out-degree was analyzed.

2.3.1. Demonstration of CSBG for Coupling Measurement. The detailed scheme proposed here is explained in the following, as illustrated in Figure 2:

(1) The object-orient software systems are constructed as directed weighted network graphs, and classes and relationship between classes are shown as nodes and edges, respectively.

(2) The package level, class level, and method level are combined to construct special weighted bipartite graphs between classes, aiming to make preparation for calculating the weighted out-degree of classes at the same layer of the package (see step 3, $(|\sum S\_\text{layer}(i)|)$), and the weighted out-degree $((|\sum S\_\text{all}(i)|))$ of all classes with classes across layers of the package (see step 3).

(3) ASS_layer, DEP_D_layer, DEP_S_layer, and GEN_layer at the same layer of the package were calculated. S_layer is determined by adding the weights of x_1, x_2, x_3, and x_4 to the four mentioned metrics, respectively, in order to calculate ASS_all, DEP_D_all, DEP_S_all, and GEN_all between classes across different layers of the package. Then, weights of x_5, x_6, x_7, and x_8 were added to these four metrics to determine S_all:

$$\begin{aligned}\left|\sum_{i=1}^{n} S\_\text{layer}(i)\right| &= \left[\left|\sum_{i=1}^{n} \text{ASS\_layer}(i)\right|, \left|\sum_{i=1}^{n} \text{DEP\_}D\text{\_layer}(i)\right|,\right.\\ &\left.\left|\sum_{i=1}^{n} \text{DEP\_}S\text{\_layer}(i)\right|, \left|\sum_{i=1}^{n} \text{GEN\_layer}(i)\right|\right]\\ &\times [x_1, x_2, x_3, x_4]^T,\\ \left|\sum_{i=1}^{n} S\_\text{all}(i)\right| &= \left[\left|\sum_{i=1}^{n} \text{ASS\_all}(i)\right|, \left|\sum_{i=1}^{n} \text{DEP\_}D\text{\_all}(i)\right|,\right.\\ &\left.\left|\sum_{i=1}^{n} \text{DEP\_}S\text{\_all}(i)\right|, \left|\sum_{i=1}^{n} \text{GEN\_all}(i)\right|\right]\\ &\times [x_5, x_6, x_7, x_8]^T.\end{aligned} \tag{1}$$

(4) S_layer and S_all are weighted to calculate the weighted out-degree S of the software system. The system coupling is calculated through dividing $\overline{S}$ by the number of classes:

$$\begin{aligned}S &= \left|\sum_{i=1}^{n} S(i)\right| = \left[\left|\sum_{i=1}^{n} S\_\text{layer}(i)\right|, \left|\sum_{i=1}^{n} S\_\text{all}(i)\right|\right] \times [\alpha, \beta]^T,\\ \overline{S} &= \frac{S}{n}.\end{aligned} \tag{2}$$

2.3.2. Calculation of the Weighted Fan-Out of Classes in a Software System. A special bipartite graph is constructed between classes of a software system. Weighted fan-out of classes in the special bipartite graph is analyzed based on characteristics of the object-oriented software.

(1) Construction of the Special Bipartite Graph in Software Systems. In the graph $G(V, E)$, if we divide the set V of nodes into two complementary subsets S and T, $S \cup T = V$, and $S \cap T = \phi$, the graph $G(V, E)$ is the bipartite graph. In the graph $G_{ij}(C_{ij}, E_{ij})$ constructed by classes C_i and C_j for the software, if only coupling relationship between classes is considered, coupling of methods and attributes in the class wouldn't be taken into account; then, the

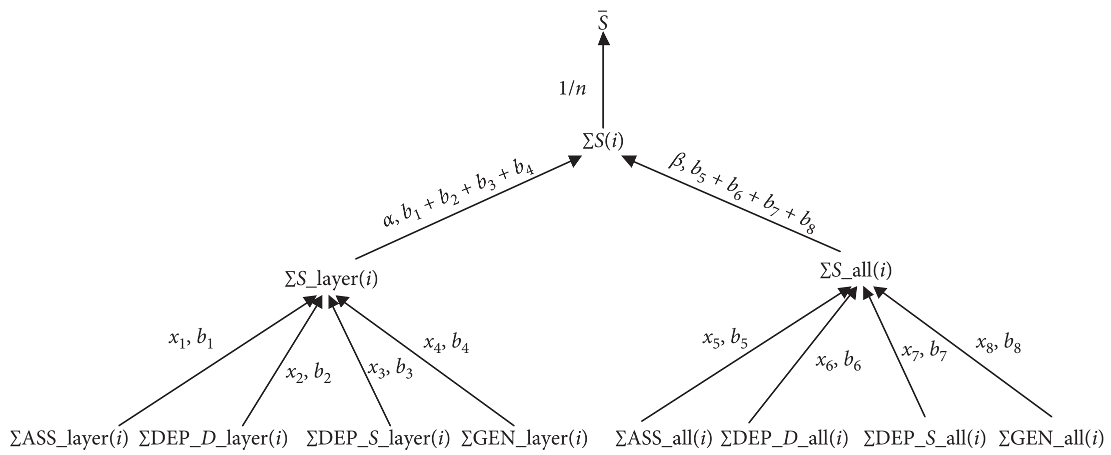


Figure 2: Illustration of the coupling measurement for a software system.

property of bipartite graph $C_i \cap C_j = \varnothing$ is satisfied. A network diagram constructed by classes C_i and C_j satisfies the following formula: $C_i \cup C_j = C_{ij}$. Moreover, the two points of a connecting edge between classes C_i and C_j are in classes C_i and C_j, respectively.

In summary, the complex coupling between classes C_i and C_j constructs a bipartite graph $G_{ij}(C_{ij}, E_{ij})$. However, $G_{ij}(C_{ij}, E_{ij})$ not only possesses the general properties of a bipartite graph, including method invocation and dependencies, but also possesses its own characteristics. In aggregation, reference, inheritance, and interface implementation between classes C_i and C_j, the two points of the connecting edge are in classes C_i and C_j, respectively. This bipartite graph $G_{ij}(C_{ij}, E_{ij})$ is defined as a special bipartite graph. However, the software system $G(C, E)$ can be considered as a set of special bipartite graphs $G_{ij}(C_{ij}, E_{ij})$ as well (Figure 3).

In the present study, the coupling of the complete bipartite graph $G_{ij}(C_{ij}, E_{ij})$ constructed by classes C_i and C_j was used to analyze the coupling of the software system $G(C, E)$.

(2) Modeling the Coupling of Special Bipartite and Calculating the Number of Weighted Fan-Out in Software Systems. In this study, a software system $G = (C_1, C_2, \ldots, C_N)$ was defined. Classes C_i and C_j were defined as two different classes in a software system: $C_i = (O_i, A_i, M_i)$. Among them, $O_i = \{O_{i1}, O_{i2}, \ldots, O_{ip}\}$ was the set of instantiated objects in class C_i and p is the number of instantiated objects. $A_i = \{A_{i1}, A_{i2}, \ldots, A_{iq}\}$ is the attribute set of class C_i, and q is the number of attribute. $M_i = \{M_{i1}, M_{i2}, \ldots, M_{ir}\}$ is the method set of class C_i, and r is the number of methods. The methods included class methods and instance methods, that is, $(C\_M \cup C\_O\_M) \subset M$.

The relationship of the special bipartite graph between classes C_i and C_j can be summarized as follows:

ASS

In class C_i, there was instantiation of class C_j (association), or one class existed in another class in the form of attribute (aggregation), which was defined as $C_j\_O_{jp'}$, where $1 \le p' \le p$.

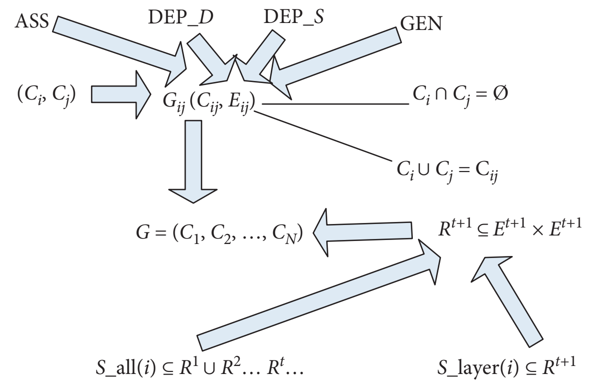

ASS
DEP_D
DEP_S
GEN
(C_i, C_j)
$G_{ij}(C_{ij}, E_{ij})$
$C_i \cap C_j = \varnothing$
$C_i \cup C_j = C_{ij}$
$G = (C_1, C_2, \ldots, C_N)$
$R^{t+1} \subseteq E^{t+1} \times E^{t+1}$
S_all(i) $\subseteq R^1 \cup R^2 \ldots R^t \ldots$
S_layer(i) $\subseteq R^{t+1}$

Figure 3: Abstract diagram of the software system network composed of a set of special bipartite graphs.

In the class C_i, instantiated object $O_{jp'}$ of class C_i was implemented $(1 \le p' \le p)$, or $O_{jp'}$ was a part of class C_i; then, there was an ASS edge between classes C_i and C_j, that is, $(C_i, C_j\_O_{jp'}) \in R_{\text{ASS}}$. The set of ASS weighted fan-out of class C_i was

$$\text{ASS}(i) = \left\{ \left(C_i, C_j\_O_{jp'}\right) \mid 1 \le j \le N, 1 \le p' \le p, \left(C_i, C_j\_O_{jp'}\right) \in R_{\text{ASS}} \right\}. \quad (3)$$

DEP

Relationship between classes is implemented defined by instance methods and variables.

In class C_i, there are instance methods of class C_j: if $C_j\_O_{jp'}\_M_{jr'}$, $1 \le j \le N, 1 \le p' \le p$, and $1 \le r' \le r$, then the relationship between classes C_i and C_j is defined as $(C_i, C_j\_O_{jp'}\_M_{jr'}) \in R_{\text{DEP\_D\_M}}$. In class C_i, there are instance variables of class C_j: if $C_j\_O_{jp'}\_A_{jq'}$, $1 \le j \le N, 1 \le p' \le p$, and $1 \le q' \le q$, then relationship between classes C_i and C_j is defined as $(C_i, C_j\_O_{jp'}\_A_{jq'}) \in R_{\text{DEP\_D\_A}}$. Under the condition of instance methods and instance variables, the set of weighted fan-out for C_i was

$$\begin{aligned}\text{DEP\_}D(i) = \{&(C_i, C_j\text{-}O_{jp'}\text{-}M_{jr'}), (C_i, C_j\text{-}O_{jp'}\text{-}A_{jq'}) \\ &\cdot \mid 1 \le j \le N, 1 \le p' \le p, 1 \le r' \le r, \\ &1 \le q' \le q, (C_i, C_j\text{-}O_{jp'}\text{-}M_{jr'}) \in R_{\text{DEP\_}D}, \\ &(C_i, C_j\text{-}O_{jp'}\text{-}A_{jq'}) \in R_{\text{DEP\_}D\text{\_}A}\}.\end{aligned} \tag{4}$$

Implementing connecting edges between two classes through class methods and class variables.

If there are class methods of C_j (static methods) in class C_i: $C_j.M_{jr'}$, and $1 \le r' \le r$, then the relationship between classes C_i and C_j is defined as $(C_i, C_j\text{-}M_{jr'}) \in R_{\text{DEP\_}S\text{\_}M}$, $1 \le r' \le r$. If there are class variables (static variables) of C_j in class C_i:$C_j.A_{jq'}$, $1 \le q' \le q$, then the relationship between classes C_i and C_j is defined as $(C_i, C_j\text{-}A_{jq'}) \in R_{\text{DEP\_}S\text{\_}A}$, $1 \le q' \le q$. Thereafter, under the conditions of class methods and class variables, the set of weighted out-degree for class C_i was

$$\begin{aligned}\text{DEP\_}S(i) = \{&(C_i, C_j\text{-}M_{jr'}), (C_i, C_j\text{-}A_{jq'}) \\ &\cdot \mid 1 \le j \le N, 1 \le r' \le r, 1 \le q' \le q, \\ &(C_i, C_j\text{-}M_{jr'}) \in R_{\text{DEP\_}S\text{\_}M}, \\ &(C_i, C_j\text{-}A_{jq'}) \in R_{\text{DEP\_}S\text{\_}A}\}.\end{aligned} \tag{5}$$

GEN

As the inheritance is preferred in software engineering, if one class is a subclass of another class, the derived connecting edge was taken into account only once in this study. Because transfer of derived connecting edges would make the software system network more complex, this study did not consider transfer of derived connecting edge but only considered the conditions that class C_i was a direct subclass of class C_j (through extension), or class C_i was implemented through interface class C_j (through implementation), or class C_i was implemented by abstract class C_j (through extension). Thus, there was a GEN connecting edge between classes C_i and C_j, which was defined as $(C_i, C_j) \in R_{\text{GEN}}$, and a set of GEN weighted out-degree for class C_i was

$$\text{GEN}(i) = \{(C_i, C_j) \mid (C_i, C_j) \in R_{\text{GEN}}\}. \tag{6}$$

(3) Determination of Weights. In software systems, one class can construct one or more special bipartite graphs with other classes. Supposing that the number of classes and the total number of weighted fan-out of all classes are definite in a software system, the first case is that the number of weighted fan-out in each class is the same or roughly the same. The second case is that there is no rule for the distribution of the number of weighted fan-out in a class. The third case is that the number of weighted fan-out of a class is heterogeneity, which approximately obeys the power-law distribution. For the second case, heterogeneity of the out-degree of the class is superior than that of the first case; however, this is impossible to be compared with the third case. For the third case, because the number of fan-out is limited for the majority of classes, only few classes have a large number of fan-out; therefore, maintenance staff can dedicate more effort on these few classes. Moreover, the maintenance workload of these classes is lower than that of the first case.

In this study, heterogeneity under the situation of fan-out was analyzed. If the distribution was the above-mentioned third case, then the larger the power-law value, the easier the maintenance, and the smaller the coupling degree. However, if the distribution was one of the other two cases, then it was considered in this study that the power-law value was equal to 1. b_1, b_2, b_3, and b_4 are the power-law values for the distribution of ASS_layer, DEP_*D*_layer, DEP_*S*_layer, and GEN_layer, respectively. b_5, b_6, b_7, and b_8 are the power-law values for the distribution of ASS_all, DEP_*D*_all, DEP_*S*_all, and GEN_all, respectively. The calculating formula for weights was as follows:

$$\begin{aligned}[x_1, x_2, x_3, x_4]^T &= \left[\sum_{i=1}^{4} \frac{b_i}{b_1}, \sum_{i=1}^{4} \frac{b_i}{b_2}, \sum_{i=1}^{4} \frac{b_i}{b_3}, \sum_{i=1}^{4} \frac{b_i}{b_4}\right]^T, \\ [x_5, x_6, x_7, x_8]^T &= \left[\sum_{i=5}^{8} \frac{b_i}{b_5}, \sum_{i=5}^{8} \frac{b_i}{b_6}, \sum_{i=5}^{8} \frac{b_i}{b_7}, \sum_{i=5}^{8} \frac{b_i}{b_8}\right]^T, \\ \alpha &= \frac{\sum_{i=4}^{8} b_i}{\sum_{i=1}^{8} b_i}, \\ \beta &= 1 - \alpha.\end{aligned} \tag{7}$$

In the present study, statistical analyses were performed for the out-degree of the three open-source software systems, and the distributions were the first and the third cases as mentioned above, demonstrating that the proposed method had a certain practical value.

2.4. Theoretical Verification of Coupling Metrics. Whether the proposed CSBG method met the mathematical properties of the measurement metrics [4] was theoretically verified.

CSBG Property 1. CSBG satisfies nonnegativity.

Proof. In an object-oriented software system $G = (c_1, c_2, \ldots, c_N)$, there are two classes $c_1, c_2 \in G$. When ASS_layerDEP_*D*_layer, DEP_*S*_layer, and GEN_layer are all 0, the minimum value of the software system CSBG(G) is 0. However, there is a maximum value M $(M > 0)$, so that the CSBG(G) value is in the range of $[0, M]$. Thus, nonnegativity of CSBG is satisfied, and the proposition is proved. □

CSBG Property 2. CSBG satisfies zero value.

Proof. As described in CSBG property 1, if the minimum value is 0, then CSBG satisfies zero-value, and the proposition is proved as well. □

CSBG Property 3. CSBG satisfies monotonicity.

Proof. If one edge is arbitrarily added in the system, the weighted out-degree of classes would increase according to CSBG measurement metrics. Obviously, the coupling increases as well. Thus, CSBG meets monotonicity and the proposition is proved. □

CSBG Property 4. CSBG meets the property of class merging.

Proof. In an object-oriented software system $G = (c_1, c_2, \ldots, c_N)$, there are two classes $c_1, c_2 \in G$, and class c' is a merger of classes c_1 and c_2. The object-oriented system G' is a system in which classes c_1 and c_2 in G are replaced by class c'. CSBG mainly calculates the weighted out-degree of classes in software systems. Therefore,

$$[\mathrm{CSBG}(c_1) + \mathrm{CSBG}(c_2) \geq \mathrm{CSBG}(c') \mid \mathrm{CSBG}(G) \geq \mathrm{CSBG}(G')]. \quad (8)$$

□

CSBG Property 5. CSBG satisfies the merge property of two irrelevant classes.

Proof. In an object-oriented software system $G = (c_1, c_2, \ldots, c_N)$, there are two classes $c_1, c_2 \in G$, and the two classes are not coupled. Moreover, class c' is the merger of classes c_1 and c_2. The object-oriented system G' is a system in which classes c_1 and c_2 in G are replaced by class c'. CSBG mainly calculates the weighted out-degree of classes in software systems. Therefore,

$$[\mathrm{CSBG}(c_1) + \mathrm{CSBG}(c_2) = \mathrm{CSBG}(c') \mid \mathrm{CSBG}(G) = \mathrm{CSBG}(G')]. \quad (9)$$

□

2.5. Comparative Experiment.

2.5. Comparative Experiment. In the next sections, the CSBG method is herein proposed for coupling measurement and the existing measurement methods were compared and analyzed in order to verify the rationality of the results of CSBG measurement.

2.5.1. Calculating the Coupling of the Software System Using CSBG. In this section, CSBG for coupling measurement was compared with the existing measurement methods.

This experiment was conducted on a simple system as an example to analyze and compare the measurement values by the existing coupling measurements. This system was composed of 6 classes (Shape.java, Point.java, Line.java, Triangle.java, Quadrilateral.java, and Square.java), which described shapes, points, edges, triangles, quadrilaterals, and squares, respectively.

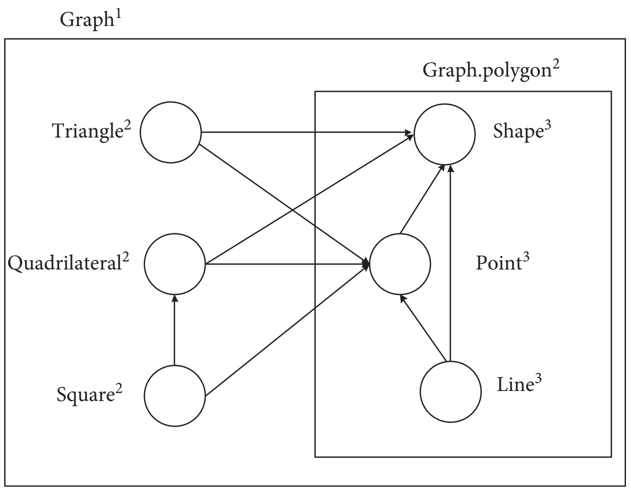
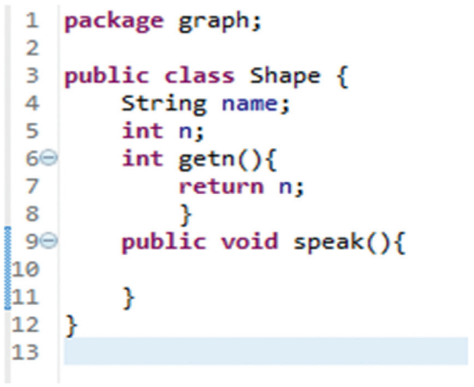
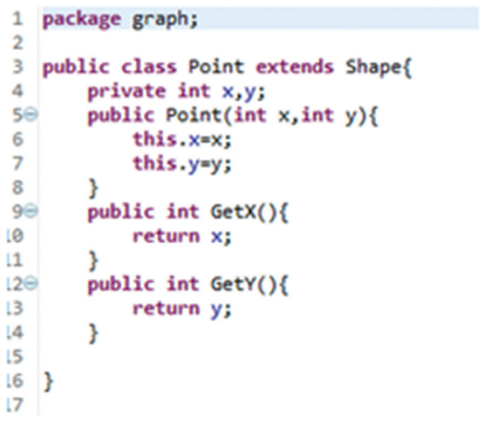


Figure 4: Diagram at package level.



```
package graph;

public class Shape {
    String name;
    int n;
    int getn(){
        return n;
        }
    public void speak(){

    }
}

```

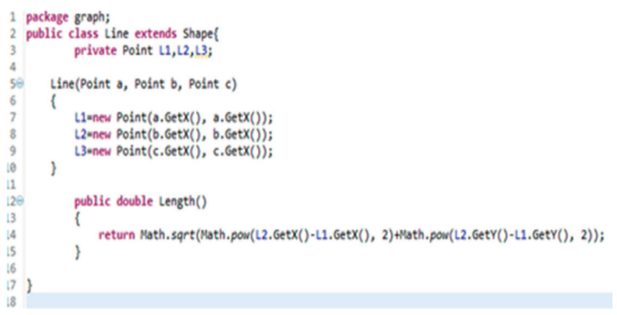
Figure 5: Demonstration of class shape.



```
package graph;

public class Point extends Shape{
    private int x,y;
    public Point(int x,int y){
        this.x=x;
        this.y=y;
    }
    public int GetX(){
        return x;
    }
    public int GetY(){
        return y;
    }

}

```

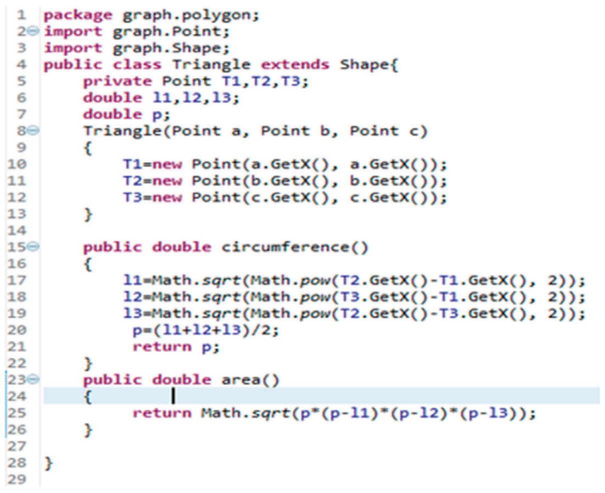
Figure 6: Demonstration of class point.

Among them, the first three classes were in package graph, and the last three classes were in package graph.polygon (hierarchy of classes in package level is shown in Figure 4). There were inheritance, combination, variable declaration, and method invocation among these classes, which were appropriate for analyzing the coupling model. Codes of classes are shown in Figures 5–10.



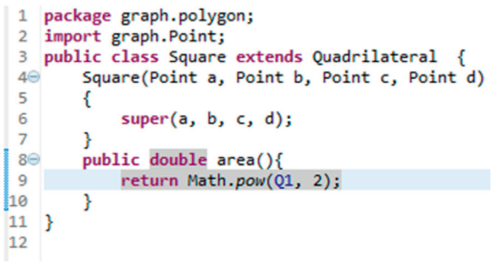
```
package graph;
public class Line extends Shape{
        private Point L1,L2,L3;

    Line(Point a, Point b, Point c)
    {
        L1=new Point(a.GetX(), a.GetX());
        L2=new Point(b.GetX(), b.GetX());
        L3=new Point(c.GetX(), c.GetX());
    }

        public double Length()
        {
            return Math.sqrt(Math.pow(L2.GetX()-L1.GetX(), 2)+Math.pow(L2.GetY()-L1.GetY(), 2));
        }

}
```

Figure 7: Demonstration of class line.



```
package graph.polygon;
import graph.Point;
import graph.Shape;
public class Triangle extends Shape{
    private Point T1,T2,T3;
    double l1,l2,l3;
    double p;
    Triangle(Point a, Point b, Point c)
    {
        T1=new Point(a.GetX(), a.GetX());
        T2=new Point(b.GetX(), b.GetX());
        T3=new Point(c.GetX(), c.GetX());
    }

    public double circumference()
    {
        l1=Math.sqrt(Math.pow(T2.GetX()-T1.GetX(), 2));
        l2=Math.sqrt(Math.pow(T3.GetX()-T1.GetX(), 2));
        l3=Math.sqrt(Math.pow(T2.GetX()-T3.GetX(), 2));
         p=(l1+l2+l3)/2;
         return p;
    }
    public double area()
    {
         return Math.sqrt(p*(p-l1)*(p-l2)*(p-l3));
    }

}
```

Figure 8: Demonstration of class triangle.



```
package graph.polygon;
import graph.Point;
public class Square extends Quadrilateral  {
    Square(Point a, Point b, Point c, Point d)
    {
        super(a, b, c, d);
    }
    public double area(){
        return Math.pow(Q1, 2);
    }
}
```

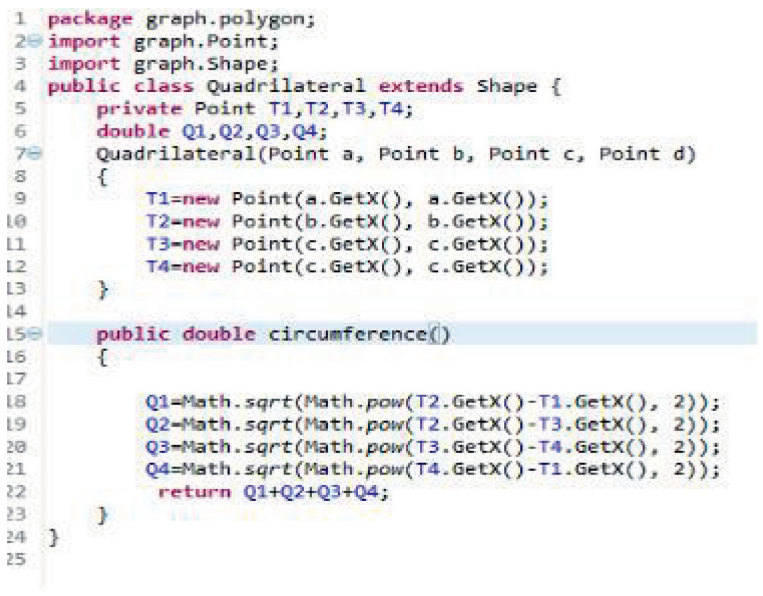
Figure 9: Demonstration of class square.

There were three classes in the package graph, including class Shape, class Point, and class Line.

There were three classes in the package graph.polygon, which were classes of Triangle, Square, and Quadrilateral.

In this study, an algorithm was designed and the program was developed based on the aforementioned mathematical model, mainly calculating the four metrics for the out-degree of classes in the same layer and different layers of the package in software systems, including ASS, DEP_*D*, DEP_*S*, and GEN. Coupling metrics, including ASS, DEP_*D*, DEP_*S*, and GEN, were corresponded to the cases described in Section 2.3. Out-degrees of classes in various layers are shown in Table 4.

The mathematical model described in Section 2.3 was herein used. Because the number of classes was small, the heterogeneity of out-degree of classes could not be reflected. Moreover, heterogeneity had little impact on the coupling in this example. Therefore, it was considered that heterogeneity was approximately the same. Coupling of software systems was calculated as follows:



```java
package graph.polygon;
import graph.Point;
import graph.Shape;
public class Quadrilateral extends Shape {
    private Point T1,T2,T3,T4;
    double Q1,Q2,Q3,Q4;
    Quadrilateral(Point a, Point b, Point c, Point d)
    {
        T1=new Point(a.GetX(), a.GetX());
        T2=new Point(b.GetX(), b.GetX());
        T3=new Point(c.GetX(), c.GetX());
        T4=new Point(c.GetX(), c.GetX());
    }

    public double circumference()
    {

        Q1=Math.sqrt(Math.pow(T2.GetX()-T1.GetX(), 2));
        Q2=Math.sqrt(Math.pow(T2.GetX()-T3.GetX(), 2));
        Q3=Math.sqrt(Math.pow(T3.GetX()-T4.GetX(), 2));
        Q4=Math.sqrt(Math.pow(T4.GetX()-T1.GetX(), 2));
         return Q1+Q2+Q3+Q4;
    }
}
```

Figure 10: Demonstration of class quadrilateral.

$$\begin{aligned}
\mathrm{CSBG}(G) &= \frac{\sum_i^6 \mathrm{CSBG}(C_i)}{6} = 6.55, \\
[x_1, x_2, x_3, x_4]^T &= [1,1,1,1]^T, \\
[x_5, x_6, x_7, x_8]^T &= [1,1,1,1]^T, \\
\alpha &= 0.5, \\
\beta &= 1-\alpha = 0.5, \\
\left|\sum_{i=1}^{n} S\_\mathrm{layer}(i)\right| &= \left[\left|\sum_{i=1}^{n}\mathrm{ASS\_layer}(i)\right|, \left|\sum_{i=1}^{n}\mathrm{DEP\_}D\mathrm{\_layer}(i)\right|, \left|\sum_{i=1}^{n}\mathrm{DEP\_}S\mathrm{\_layer}(i)\right|, \left|\sum_{i=1}^{n}\mathrm{GEN\_layer}(i)\right|\right] \times [x_1, x_2, x_3, x_4]^T \\
&= [6, 10, 0, 3] \times [1,1,1,1]^T \\
&= 19, \\
\left|\sum_{i=1}^{n} S\_\mathrm{all}(i)\right| &= \left[\left|\sum_{i=1}^{n}\mathrm{ASS\_all}(i)\right|, \left|\sum_{i=1}^{n}\mathrm{DEP\_}D\mathrm{\_all}(i)\right|, \left|\sum_{i=1}^{n}\mathrm{DEP\_}S\mathrm{\_all}(i)\right|, \left|\sum_{i=1}^{n}\mathrm{GEN\_all}(i)\right|\right] \times [x_5, x_6, x_7, x_8]^T \\
&= [24, 38, 0, 5] \times [1,1,1,1]^T \\
&= 67, \\
S = \left|\sum_{i=1}^{n} S(i)\right| &= \left[\left|\sum_{i=1}^{n} S\_\mathrm{layer}(i)\right|, \left|\sum_{i=1}^{n} S\_\mathrm{all}(i)\right|\right] \times [\alpha, \beta]^T \\
&= [19, 67] \times [0.5, 0.5]^T \\
&= 43, \\
\overline{S} &= \frac{S}{n} = \frac{43}{6} = 7.17.
\end{aligned} \tag{10}$$

Table 4: Out-degrees of classes at the same layer and all layers of the package.

Class name	Out-degree at the same layer					Out-degree of all layers				
	ASS	DEP_*D*	DEP_*S*	GEN	Total	ASS	DEP_*D*	DEP_*S*	GEN	Total
Quadrilateral	0	0	0	0	0	8	16	0	1	25
Triangle	0	0	0	0	0	6	12	0	1	19
Line	6	10	0	1	17	6	10	0	1	17
Point	0	0	0	1	1	0	0	0	1	1
Shape	0	0	0	0	0	0	0	0	0	0
Square	0	0	0	1	1	4	0	0	1	5
Total	6	10	0	3	19	24	38	0	5	67

2.5.2. Analysis of the Results of Various Methods for Coupling Measurement. Coupling of software systems was calculated based on existing measurement methods, which is shown in Table 5. In addition to CSBG coupling measurement, other measurement methods mainly focus on measurement of a certain local fine-grained aspect. These methods were based on the theory of reductionism, which did not investigate the coupling of software systems from an overall and global perspective. Therefore, the measurement values were mostly either very large or very small, and several metrics were equal to 0. In addition, discrimination of these metrics was not significant compared with other methods for coupling measurement. The metrics calculated by CSBG had a better discrimination. Therefore, the existed methods have limitations, which cannot accordingly satisfy an effective coupling measurement for complex software systems. CSBG not only can consider a complex relationship between classes in object-oriented software systems but also analyze the complexity of classes and the special bipartite graph composed of classes from the prospective of overall package level. Therefore, the CSBG measurement method contained a certain rationality in theory.

3. Results

3.1. Application of CSBG Measurement Metrics in the Three Open-Source Software Systems. In order to further validate the effects of CSBG, this study used CSBG to measure and analyze coupling between classes in the three Java open-source software systems from different fields, including Art of Illusion [48], JabRef [49], and GanttProject [50]. Some studies have reported results of class cohesion metrics for the three open-source software systems [51–53]; it is feasible to further study the complexity of the three open-source software systems if there is a more reasonable method for coupling measurement. In order to verify the effects of the CSBG measurement method in actual open-source software systems, three Java open-source software systems from different aforementioned fields were used. Art of Illusion is a software system for 3D rendering, modeling, and animation. JabRef is a graphical application for managing bibliographic database. GanttProject is a software system for project scheduling characterized by resource calendar, management, and import or export (MS Project, PDF, HTML). The reasons to use the three open-source software systems in the measurement were because (1) these systems were based on object-oriented Java; (2) the classes in the systems had a certain scale; (3) the three systems were from different fields; and (4) the source codes were available as well. Scholars can freely download the source codes from an open-source website (http://sourceforge.net).

3.2. Association of Coupling with Statistical Characteristics of the Three Open-Source Software Systems. Firstly, the program was developed and out-degree of classes at the same layer and all layers of the package was eventually obtained, including ASS, DEP_*D*, DEP_*S*, and GEN.

In this section, DEP_*D* and DEP_*S* were analyzed, and the results are shown in Figures 11–18. In the experimental results, DEP(i) was a nonstandardized part of probability distribution $P(i)$. If, $P(i) \sim i^{-\gamma}$, then DEP $\sim (i)^{-\gamma}$. A linear function was fitted using the double logarithmic method that was fitted to estimate Gamma index γ (R is the Pearson's correlation coefficient and SD is standard deviation; γ is also expressed as B in the following table).

Although inheritance between classes increases coupling of the system, this is encouraged by the software system, which is conducive to reduce function definition and attribute definition in order to create a new class; thus, it is a poor coupling. It can be seen from linear distribution of GEN (Table 6) that neither all classes have an inheritance relationship, nor the GEN fan-out of all classes were very large or very small. However, classes with values equal to 0 or 1 were dominant.

Pearson's correlation coefficient (R) and SD value provided the quality of the linear fitting; the larger the R value, the better the quality of the linear fitting, and B is estimated Gamma index γ. Moreover, the smaller the SD value, the better the quality of the linear fitting. As shown in Table 7, if 0.95 is considered to be the minimum value, it can be approximated that the distribution obeyed the power-law distribution except that ASS value in JabRef was relatively small (0.91651 and 0.88148). Furthermore, the distributions of ASS layer, ASS_all layer, DEP_*D* layer, DEP_*S* layer, DEP_*D*_all layer, and DEP_*S*_all layer were assumed to obey power-law distribution. The results demonstrated that there was a certain rule for the number of fan-out of classes in the form of ASS and DEP, which was not the case that the values were mostly large or small. However, they had "scale-free" property for complex networks, which obeyed the power-law distribution. In actual software development process, if software developers excessively pursue low coupling between classes, a class may be divided into two or more subclasses; thus, system complexity may be accordingly increased. The process of determination of the range of coupling

Table 5: Results of various methods for coupling measurement.

	Quadrilateral	Triangle	Line	Point	Shape	Square	Software system
CSBG	17.5	13.3	5.1	0.3	0	3.1	7.17
CBO	3	2	2	5	4	2	
CBO′	1	1	1	4	0	1	
RFC	0	0	0	38	0	0	
RFCα	0	0	0	38	0	0	
RFC′	0	0	0	38	0	0	
MPC	0	0	0	0	0	0	
DAC	8	6	6	0	0	4	
DAC′	1	1	1	0	0	1	
COF							0.2
ICP	16	12	10	0	0	1	
IH-ICP	0	0	0	0	0	1	
NIH-ICP	16	12	10	0	0	0	
SIMAS	0	0	0	0	0	0	
PIM	16	12	10	0	0	1	
PIMAS	16	12	10	0	0	1	
INAG	1	1	1	0	0	1	
ACAIC	0	0	0	0	0	0	
OCAIC	4	3	3	0	0	0	
ACMIC	0	0	0	0	0	0	
OCMIC	0	0	0	0	0	0	
AMMC	0	0	0	0	0	1	
OMMC	16	12	10	0	0	1	
ICF	0	0	0	1	0	0	
FCF	1	1	1	0	0	1	

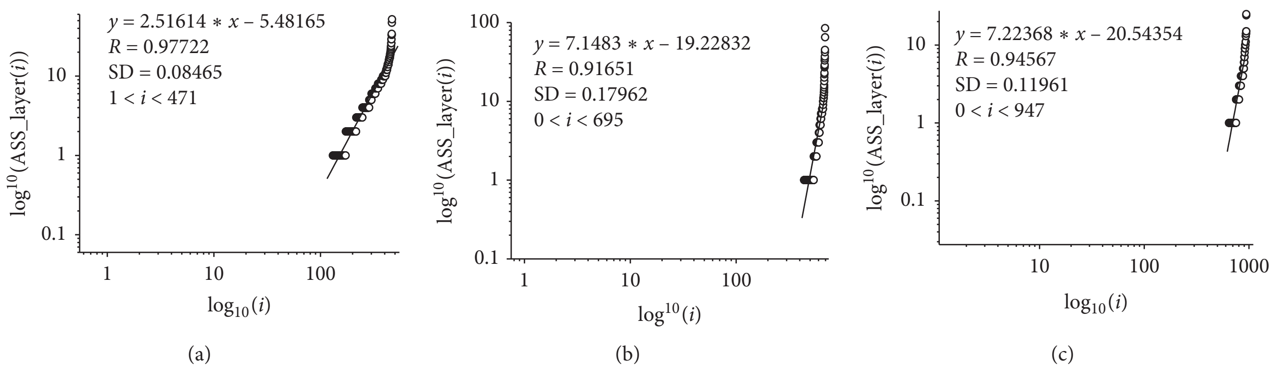


Figure 11: The double logarithmic diagrams of the fan-out of ASS invocation for classes at the same layer of a package.

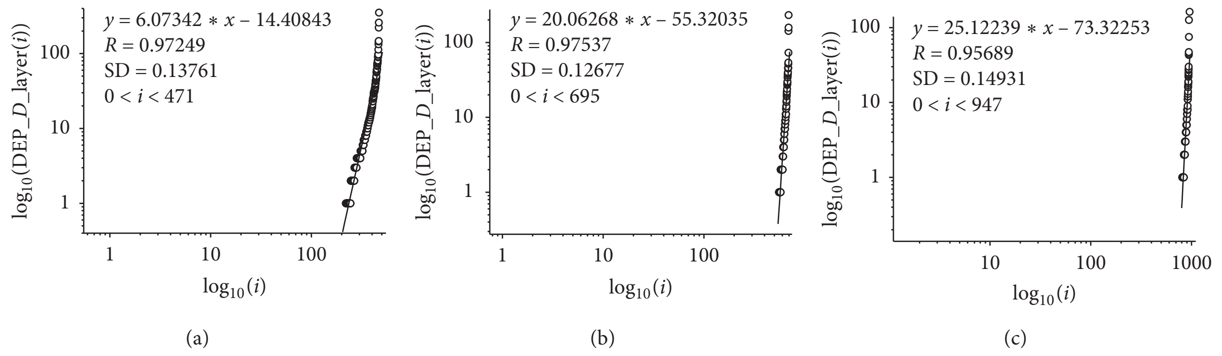


Figure 12: The double logarithmic diagrams of the fan-out of DEP_D invocation for classes at the same layer of a package.

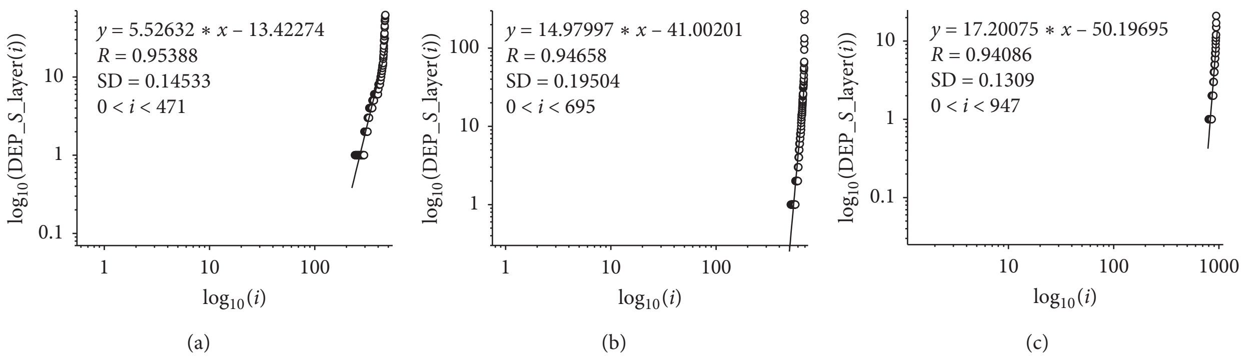


Figure 13: The double logarithmic diagrams of the fan-out of DEP_S invocation for classes at the same layer of a package.

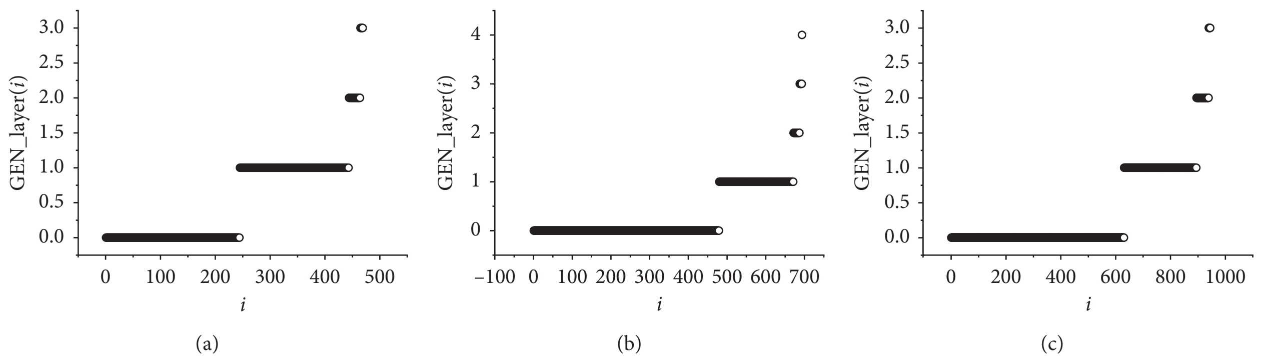


Figure 14: The double logarithmic diagrams of fan-out of GEN invocation for classes at the same layer of a package.

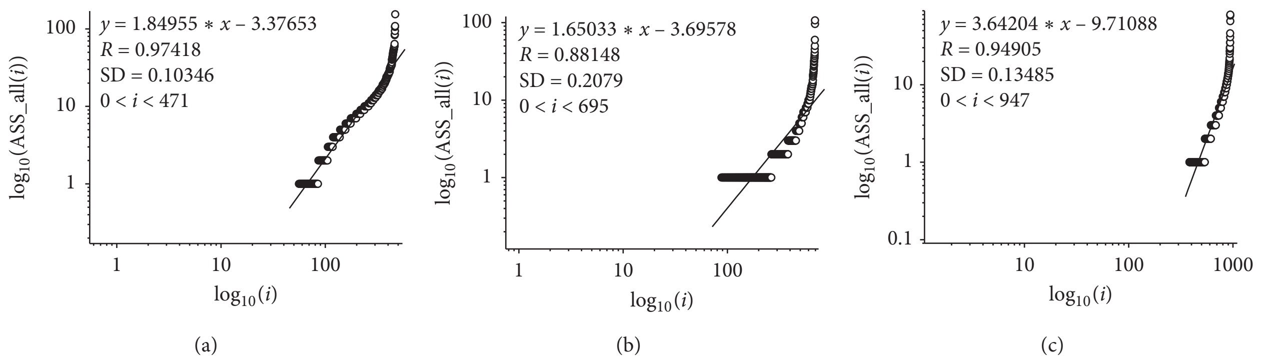


Figure 15: The double logarithmic diagrams of the fan-out of ASS invocation for classes at all layers of a package.

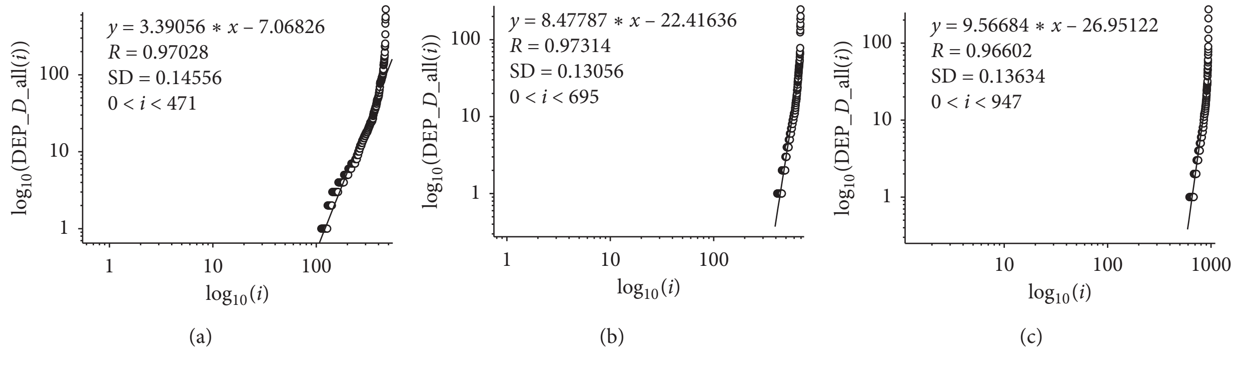


Figure 16: The double logarithmic graph of fan-out of DEP_D invocation for classes at all layers of a package.

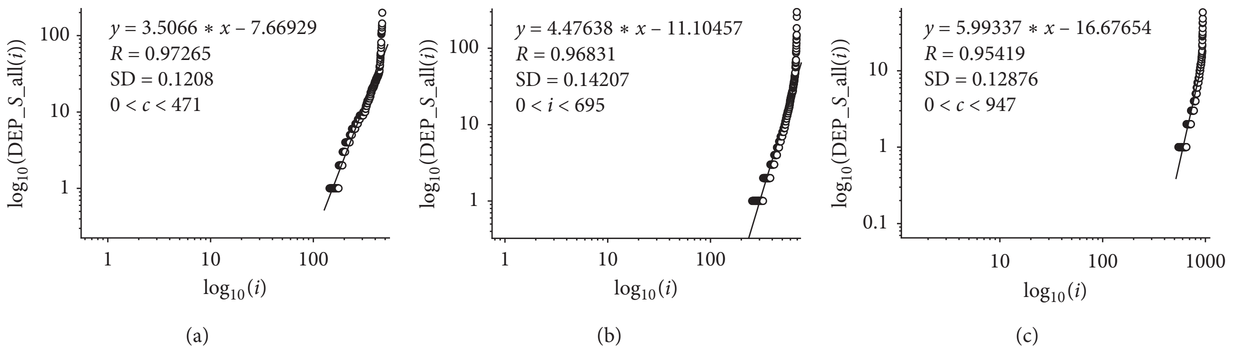


Figure 17: The double logarithmic diagrams of fan-out of DEP_*S* invocation for classes at all layers of a package.

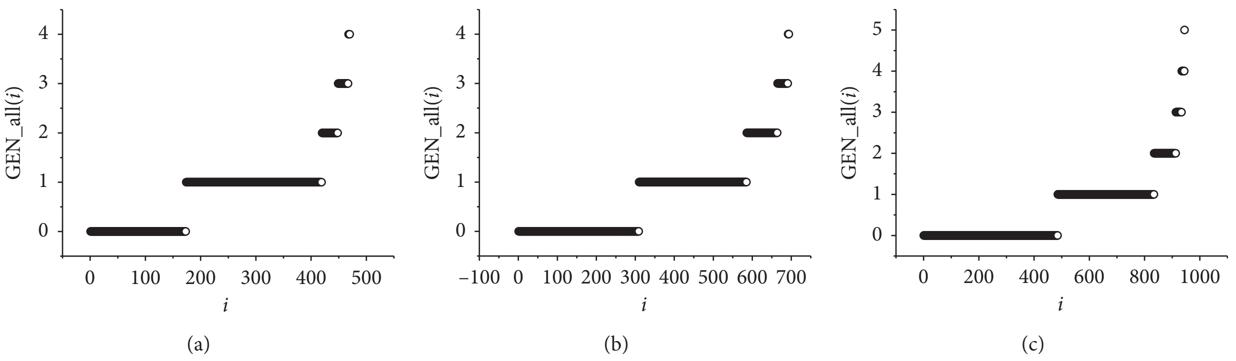


Figure 18: The double logarithmic diagrams of fan-out of GEN invocation for classes at all layers of a package.

Table 6: Values of fan-out for different classes of GEN.

	Value of fan-out	0	1	2	3	4	5	7
GEN_layer	Illusion	244	199	21	5	1	0	0
	JabRef	479	192	16	6	1	0	0
	GanttProject	630	264	45	6	0	1	0
GEN_all	Illusion	173	246	29	19	3	0	0
	JabRef	309	276	79	27	3	0	0
	GanttProject	485	349	79	21	10	1	1

Table 7: Values of *R*, SD, and *B* parameters.

	Software system	*R*	SD	*B*
ASS_layer	Illusion	0.9772	0.08465	2.51614
	JabRef	0.91651	0.17962	7.1483
	GanttProject	0.94567	0.11961	7.22368
DEP_*D*_layer	Illusion	0.97249	0.31761	6.07342
	JabRef	0.97537	0.12677	20.06268
	GanttProject	0.95689	0.14931	25.12239
DEP_*S*_layer	Illusion	0.95388	0.14533	5.52632
	JabRef	0.94658	0.19504	14.97997
	GanttProject	0.94086	0.1309	17.20075
ASS_all	Illusion	0.97414	0.10346	1.84955
	JabRef	0.88148	0.2079	1.65033
	GanttProject	0.94905	0.13485	3.64204
DEP_*D*_all	Illusion	0.97028	0.14556	3.39056
	JabRef	0.97314	0.13056	8.47787
	GanttProject	0.96602	0.13634	9.56684
DEP_*S*_all	Illusion	0.97265	0.1208	3.5066
	JabRef	0.96831	0.14207	4.47638
	GanttProject	0.95419	0.12876	5.99337

between classes in software systems is significant. Based on data analysis, it can be seen that scale-free" property of complex networks motivated software developers to pay more attention to the distribution range of the coupling in large-scale software systems, which could provide a reliable reference for developing more reasonable software systems.

3.3. Coupling Measurement for the Three Open-Source Software Systems. According to the results of the above-mentioned analysis, out-degrees of classes were often equal to 0, 1, and 2 for class inheritance in generalization, interface implementation, and implementation of abstract classes, which were approximately linearly distributed. Therefore, the power-law value of GEN was approximated to 1.

3.3.1. Calculation of Coupling Measurement for Art of Illusion. According to the CSBG method for coupling measurement, coupling of the software system for Art of Illusion was calculated as follows:

$$\begin{aligned}
[x_1, x_2, x_3, x_4]^T &= \left[\sum_{i=1}^{4}\frac{b_i}{b_1}, \sum_{i=1}^{4}\frac{b_i}{b_2}, \sum_{i=1}^{4}\frac{b_i}{b_3}, \sum_{i=1}^{4}\frac{b_i}{b_4}\right]^T \\
&= \left[\sum_{i=1}^{4}\frac{b_i}{2.51614}, \sum_{i=1}^{4}\frac{b_i}{6.07342}, \sum_{i=1}^{4}\frac{b_i}{5.52632}, \sum_{i=1}^{4}\frac{b_i}{1}\right]^T \\
&= [6.007567, 2.48858, 2.735252, 15.11588]^T, \\
[x_5, x_6, x_7, x_8]^T &= \left[\sum_{i=5}^{8}\frac{b_i}{b_5}, \sum_{i=5}^{8}\frac{b_i}{b_6}, \sum_{i=5}^{8}\frac{b_i}{b_7}, \sum_{i=5}^{8}\frac{b_i}{b_8}\right]^T \\
&= \left[\sum_{i=5}^{8}\frac{b_i}{1.84955}, \sum_{i=5}^{8}\frac{b_i}{3.39056}, \sum_{i=5}^{8}\frac{b_i}{3.5066}, \sum_{i=5}^{8}\frac{b_i}{1}\right]^T \\
&= [5.269774, 2.874661, 2.779533, 9.74671]^T, \\
\alpha &= \frac{\sum_{i=4}^{8} b_i}{\sum_{i=1}^{8} b_i} = 0.392023, \\
\beta &= 1 - \alpha = 0.607977, \\
\left|\sum_{i=1}^{n} S\_\text{layer}(i)\right| &= \left[\left|\sum_{i=1}^{n}\text{ASS\_layer}(i)\right|, \left|\sum_{i=1}^{n}\text{DEP\_}D\text{\_layer}(i)\right|, \left|\sum_{i=1}^{n}\text{DEP\_}S\text{\_layer}(i)\right|, \left|\sum_{i=1}^{n}\text{GEN\_layer}(i)\right|\right] \times [x_1, x_2, x_3, x_4]^T \\
&= [2479, 5848, 2005, 260] \times [6.007567, 2.48858, 2.735252, 15.11588]^T \\
&= 38861.91, \\
\left|\sum_{i=1}^{n} S\_\text{all}(i)\right| &= \left[\left|\sum_{i=1}^{n}\text{ASS\_all}(i)\right|, \left|\sum_{i=1}^{n}\text{DEP\_}D\text{\_all}(i)(i)\right|, \left|\sum_{i=1}^{n}\text{DEP\_}S\text{\_all}(i)\right|, \left|\sum_{i=1}^{n}\text{GEN\_all}(i)\right|\right] \times [x_5, x_6, x_7, x_8]^T \\
&= [6705, 13883, 6255, 373] \times [5.269774, 2.874661, 2.779533, 9.74671]^T \\
&= 96264.25, \\
S &= \left|\sum_{i=1}^{n} S(i)\right| = \left[\left|\sum_{i=1}^{n} S\_\text{layer}(i)\right|, \left|\sum_{i=1}^{n} S\_\text{all}(i)\right|\right] \times [\alpha, \beta]^T \\
&= [38861.91, 96264.25] \times [0.392023, 0.607977]^T \\
&= 73761.21, \\
\overline{S} &= \frac{S}{n} = \frac{73761.21}{470} = 156.9387.
\end{aligned} \tag{11}$$

3.3.2. Calculation of Coupling Measurement for JabRef. According to CSBG for coupling measurement, coupling of the software system for JabRef was calculated as follows:

$$\begin{aligned}
[x_1, x_2, x_3, x_4]^T &= \left[\sum_{i=1}^{4}\frac{b_i}{b_1}\frac{b_i}{b_1}, \sum_{i=1}^{4}\frac{b_i}{b_2}, \sum_{i=1}^{4}\frac{b_i}{b_3}, \sum_{i=1}^{4}\frac{b_i}{b_4}\right]^T \\
&= \left[\sum_{i=1}^{4}\frac{b_i}{7.1483}\frac{b_i}{7.1483}, \sum_{i=1}^{4}\frac{b_i}{20.06268}, \sum_{i=1}^{4}\frac{b_i}{14.97997}, \sum_{i=1}^{4}\frac{b_i}{1}\right]^T \\
&= [6.042129, 2.152801, 2.883247, 43.19095]^T, \\
[x_5, x_6, x_7, x_8]^T &= \left[\sum_{i=5}^{8}\frac{b_i}{b_5}, \sum_{i=5}^{8}\frac{b_i}{b_6}, \sum_{i=5}^{8}\frac{b_i}{b_7}, \sum_{i=5}^{8}\frac{b_i}{b_8}\right]^T \\
&= \left[\sum_{i=5}^{8}\frac{b_i}{1.65033}, \sum_{i=5}^{8}\frac{b_i}{8.47787}, \sum_{i=5}^{8}\frac{b_i}{4.47638}, \sum_{i=5}^{8}\frac{b_i}{1}\right]^T \\
&= [9.45543, 1.840625, 3.485982, 15.60458]^T, \\
\alpha &= \frac{\sum_{i=4}^{8} b_i}{\sum_{i=1}^{8} b_i} = 0.26540419, \\
\beta &= 1 - \alpha = 0.734596, \\
\left|\sum_{i=1}^{n} S\_\mathrm{layer}(i)\right| &= \left[\left|\sum_{i=1}^{n}\mathrm{ASS\_layer}(i)\right|, \left|\sum_{i=1}^{n}\mathrm{DEP\_}D\mathrm{\_layer}(i)\right|, \left|\sum_{i=1}^{n}\mathrm{DEP\_}S\mathrm{\_all}(i)\right|, \left|\sum_{i=1}^{n}\mathrm{GEN\_layer}(i)\right|\right] \times [x_1, x_2, x_3, x_4]^T \\
&= [1326, 2032, 2396, 246] \times [6.042129, 2.152801, 2.883247, 43.19095]^T \\
&= 29375.8, \\
\left|\sum_{i=1}^{n} S\_\mathrm{all}(i)\right| &= \left[\left|\sum_{i=1}^{n}\mathrm{ASS\_all}(i)\right|, \left|\sum_{i=1}^{n}\mathrm{DEP\_}D\mathrm{\_all}(i)\right|, \left|\sum_{i=1}^{n}\mathrm{DEP\_}S\mathrm{\_all}(i)\right|, \left|\sum_{i=1}^{n}\mathrm{GEN\_all}(i)\right|\right] \times [x_5, x_6, x_7, x_8]^T \\
&= [3430, 4403, 6733, 527] \times [9.45543, 1.840625, 3.485982, 15.60458]^T \\
&= 71947.46, \\
S &= \left|\sum_{i=1}^{n} S(i)\right| = \left[\left|\sum_{i=1}^{n} S\_\mathrm{layer}(i)\right|, \left|\sum_{i=1}^{n} S\_\mathrm{all}(i)\right|\right] \times [\alpha, \beta]^T \\
&= [29375.8, 71947.46] \times [0.26540419, 0.734596]^T \\
&= 60648.76, \\
\overline{S} &= \frac{S}{n} = \frac{60648.76}{694} = 87.39015.
\end{aligned} \tag{12}$$

3.3.3. Calculation of Coupling Measurement for GanttProject. According to CSBG for coupling measurement, coupling of the software system for GanttProject was calculated as follows:

$$
\begin{aligned}
[x_1, x_2, x_3, x_4]^T &= \left[\sum_{i=1}^{4}\frac{b_i}{b_1}, \sum_{i=1}^{4}\frac{b_i}{b_2}, \sum_{i=1}^{4}\frac{b_i}{b_3}, \sum_{i=1}^{4}\frac{b_i}{b_4}\right]^T \\
&= \left[\sum_{i=1}^{4}\frac{b_i}{7.22368}, \sum_{i=1}^{4}\frac{b_i}{25.12239}, \sum_{i=1}^{4}\frac{b_i}{17.20075}, \sum_{i=1}^{4}\frac{b_i}{1}\right]^T \\
&= [6.997378, 2.012023, 2.93864, 50.54682]^T, \\
[x_5, x_6, x_7, x_8]^T &= \left[\sum_{i=5}^{8}\frac{b_i}{b_5}, \sum_{i=5}^{8}\frac{b_i}{b_6}, \sum_{i=5}^{8}\frac{b_i}{b_7}, \sum_{i=5}^{8}\frac{b_i}{b_8}\right]^T \\
&= \left[\sum_{i=5}^{8}\frac{b_i}{3.64204}, \sum_{i=5}^{8}\frac{b_i}{9.56684}, \sum_{i=5}^{8}\frac{b_i}{5.99337}, \sum_{i=5}^{8}\frac{b_i}{1}\right]^T \\
&= [5.5496, 2.111695, 3.370766, 20.20225]^T, \\
\alpha &= \frac{\sum_{i=4}^{8} b_i}{\sum_{i=1}^{8} b_i} = 0.285548, \\
\beta &= 1 - \alpha = 0.714452, \\
\left|\sum_{i=1}^{n} S\_\text{layer}(i)\right| &= \left[\left|\sum_{i=1}^{n}\text{ASS\_layer}(i)\right|, \left|\sum_{i=1}^{n}\text{DEP\_}D\text{\_layer}(i)\right|, \left|\sum_{i=1}^{n}\text{DEP\_}S\text{\_layer}(i)\right|, \left|\sum_{i=1}^{n}\text{GEN\_layer}(i)\right|\right] \times [x_1, x_2, x_3, x_4]^T \\
&= [1102, 1331, 565, 379] \times [6.997378, 2.012023, 2.93864, 50.54682]^T \\
&= 31206.69, \\
\left|\sum_{i=1}^{n} S\_\text{all}(i)\right| &= \left[\left|\sum_{i=1}^{n}\text{ASS\_all}(i)\right|, \left|\sum_{i=1}^{n}\text{DEP\_}D\text{\_all}(i)\right|, \left|\sum_{i=1}^{n}\text{DEP\_}S\text{\_all}(i)\right|, \left|\sum_{i=1}^{n}\text{GEN\_all}(i)\right|\right] \times [x_5, x_6, x_7, x_8]^T \\
&= [3201, 3833, 2291, 622] \times [5.5491, 2.111695, 3.370766, 20.20225]^T \\
&= 46138.17, \\
S &= \left|\sum_{i=1}^{n} S(i)\right| = \left[\left|\sum_{i=1}^{n} S\_\text{layer}(i)\right|, \left|\sum_{i=1}^{n} S\_\text{all}(i)\right|\right] \times [\alpha, \beta]^T \\
&= [31206.69, 46138.17] \times [0.285548, 0.714452]^T \\
&= 41874.52, \\
\overline{S} &= \frac{S}{n} = \frac{41874.52}{946} = 44.26482.
\end{aligned}
\tag{13}
$$

The three aforementioned open-source software systems were analyzed from the points of view of package level, class level, and method level using CSBG for coupling measurement. A program was also developed to calculate various metrics; thus, the coupling of the three open-source software systems in descending order was the Art of Illusion, JabRef, and GanttProject, suggesting that it was feasible to use CSBG for coupling measurement of software systems that contained a certain practical value.

4. Conclusion

Based on bipartite graphs for complex networks, by comprehensive consideration of the weighted fan-out between classes from points of view of package level, class level, and method level, this study expressed that the interaction of classes is a special bipartite graph, while a software system is a set of these special bipartite graphs. For this purpose, first, this study analyzed the four relationships for a software system, including ASS, DEP_*D*, DEP_*S*, and GEN, and coupling relationship for a class with other classes in the same layer of package was considered as well. Moreover, coupling relationship for classes in a package with other classes in different layers of the package was taken into account. Therefore, the CSBG method for coupling measurement of software systems was proposed, which was completely in compliance with the mathematical characteristics of the widely accepted metrics. Second, for a software system, other typical methods and CSBG method were compared for the purpose of coupling measurement, and the results revealed that the measured value was either large or small due to the defects of other measurement methods that were analyzed from an overall and global perspective. Moreover, the corresponding values were mostly equal to 0. Therefore, there were some defects in other measurement methods, while CSBG had its rationality. Eventually, a program was developed based on the CSBG method to apply the three open-source software systems (Art of Illusion, JabRef, and GanttProject). The results demonstrated that coupling of the three open-source software systems in the descending order was the Art of Illusion, JabRef, and GanttProject. Although inheritance between classes increases coupling of the system, this is also followed by software engineering, which is conducive to reduce function definition and attribute definition in order to create a new class, and thus, this is weak coupling. It can be concluded from the linear distribution of GEN that all classes either had an inheritance relationship, or that the number of GEN fan-out of all classes was very large or very small. However, classes with values equal to 0 or 1 were accounted. Furthermore, it was revealed that in the same layer and total layers of the package, fan-out values of ASS, DEP_*D*, and DEP_*S* obeyed the scale-free property of complex networks. These findings provided empirical support for the CSBG method. The statistical power-law metrics were applied to the method for coupling measurement proposed in this study in order to calculate the coupling of the three open-source software systems, which provided a reliable reference for further investigation of coupling between classes in software systems using statistics of complex networks. In [54], it was mentioned that cohesion distribution of the majority classes of a software system contained a certain regularity. In other words, it was not the case that neither cohesion of all classes was very large nor very small. In the empirical analysis of coupling, the values of coupling metrics had a regularity similar to class cohesion. Although coupling represented the degree of interdependence between classes, the greater the coupling, the more complex the software from an intuitive aspect. However, excessive pursuit of "high cohesion and low coupling" of software systems increases the workload of software developers and the complexity of software systems as well. Therefore, the empirical evidence showed that within a certain range, reducing the coupling was helpful to attenuate the complexity of the software, while excessively blindly pursuit of low coupling increases the complexity of software systems.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 61602400). This work was also supported in part by the Key Research and Development Program of Hangzhou under Grant 20182011A46.

References

[1] W. P. Stevens, G. J. Myers, and L. L. Constantine, "Structured design," *IBM Systems Journal*, vol. 13, no. 2, pp. 115–139, 1974.

[2] W. P. Stevens, G. J. Myers, and L. L. Constantine, "Structured design," *IBM Systems Journal*, vol. 38, no. 2, pp. 231–256, 1999.

[3] E. Yourdon and L. L. Constantine, *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*, Yourdon Press, Englewood Cliffs, NJ, USA, 1979.

[4] L. C. Briand, J. W. Daly, and J. K. Wust, "A unified framework for coupling measurement in object-oriented systems," *IEEE Transactions on Software Engineering*, vol. 25, no. 1, pp. 91–121, 1999.

[5] M. D'Ambros, M. Lanza, and R. Robbes, "On the relationship between change coupling and software defects," in *Proceedings of the 2009 16th Working Conference on Reverse Engineering*, Lille, France, October 2009.

[6] T. Gyimothy, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," *IEEE Transactions on Software Engineering*, vol. 31, no. 10, pp. 897–910, 2005.

[7] P. Yu, T. Systa, and H. Muller, "Predicting fault-proneness using OO metrics. An industrial case study," in *Proceedings of the Sixth European Conference on Software Maintenance and Reengineering*, Budapest, Hungary, March 2002.

[8] L. C. Briand, J. Wuest, and H. Lounis, "Using coupling measurement for impact analysis in object-oriented systems,"

in *Proceedings of the IEEE International Conference on Software Maintenance*, Oxford, UK, September 1999.

[9] F. G. Wilkie and B. A. Kitchenham, "Coupling measures and change ripples in C++ application software," *Journal of Systems & Software*, vol. 52, no. 2-3, pp. 157–164, 2000.

[10] G. Antoniol, R. Fiutem, and L. Cristoforetti, "Using metrics to identify design patterns in object-oriented software," in *Proceedings of the Fifth International Software Metrics Symposium*, Bethesda, MD, USA, November 1998.

[11] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476–493, 1994.

[12] S. R. Chidamber and C. F. Kemerer, "Towards a metrics suite for object oriented design," in *Proceedings of the ACM Conference on Object Oriented Programming, Systems, Languages and Applications*, Orlando, FL, USA, 1991.

[13] W. Li and S. Henry, "Object-oriented metrics that predict maintainability," *Journal of Systems and Software*, vol. 23, no. 2, pp. 111–122, 1993.

[14] Y. Lee, "Measuring the coupling and cohesion of an object-oriented program based on information flow," in *Proceedings of the International Conference on Software Qualiy*, Maribor, Slovenia, 1995.

[15] L. Briand, P. Devanbu, and W. Melo, "An investigation into coupling measures for C++ software engineering," in *Proceedings of the 19th International Conference on Software engineering*, Boston, MA, USA, May 1997.

[16] H. Li and B. Li, "A pair of coupling metrics for software networks," *Journal of Systems Science and Complexity*, vol. 24, no. 1, pp. 51–60, 2011.

[17] E. Arisholm, L. C. Briand, and A. Foyen, *Dynamic Coupling Measurement for Object-Oriented Software*, IEEE Press, Piscataway, NJ, USA, 2004.

[18] J. K. Chhabra and V. Gupta, "A survey of dynamic software metrics," *Journal of Computer Science & Technology*, vol. 25, no. 5, pp. 1016–1029, 2010.

[19] D. Poshyvanyk and A. Marcus, "The conceptual coupling metrics for object-oriented systems," in *Proceedings of the 2006 22nd IEEE International Conference on Software Maintenance*, Philadelphia, PA; USA, September 2006.

[20] D. Poshyvanyk, A. Marcus, R. Ferenc, and T. Gyimóthy, "Using information retrieval based coupling measures for impact analysis," *Empirical Software Engineering*, vol. 14, no. 1, pp. 5–32, 2009.

[21] M. Gethers and D. Poshyvanyk, "Using relational topic models to capture coupling among classes in object-oriented software systems," in *Proceedings of the 2010 IEEE International Conference on Software Maintenance*, Timisoara, Romania, September 2010.

[22] H. Gall, K. Hajek, and M. Jazayeri, "Detection of logical coupling based on product release history," in *Proceedings of the International Conference on Software Maintenance*, Washington, DC, USA, April 1998.

[23] T. Zimmermann, A. Zeller, P. Weissgerber, and S. Diehl, "Mining version histories to guide software changes," *IEEE Transactions on Software Engineering*, vol. 31, no. 6, pp. 429–445, 2005.

[24] E. J. Weyuker, "Evaluating software complexity measures," *IEEE Transactions on Software Engineering*, vol. 14, no. 9, pp. 1357–1365, 1988.

[25] I. Vessey and R. Weber, "Research on structured programming: an empiricist's evaluation," *IEEE Transactions on Software Engineering*, vol. SE-10, no. 4, pp. 397–407, 2009.

[26] L. C. Briand, S. Morasca, and V. R. Basili, "Property-based software engineering measurement," *IEEE Transactions on Software Engineering*, vol. 22, no. 1, pp. 68–86, 1996.

[27] L. C. Briand, J. W. Daly, and J. Wüst, "A unified framework for cohesion measurement in object-oriented systems," *Empirical Software Engineering*, vol. 3, no. 1, pp. 65–117, 1998.

[28] D. J. Watts and S. H. Strogatz, "Collective dynamics of "small-world" networks," *Nature*, vol. 393, 1998.

[29] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[30] W. Pan, H. Ming, C. K. Chang, Z. Yang, and D.-K. Kim, "ElementRank: ranking java software classes and packages using multilayer complex network-based approach," *IEEE Transactions on Software Engineering*, 2019.

[31] W. Pan, B. Li, J. Liu, Y. Ma, and B. Hu, "Analyzing the structure of java software systems by weighted K-core decomposition," *Future Generation Computer Systems*, vol. 83, pp. 431–444, 2018.

[32] C. R. Myers, "Software systems as complex networks: structure, function, and evolvability of software collaboration graphs," *Physical Review E*, vol. 68, no. 4, 15 pages, 2003.

[33] N. LaBelle and E. Wallingford, "Inter package dependency networks in open source software," 2004, http://arxiv.org/abs/0411096.

[34] D. Hyland-Wood, D. Carrington, and S. Kaplan, "Scale-free nature of java software package," class and method collaboration graphs," Techical report no. TR-MS1286, University of Maryland College, College Park, MD, USA, 2006.

[35] Y. Xiang, W. Pan, H. Jiang, Y. Zhu, and H. Li, "Measuring software modularity based on software networks," *Entropy*, vol. 21, no. 4, p. 344, 2019.

[36] W. Pan and C. Chai, "Measuring software stability based on complex networks in software," *Cluster Computing*, vol. 22, no. S2, pp. 2589–2598, 2019.

[37] W. Pan, B. Song, K. Li, and K. Zhang, "Identifying key classes in object-oriented software using generalizedk-core decomposition," *Future Generation Computer Systems*, vol. 81, pp. 188–202, 2018.

[38] W. Pan and C. Chai, "Structure-aware mashup service clustering for cloud-based internet of things using genetic algorithm based clustering algorithm," *Future Generation Computer Systems*, vol. 87, pp. 267–277, 2018.

[39] W. Pan, J. Dong, K. Liu, and J. Wang, "Topology and topic-aware service clustering," *International Journal of Web Services Research*, vol. 15, no. 3, pp. 18–37, 2018.

[40] F. Liljeros, C. R. Edling, L. A. N. Amaral, H. E. Stanley, and Y. Åberg, "The web of human sexual contacts," *Nature*, vol. 411, no. 6840, pp. 907-908, 2001.

[41] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási, "The large-scale organization of metabolic networks," *Nature*, vol. 407, no. 6804, pp. 651–654, 2002.

[42] P.-P. Zhang, K. Kan Chen, Y. He et al., "Model and empirical study on some collaboration networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 360, no. 2, pp. 599–616, 2006.

[43] Q. Xuan, F. Du, and T. J. Wu, "Empirical analysis of internet telephone network : from user ID to phone," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 19, no. 2, Article ID 023101, 2009.

[44] M.-S. Shang, L. Lü, Y.-C. Zhang, and T. Zhou, "Empirical analysis of web-based user-object bipartite networks," *EPL (Europhysics Letters)*, vol. 90, no. 4, p. 48006, 2010.

[45] K.-I. Goh, M. E. Cusick, D. Valle, B. Childs, M. Vidal, and A.-L. Barabasi, "The human disease network," *Proceedings of*

the National Academy of Sciences, vol. 104, no. 21, pp. 8685–8690, 2007.

[46] B. Kitchenham, "What's up with software metrics?-a preliminary mapping study," *Journal of Systems and Software*, vol. 83, no. 1, pp. 37–51, 2010.

[47] J. Eder and M. Schrefl, "Coupling and cohesion in object-oriented systems," in *Proceedings of the International Workshop on Object Orientation in Operating Systems*, pp. 264–272, Dordan, France, September 1992.

[48] Illusion, 2012, http://sourceforge.net/projects/aoi/.

[49] JabRef, 2012, http://sourceforge.net/projects/jabref/.

[50] GanttProject, 2012, http://sourceforge.net/projects/ganttproject/.

[51] J. Al Dallal and L. C. Briand, "An object-oriented high-level design-based class cohesion metric," *Information and Software Technology*, vol. 52, no. 12, pp. 1346–1361, 2010.

[52] J. Al Dallal, "Measuring the discriminative power of object-oriented class cohesion metrics," *IEEE Transactions on Software Engineering*, vol. 37, pp. 778–804, 2011.

[53] J. Al Dallal, "The impact of accounting for special methods in the measurement of object-oriented class cohesion on refactoring and fault prediction activities," *Journal of Systems and Software*, vol. 85, no. 5, pp. 1042–1057, 2012.

[54] A. Gu, X. Zhou, Z. Li, Q. Li, and L. Li, "Measuring object-oriented class cohesion based on complex networks," *Arabian Journal for Science and Engineering*, vol. 42, no. 8, pp. 3551–3561, 2017.

Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 8906091, 15 pages
https://doi.org/10.1155/2020/8906091

Research Article

Study on the Thin Plate Model with Elastic Foundation Boundary of Overlying Strata for Backfill Mining

Dongdong Chen,[1] Xiaoyu Wu,[1] Shengrong Xie,[1] Yanding Sun,[1] Qing Zhang,[1] En Wang,[1] Yaohui Sun,[1] Long Wang,[1] Hui Li,[1] Zaisheng Jiang,[1] and Xiaowei Wu[2]

[1]*School of Energy and Mining Engineering, China University of Mining & Technology, Beijing 100083, China*
[2]*Yongding Zhuang Mine, Datong Coal Mine Group, Datong 037024, China*

Correspondence should be addressed to Shengrong Xie; xsrxcq@163.com

Received 15 December 2019; Accepted 13 February 2020; Published 17 March 2020

Academic Editor: Richard I. Avery

In order to better study the movement principles of overlying strata during backfill mining, we established a thin plate model on an elastic foundation with elastic foundation boundary of the main roof. And by the finite difference method, the variation principles of the main roof's principal moments and maximum subsidence ω_0 with the elastic foundation coefficient k_1 of the coal seam, the elastic foundation coefficient k_2 of backfill body, the thickness h, Young's modulus E, and Poisson's ratio μ of main roof are calculated and studied. Using these calculations, we were able to determine that the main roof had three principal bending moment extreme points, including M_{zz} in backfill areas, M_c of the long side area, and M_d of the short side area. The distance L_c of M_c advancing coal wall continuously increased with the increase in k_2, while the principal moment of main roof's middle area decreased with an increase in k_2; when k_2 became larger, the maximum principal moment in the midpoint of main roof transferred to the surrounding and the maximum principal moments was in four-corner area; M_c and M_d decreased with an increase in k_2, and M_d was more sensitive to k_2 than M_c; and M_d decreased significantly with the increase in k_2. L_c continuously decreased with the increase in k_1, while M_c, M_d and M_{zz} increased with the increase in k_1 and the reduced amplitude of M_{zz} was the minimum. The effect of μ on principal bending moments and ω_0 was very small; The growth rate of M_{zz} was the largest when E or h increased. M_d, M_{zz}, and L_c remained unchanged when k_1, k_2, and Young's modulus E of the main roof increased while the ratio value remained constant ($k_1/k_2/E$). Finally, the theoretical calculations were applied to the I26 backfill working face in the Xingdong mine to calculate the final subsidence amounts of the main roof. Field observations and theoretical calculations were about 48 mm, verifying the method's applicability.

1. Introduction

Strata movement gradually develops on the surface after underground coal seam mining, causing surface subsidence, destruction of ground facilities, and so on. Solid backfill mining, paste backfill mining, high-water backfill mining, and other backfill mining methods are commonly used to prevent surface subsidence caused by underground mining. Qian et al. [1] put forward the concept of green mining in coal mine and expounded the basic methods of mining settlement and backfilling control. Wu et al. [2], through image algorithms and so on, analyzed the mining landscape changes before and after subsidence with the support of GIS technology. According to the characteristics of mining subsidence, Jung et al. studied the comprehensive prediction and calculation methods related to mining subsidence, and so on [3–9]. In view of the subsidence characteristics of the surface steps caused by the mining of the shallow and extra thick coal seam, Ju and Xu [10] put forward three possible control methods for surface stepped subsidence. Based on the field investigation and study of overburden damage in ultra-thick coal seam mining, the statistical formula was presented to estimate the maximum heights of failure zone in the LTCC operation [11]. The characteristics of displacement and ground subsidence caused by underground mining are studied by means of remote sensing and

Geographic Information System [12–14]. Li et al. studied the mechanical characteristics of the gangue filling body, the mechanical and geometric characteristics of the hydraulic support for gangue filling, and the characteristics of the mining pressure in the solid filling face; they obtained that the solid backfilling method was an effective method to prevent hard-roof-induced face bursts, and the equivalent mining height model is capable of predicting surface deformation [15–20]. Benzaazoua et al. studied, respectively, the hardening process of cemented backfill, the preparation, microstructure characteristics, mechanical parameters, and backfilling technology of the paste backfill materials [21–24]. Based on the basic mechanical experiments of ultra-high-water materials, Ding et al. [25] found that it was good backfilling material. Yan et al. [26] put forward the pump filling technology of ZKD high-water quick-setting material, which was effectively applied to engineering practice. Feng et al. [27] introduced four kinds of backfilling methods for goaf with high water content materials and analyzed the advantages and disadvantages of each method.

There have been many studies on the backfill mining process and issues that may arise. Zhang et al. [28] analyzed the interaction between the backfilling body and overburden strata in a fully mechanized backfilling mining face. Miao et al. [29] proposed the beam model on an elastic foundation with a fixed boundary condition to study the relationship between subsidence of the main roof and Young's modulus of backfill body. Chen et al. [30] built a beam model on an elastic foundation with an elastic foundation boundary to analyze the variation of the main roof's maximum subsidence with Young's modulus of backfill body and elastic foundation coefficient of coal seam. Li et al. [31] proposed a thin plate model on an elastic foundation of the main roof with a fixed boundary condition to study the variation of the main roof's subsidence and maximum tensile stress with an elastic foundation coefficient of the backfill body. Huang et al. [32] analyzed overlying strata movement in backfill mining using a similar physical simulation, and although these results are beneficial for scientific backfill mining, there are specific scenarios that they are applicable. The beam model in backfill mining is only suitable for roof mechanics analysis of the middle area in the working face with large length-width ratio.

The thin plate model with the main roof being elastic and fixed boundary condition for backfill mining simplifies the main roof in order for it to be a fixed boundary condition. However, the main roof is bound to sink in the coal seam support area when the coal seam is thick and soft, making the fixed boundary condition unsuitable. The Winkler elastic foundation model should therefore be considered because of the weak shear-bearing capacity and non-deformability of the coal seam [33]. The main roof can then be regarded as an elastic foundation, and the elastic foundation model is able to analyze its mechanics and displacement characteristics during backfill mining.

Here, we established a thin plate model on elastic foundation of the main roof with an elastic foundation boundary to study the variations of the internal force field and displacement field with various elastic foundation coefficients of the backfill body, elastic foundation coefficients of the coal seam, thicknesses of main roof, Young's modulus of the main roof, and Poisson's ratio of the main roof, and the fracture position and fracture conditions of the main roof were then determined and compared. Finally, a determination method for the elastic foundation coefficient of the backfill body is proposed in this paper. These theoretical calculations were then verified using the I26 mining face from Xingdong coal mine. These results have practical value in further developments in backfill mining technology.

2. The Thin Plate Model on Elastic Foundation with the Elastic Foundation Boundary in Backfill Mining

2.1. The Actual Surrounding Rock Condition of the Main Roof in Backfill Mining. The main roof is clamped between the overlying strata and immediate roof after mining. Due to the large compressive deformation and the weak shear-bearing capacity of the coal seam, the coal seam is best represented by the Winkler elastic foundation model. Similarly, the backfill body is also best represented by the Winkler elastic foundation model. Figure 1 shows the surrounding rock relationship during backfill mining.

2.2. Mechanical Hypothesis Condition of the Main Roof in Backfill Mining

2.2.1. Hypothesis of Elastic Foundation Boundary. For this study, the coal seam is analyzed using the Winkler elastic foundation model. The displacement of the main roof is mainly limited by the immediate roof and coal seam. In general, Young's modulus of the coal seam is much lower than that of the main roof and immediate roof; therefore, the coal seam is the main factor that limits the displacement of the main roof. The elastic foundation coefficient k_0 can be derived from equation (1) and is given by

$$k_0 = \frac{E_1 E_0}{E_1 h_0 + E_0 h_1} \approx \frac{E_1}{h_1} = k_1, \quad (1)$$

where k_0 is the composite elastic foundation coefficient of the immediate roof and coal seam; k_1 is the elastic foundation coefficient of the coal seam; h_0 is the thickness of the immediate roof; E_0 is Young's modulus of the immediate roof; h_1 is the thickness of the coal seam; E_1 is Young's modulus of the coal seam.

2.2.2. Hypothesis of Elastic Foundation of the Backfill Body. Solid backfill material, paste backfill material, and high-water backfill material approximately satisfy the Winkler elastic foundation model.

2.2.3. Dimension Requirement of the Elastic Thin Plate of the Main Roof. In order to satisfy the requirements to be considered a "thin plate," the dimensions of the roof must meet the requirement of

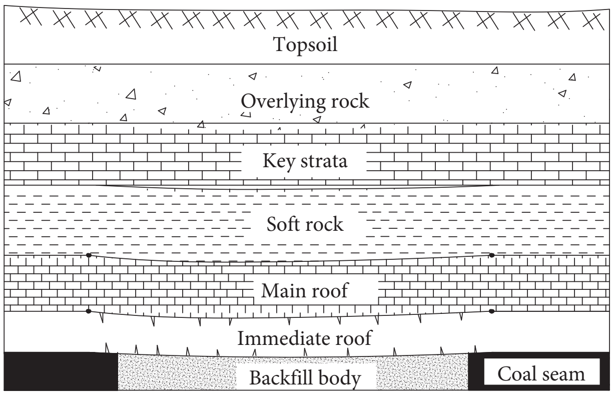


FIGURE 1: Depiction of the surrounding rock in backfill mining.

$$\left(\frac{1}{8} \sim \frac{1}{5}\right) \geq \frac{h}{L} \geq \left(\frac{1}{100} \sim \frac{1}{80}\right), \tag{2}$$

where h is the thickness of the plate; L is the short side length of the plate.

When h and L of the plate satisfy equation (2), the plate can be seen as a thin plate.

In general, the ratio of the thickness of the roof to the short side length satisfies equation (2); that is, the main roof satisfies dimension requirements of the elastic thin plate.

2.3. Mechanical Model of the Thin Plate on Elastic Foundation with Elastic Foundation Boundary in Backfill Mining

2.3.1. Mechanical Model. The mechanical model of a thin plate on an elastic foundation with an elastic foundation boundary of a main roof in backfill mining is shown in Figure 2.

The rectangle area ABCD is the backfill area S_2. The length of AB is $2a$ and the length of AD is $2b$. A coordinate system is established by using the midpoint of the main roof on the backfill body as the origin. Area S_1 of the main roof, which is outside of ABCD and inside of $A_1B_1C_1D_1$, is approximately the clamping area of the elastic layer. The displacement of the main roof in area S_1 is mainly determined by the stiffness of the coal seam. The load carried by the main roof is q_0.

2.3.2. Deflection Equation. According to the theory of a plate [34–36], the partial differential equation of the deflection of the main roof in an elastic foundation area S_1 can be defined as

$$\frac{\partial^4 \omega_1}{\partial x^4} + 2\frac{\partial^4 \omega_1}{\partial x^2 \partial y^2} + \frac{\partial^4 \omega_1}{\partial y^4} = \frac{1}{D}\left(-k_1 \omega_1\right), \tag{3}$$

where ω_1 is the deflection function of the main roof in the area S_1; and k_1 is the elastic foundation coefficient of the coal seam. The stiffness of a thin plate is $D = Eh^3/(12\text{–}12\mu^2)$, where E is Young's modulus of the main roof; h is the thickness of the main roof; and μ is Poisson's ratio of the main roof.

The partial differential equation of deflection of the main roof in the backfill area S_2 can be defined as

$$\frac{\partial^4 \omega_2}{\partial x^4} + 2\frac{\partial^4 \omega_2}{\partial x^2 \partial y^2} + \frac{\partial^4 \omega_2}{\partial y^4} = \frac{1}{D}\left(q_0 - k_2 \omega_2\right), \tag{4}$$

where ω_2 is the deflection function of the main roof in area S_2; q_0 is the load carried by the main roof; and k_2 is the elastic foundation coefficient of backfill body.

2.3.3. Boundary Conditions

(1) Continuous Boundary Condition. Edges AB, BC, CD, and AD are the main roof's interface between the coal body and the backfill body. Therefore, deflection, rotation angle, bending moment, and shear force are the same on the interface, and a continuous condition can be achieved by combining all the relevant equations as equations (5) and (6). A continuous boundary condition for the main roof can, therefore, be expressed as

$$\begin{cases} \begin{cases} -a \leq x \leq a, \\ y = b, \end{cases} & \begin{cases} \dfrac{\partial}{\partial y}\nabla^2 \omega_1 = \dfrac{\partial}{\partial y}\nabla^2 \omega_2, \\ \dfrac{\partial^2 \omega_1}{\partial y^2} + \mu \dfrac{\partial^2 \omega_1}{\partial x^2} = \dfrac{\partial^2 \omega_2}{\partial y^2} + \mu \dfrac{\partial^2 \omega_2}{\partial x^2}, \\ \dfrac{\partial \omega_1}{\partial y} = \dfrac{\partial \omega_2}{\partial y}, \\ \omega_1(x, b) = \omega_2(x, b), \end{cases} \\ \begin{cases} -a \leq x \leq a, \\ y = -b, \end{cases} & \begin{cases} \dfrac{\partial}{\partial y}\nabla^2 \omega_1 = \dfrac{\partial}{\partial y}\nabla^2 \omega_2, \\ \dfrac{\partial^2 \omega_1}{\partial y^2} + \mu \dfrac{\partial^2 \omega_1}{\partial x^2} = \dfrac{\partial^2 \omega_2}{\partial y^2} + \mu \dfrac{\partial^2 \omega_2}{\partial x^2}, \\ \dfrac{\partial \omega_1}{\partial y} = \dfrac{\partial \omega_2}{\partial y}, \\ \omega_1(x, -b) = \omega_2(x, -b), \end{cases} \end{cases} \tag{5}$$

$$\begin{cases} \begin{cases} -b \le y \le b, \\ x = a, \end{cases} & \begin{cases} \dfrac{\partial}{\partial x}\nabla^2 \omega_1 = \dfrac{\partial}{\partial x}\nabla^2 \omega_2, \\ \dfrac{\partial^2 \omega_1}{\partial x^2} + \mu \dfrac{\partial^2 \omega_1}{\partial y^2} = \dfrac{\partial^2 \omega_2}{\partial x^2} + \mu \dfrac{\partial^2 \omega_2}{\partial y^2}, \\ \dfrac{\partial \omega_1}{\partial x} = \dfrac{\partial \omega_2}{\partial x}, \\ \omega_1 (a, y) = \omega_2 (a, y), \end{cases} \\ \begin{cases} -b \le y \le b, \\ x = -a, \end{cases} & \begin{cases} \dfrac{\partial}{\partial x}\nabla^2 \omega_1 = \dfrac{\partial}{\partial x}\nabla^2 \omega_2, \\ \dfrac{\partial^2 \omega_1}{\partial x^2} + \mu \dfrac{\partial^2 \omega_1}{\partial y^2} = \dfrac{\partial^2 \omega_2}{\partial x^2} + \mu \dfrac{\partial^2 \omega_2}{\partial y^2}, \\ \dfrac{\partial \omega_1}{\partial x} = \dfrac{\partial \omega_2}{\partial x}, \\ \omega_1 (-a, y) = \omega_2 (-a, y). \end{cases} \end{cases} \tag{6}$$

(2) Boundary Condition. A_1B_1 length is $2x_0$. A_1D_1 length is $2y_0$ (refer to Figure 2). At an infinite distance from the backfill area S_2, the mining effect is very small. Therefore, the deflection and rotation angle of the main roof are both zero and satisfy the fixed boundary conditions at an infinite distance from the backfill area S_2. The fixed boundary condition can then be defined as

$$\begin{cases} \begin{cases} x = -x_0 \to -\infty, \\ -y_0 \le y \le y_0, & \omega_1 = 0, \ \dfrac{\partial \omega_1}{\partial x} = 0, \end{cases} \\ \begin{cases} x = x_0 \to +\infty, \\ -y_0 \le y \le y_0, & \omega_1 = 0, \ \dfrac{\partial \omega_1}{\partial x} = 0, \end{cases} \\ \begin{cases} y = -y_0 \to -\infty, \\ -x_0 \le x \le x_0, & \omega_1 = 0, \ \dfrac{\partial \omega_1}{\partial y} = 0, \end{cases} \\ \begin{cases} y = y_0 \to +\infty, \\ -x_0 \le x \le x_0, & \omega_1 = 0, \ \dfrac{\partial \omega_1}{\partial y} = 0. \end{cases} \end{cases} \tag{7}$$

3. The Finite Difference Method for Solving the Partial Differential Equations

As mentioned above, it is extremely difficult to obtain the exact solutions of the above partial differential equations, but the approximate solutions satisfy engineering and practical mining requirement. The finite difference method is an effective method to obtain the approximate solutions of differential equations [37–39].

3.1. The Finite Difference Method

3.1.1. The Nodal Layout of the Difference Equation. According to the finite difference theory, the difference equation involving 13 nodes is needed to solve the partial differential equation [34].

The difference grid layout of 13 nodes is shown in Figure 3. $\Delta x = \Delta y = d$ is the nodal space. Point J_0 is a feature node. Deflection of point J_0 is expressed as ω_{ij}. Numbers of the remaining nodes are determined by the intersection of the vertical and horizontal lines (refer to Figure 3).

3.1.2. The Difference Equations of the Partial Differential Equations. Combining with the difference grid layout and the finite difference method of the partial differential equations (refer to Figure 3), the difference equations of the partial differential equations (3) and (4) at feature node J_0 can be obtained by

$$\begin{aligned} &\left(20 + d^4 \frac{k_1}{D}\right)\omega_{ij} - 8\left(\omega_{i+1,j} + \omega_{i-1,j} + \omega_{i,j+1} + \omega_{i,j-1}\right) \\ &\quad + 2\left(\omega_{i+1,j+1} + \omega_{i+1,j-1} + \omega_{i-1,j+1} + \omega_{i-1,j-1}\right) \\ &\quad + \omega_{i+2,j} + \omega_{i-2,j} + \omega_{i,j+2} + \omega_{i,j-2} = 0, \end{aligned} \tag{8}$$

where d is the nodal space; ω_{ij}, $\omega_{i+1,j}$, $\omega_{i-1,j}$, $\omega_{i,j+1}$, $\omega_{i,j-1}$, $\omega_{i+1,j+1}$, $\omega_{i+1,j-1}$, $\omega_{i-1,j+1}$, $\omega_{i-1,j-1}$, $\omega_{i+2,j}$, $\omega_{i-2,j}$, $\omega_{i,j+2}$ and $\omega_{i,j-2}$ are nodal deflection, respectively (refer to Figure 3).

$$\begin{aligned} &\left(20 + d^4 \frac{k_2}{D}\right)\omega_{ij} - 8\left(\omega_{i+1} + \omega_{i-1,j} + \omega_{i,j+1} + \omega_{i,j-1}\right) \\ &\quad + 2\left(\omega_{i+1j+1} + \omega_{i+1,j-1} + \omega_{i-1,j+1} + \omega_{i-1,j-1}\right) + \omega_{i+2,j} \\ &\quad + \omega_{i-2,j} + \omega_{i,j+2} + \omega_{i,j-2} = \frac{q_0 d^4}{D}. \end{aligned} \tag{9}$$

3.1.3. The Difference Equations of Boundary Condition. Theoretically, the exact fixed boundary condition can be satisfied when x_0 and y_0 approach infinity (refer to equation (7)), but the finite difference method cannot calculate this infinite region. Actually, when the outer boundary is three times the maximum length of the mining area ($y_0 = x_0 = 3\max\{2a,\ 2b\}$), the influence of mining on this boundary is very weak, which approximately satisfies both the fixed boundary condition and engineering requirements. Nodal space d is taken as 0.2 m. The difference equations of the boundary condition in feature node J_0 can be expressed as

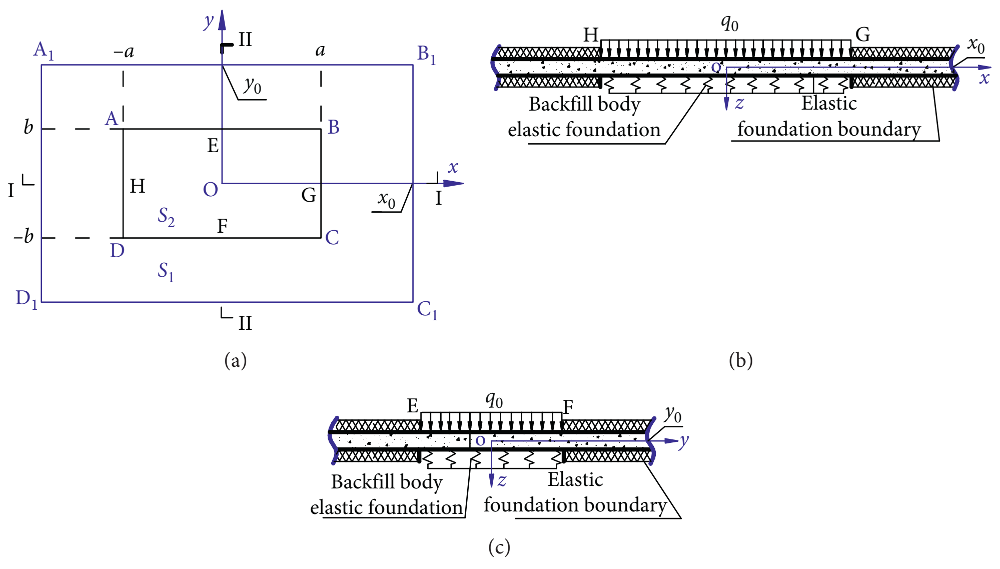


Figure 2: The thin plate model on an elastic foundation with an elastic foundation boundary. (a) Vertical view; (b) I-I section plane; (c) II-II section plane.

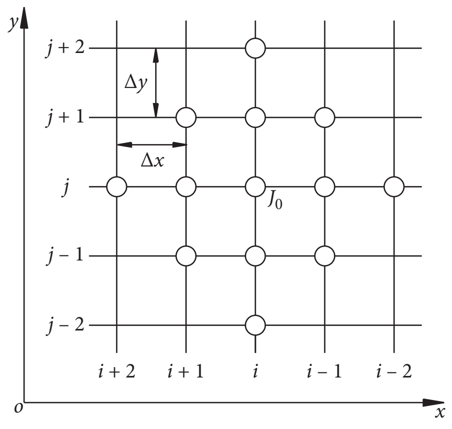


Figure 3: The difference grid layout of 13 nodes.

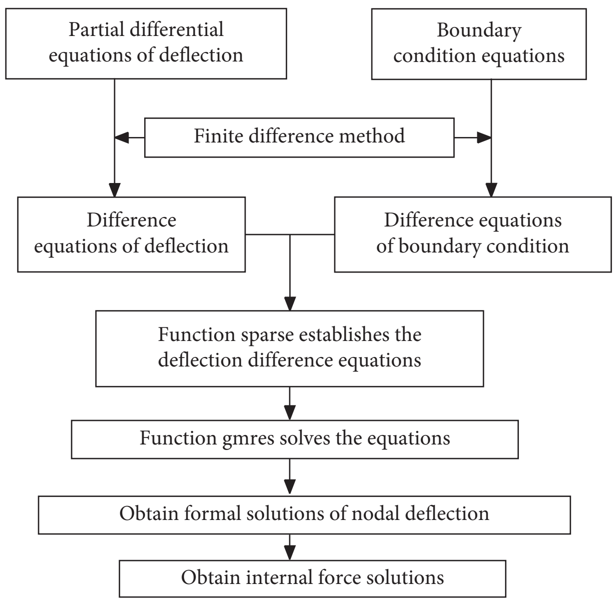

Partial differential equations of deflection
Boundary condition equations
Finite difference method
Difference equations of deflection
Difference equations of boundary condition
Function sparse establishes the deflection difference equations
Function gmres solves the equations
Obtain formal solutions of nodal deflection
Obtain internal force solutions

Figure 4: Process of solving partial differential equations.

$$\begin{cases} \begin{cases} -y_0 \le y \le y_0, \\ x = -x_0, \end{cases} \begin{cases} \omega_{i,j} = 0, \\ \left(\frac{\partial \omega}{\partial x}\right)_{ij} = \dfrac{\omega_{i-1,j} - \omega_{i+1,j}}{2d} = 0, \end{cases} \\ \begin{cases} -y_0 \le y \le y_0, \\ x = x_0, \end{cases} \begin{cases} \omega_{i,j} = 0, \\ \left(\frac{\partial \omega}{\partial x}\right)_{ij} = \dfrac{\omega_{i-1,j} - \omega_{i+1,j}}{2d} = 0, \end{cases} \\ \begin{cases} -x_0 \le x \le x_0, \\ y = y_0, \end{cases} \begin{cases} \omega_{i,j} = 0, \\ \left(\frac{\partial \omega}{\partial y}\right)_{ij} = \dfrac{\omega_{i,j-1} - \omega_{i,j+1}}{2d} = 0, \end{cases} \\ \begin{cases} -x_0 \le x \le x_0, \\ y = -y_0, \end{cases} \begin{cases} \omega_{i,j} = 0, \\ \left(\frac{\partial \omega}{\partial y}\right)_{ij} = \dfrac{\omega_{i,j-1} - \omega_{i,j+1}}{2d} = 0. \end{cases} \end{cases} \tag{10}$$

3.2. Calculation Process. According to the finite difference method, the 13-node difference equation should be established for any node whose deflection is unknown within the area of S_1 and the S_2 area of the main roof. The nodes whose deflection is unknown can then be obtained by constructing all nodal difference equations and the boundary condition equations.

Figure 4 presents the specific process of solving the partial differential equations. The process of solving partial differential equations needs to adopt the Sparse function in Matlab to form algebraic equations for a sparse matrix, and each nodal deflection solution can be obtained by using function Gmres [37] to solve the algebraic equations.

After solving each nodal deflection of the main roof, the nodal internal force solutions can be obtained by

$$\begin{cases} (M_x)_{ij} = -D\left(\frac{\partial^2 \omega}{\partial x^2} + \mu \frac{\partial^2 \omega}{\partial y^2}\right)_{ij} \\ = -\dfrac{D}{d^2}\left[\left(\omega_{i-1,j} - 2\omega_{i,j} + \omega_{i+1,j}\right) - \mu\left(\omega_{i,j-1} - 2\omega_{i,j} + \omega_{i,j+1}\right)\right], \\ \left(M_y\right)_{ij} = -D\left(\frac{\partial^2 \omega}{\partial y^2} + \mu \frac{\partial^2 \omega}{\partial x^2}\right)_{ij} \\ = -\dfrac{D}{d^2}\left[\left(\omega_{i,j-1} - 2\omega_{i,j} + \omega_{i,j+1}\right) - \mu\left(\omega_{i-1,j} - 2\omega_{i,j} + \omega_{i+1,j}\right)\right], \\ \left(M_{xy}\right)_{ij} = -D(1-\mu)\left(\frac{\partial^2 \omega}{\partial x \partial y}\right)_{ij} \\ = -\dfrac{D(1-\mu)}{4d^2}\left(\omega_{i-1,j-1} - \omega_{i+1,-1} + \omega_{i+1,j+1} - \omega_{i-1,j+1}\right), \end{cases} \tag{11}$$

where $(M_x)_{ij}$ is the nodal moment of the x-component; $(M_y)_{ij}$ is the nodal moment of the y-component; and $(M_{xy})_{ij}$ is the nodal twisting moment.

4. Example and Analysis

The thin plate model on an elastic foundation with an elastic foundation boundary in backfill mining was investigated under realistic geological conditions. Variations of the internal force and deflection of the main roof with E, h, μ, k_1 and k_2 were studied using the above equations for theoretical calculations.

The I26 working face of the Xingdong mine (length AB = 70 m and advancing length AD = 200 m in Figure 2) adopted the backfill bag of high-water material to backfill mining, with a backfill rate of 100%. This working face was used to obtain realistic conditions for the calculations. Based on the above analysis, when $A_1B_1 = A_1D_1 = 600$ m, the outer boundary can be regarded as fixed boundary.

In the I26 working face, the average angle of the coal seam was 4–6°, mining height was 4.5 m, and the thickness of the main roof was 10 m. Young's modulus of the main roof was 26 Gpa and the elastic foundation coefficient k_1 was 1000 MN/m^3. The key rock thickness, which overlies the main roof, was 110 m and the average unit weight of the rock was 0.024 MN/m^3. Therefore, the load carried by the main roof was $q_0 = 2.64$ Mpa. The elastic foundation coefficient k_2 of the high-water material used during the backfill mining process was 20 MN/m^3–220 MN/m^3.

4.1. Fracture Criterion.

$$M_1, M_3 = \frac{M_x + M_y}{2} \pm \sqrt{\left(\frac{M_x - M_y}{2}\right)^2 + \left(M_{xy}\right)^2}, \tag{12}$$

where M_1 is the maximum principal bending moment; M_3 is the minimum principal bending moment; M_x is the moment of x-component; M_y is the moment of y-component; and M_{xy} is the nodal twisting moment.

$$\left.\begin{matrix} (M_1)_{ij} \\ (M_3)_{ij} \end{matrix}\right\} = \frac{(M_x)_{ij} + \left(M_y\right)_{ij}}{2} \pm \sqrt{\left(\frac{(M_x)_{ij} - \left(M_y\right)_{ij}}{2}\right)^2 + \left(M_{xy}\right)_{ij}^2}, \tag{13}$$

where $(M_1)_{ij}$ is the nodal maximum principal moment; $(M_3)_{ij}$ is the nodal minimum principal moment.

Because the tensile strength of the rock is much less than the compressive strength, the principal moment of rock is compared with the ultimate bending moment of rock to judge whether the rock has failed or not. The maximum principal moment M_1 and minimum principal moment M_3 are obtained by equation (12), and the difference equation of equation (12) is equation (13). The moments $(M_x)_{ij}$, $(M_y)_{ij}$ and $(M_{xy})_{ij}$ in equation (13) can be obtained by equation (11). The extreme points of the principal moment are extracted from the cloud chart of the principal moment, which are shown in Figure 5.

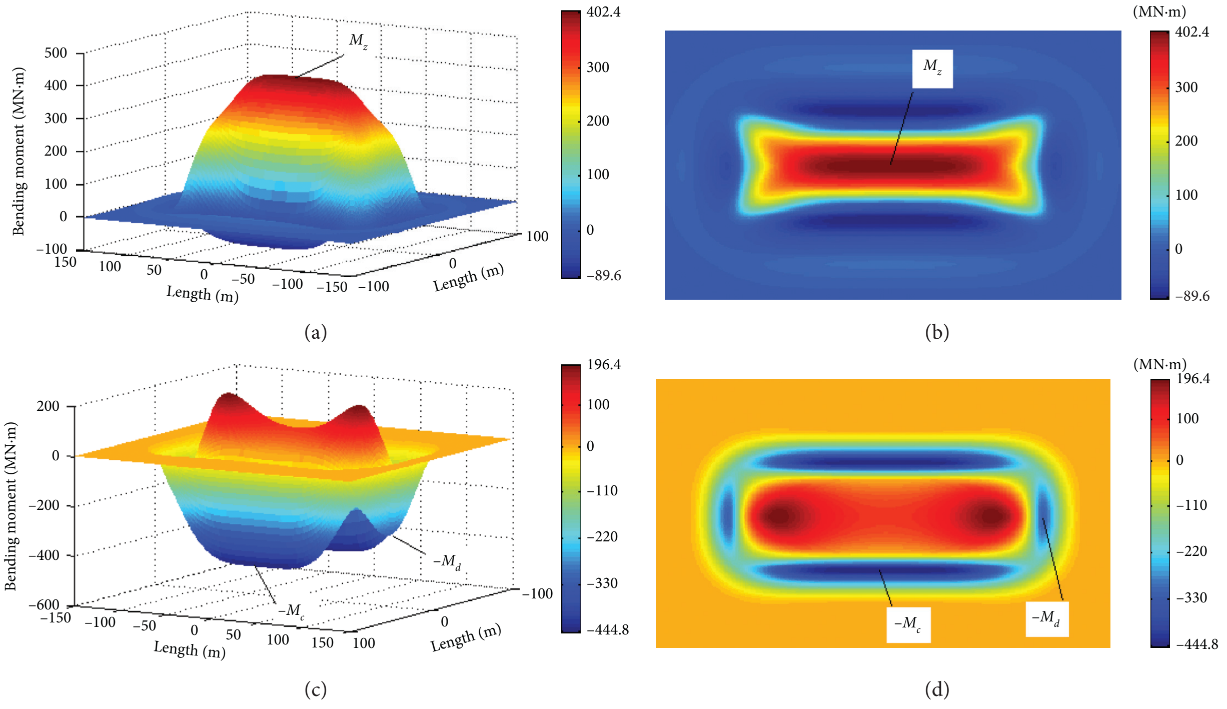


FIGURE 5: Principal moment feature when $k_2 = 20\,\text{MN/m}^3$. (a) Three-dimensional cloud chart of principal moment M_1; (b) plane cloud chart of principal moment M_1; (c) cloud chart of principal moment M_3; (d) plane cloud chart of principal moment M_3.

4.2. *Effect of Elastic Foundation Coefficient k_2 of Backfill Body in Backfill Mining*

4.2.1. Basic Principles and Distribution Pattern of the Principal Moment. The elastic foundation coefficient of the backfill body is the key material factor to decide the backfill effect. Here, the backfill rate was 100%. The elastic foundation coefficient k_2 was ~$20\,\text{MN/m}^3$–$220\,\text{MN/m}^3$. When $k_2 = 20\,\text{MN/m}^3$ and $k_2 = 180\,\text{MN/m}^3$, cloud charts of the principal moment in backfill mining were plotted, as shown in Figures 5 and 6.

From Figures 5 and 6, we can obtain the distribution of the principal moment during backfill mining: principal moments M_1 and M_3 in the middle area of the main roof directly overlying the backfill body were all positive bending moments; namely, the lower surface of the main roof bores the tensile stress and the upper surface of the main roof bores the compressive stress in this area. The nodal principal moments M_3 on the long side and short side area with a certain distance to the coal wall (L_c) were negative bending moments; namely, the upper surface of the main roof bores the tensile stress and the lower surface of main roof bores the compressive stress in this area.

According to the distribution characteristics of the principal moment in Figures 5 and 6, the coordinates of the extreme points of the main roof's principal moment were plotted, as shown in Figure 7.

When the value of k_2 was smaller ($k_2 = 20\,\text{MN/m}^3$), and according to the distribution characteristics of the principal moment in Figure 5, the coordinates of the extreme points of the principal moment were plotted, as shown in Figure 7(a). The maximum absolute value of the principal moment of main roof in backfill area was at the midpoint with a coordinate of (0, 0), which was set to M_z, and $M_z = M_1|_{(0,\,0)}$; the maximum absolute value of the principal moment on the long side area with a certain distance to the coal wall (L_c) was the negative value of the minimum principal moment M_3, which was set to M_c, the coordinates of which were $(0, b + L_c)$ and $(0, -b-L_c)$. The distance of the M_c advancing coal wall was set to L_c, and $M_c = -M_3|_{(0,b+L_c)} = -M_3|_{(0,-b-L_c)}$; the maximum absolute value of the principal moment on the short side area with a certain distance to the coal wall (L_c) was the negative value of the minimum principal moment M_3, which was set to M_d, and the coordinates of M_d are $(a + L_d, 0)$ and $(-a-L_d, 0)$. The distance of M_d advancing coal wall was set to L_d, and $M_d = -M_3|_{(a+L_d,0)} = -M_3|_{(-a-L_d,0)}$.

The coordinates of the extreme points of the principal moment from Figure 6 were plotted when k_2 was larger ($k_2 = 180\,\text{MN/m}^3$), as shown in Figure 7(b). The maximum principal moment of main roof in backfill area was set to M_{zc} whose coordinates were $(-a + L_{dc}, b-L_{cc})$ and because of the symmetry, there were four points whose moments were all taken as $M_{zc}\,(M_{zc}\,|_{(-a+L_{dc},b-L_{cc})} > M_z = M_1\,|_{(0,0)})$. In Figure 7(b), the maximum absolute value of the principal moment of the long side area with a certain distance to the coal wall (L_c) was set to M_c, and $M_c = -M_3|_{(0,b+L_c)} = -M_3|_{(0,-b-L_c)}$; the maximum absolute value of the principal moment of the short side area with a certain distance to the coal wall (L_c) wall was set to M_d ($M_d = -M_3|_{(a+L_d,\,0)} = -M_3|_{(-b-L_c,\,0)}$).

As shown in Figures 5 and 6, when the value of k_2 became large ($k_2 = 180\,\text{MN/m}^3$), the maximum principal

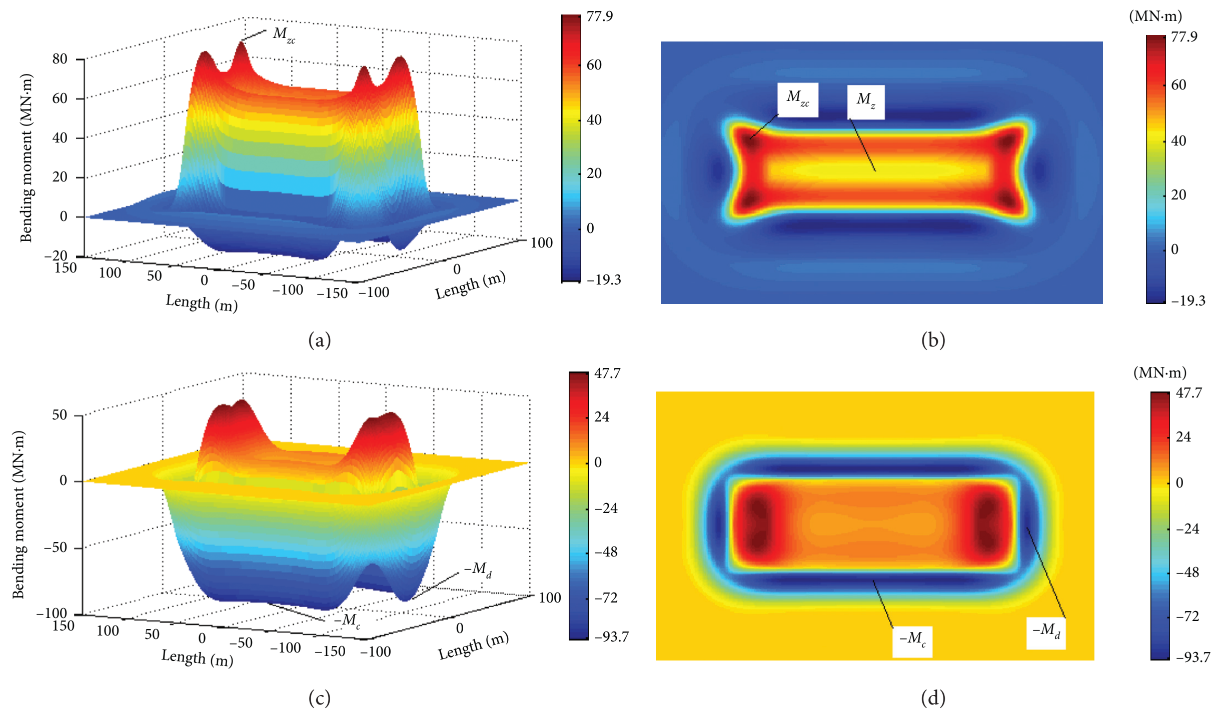


Figure 6: Principal moment feature when $k_2 = 180\,\text{MN/m}^3$. (a) Three-dimensional cloud chart of principal moment M_1; (b) plane cloud chart of principal moment M_1; (c) three-dimensional cloud chart of principal moment M_3; (d) plane cloud chart of principal moment M_3.

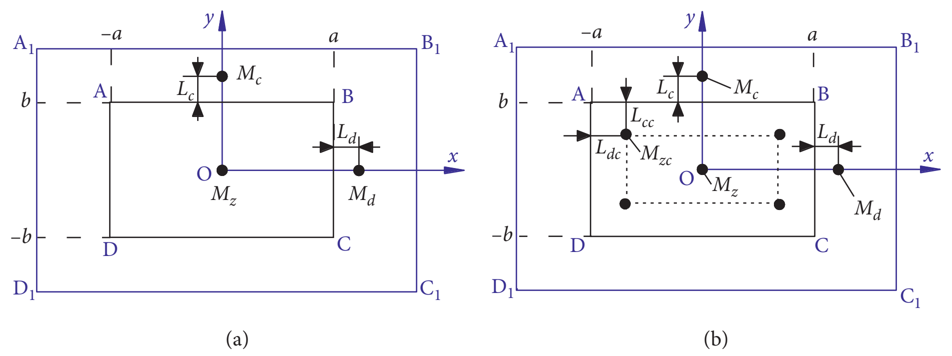


Figure 7: The coordinate of the extreme points of the main roof principal moment during backfill mining: (a) $k_2 = 20\,\text{MN/m}^3$; (b) $k_2 = 180\,\text{MN/m}^3$.

moment M_z in the midpoint of the main roof transferred to the surrounding area and the points of the maximum principal moment were in the four corners of the study area. The maximum absolute value of the principal moment of main roof in the backfill area is denoted as M_{zz}, and M_{zz} meets the following relationship: $M_{zz} = \max\{M_1|_{(0,0)}, M_{zc}|_{(-a+L_{dc},b-L_{cc})}\}$

Based on the above analysis, there appear to be three types of extreme points of the principal moments: the maximum absolute value of the principal moment M_{zz} of the main roof in backfill area, the maximum absolute value of the principal moment M_c in the long side area, and the maximum absolute value of the principal moment M_d in the short side area. Based on the above analysis and rock tensile properties, the lower surface of the middle area of main roof and the upper surface of long side and short side areas in front of the coal wall are the most susceptible to fracturing. Using the plate model on an elastic foundation with an elastic foundation boundary during backfill mining, the principles governing the fracturing of main roof are determined by the values of M_c, M_{zz} and M_d. The main roof can only fracture when the principal moment is larger than M_s (the ultimate bending moment of the main roof).

During backfill mining, the principal moments and the maximum subsidence ω_0 of the main roof are related to k_2, k_1, h, E and μ. The principal movements and L_c were then calculated with variations in the above mentioned variables and analyzed in order to study the variations in the maximum subsidence of the main roof.

4.2.2. Effects of the Elastic Foundation Coefficient k_2 of the Backfill Body. The following calculation examines the effect of the elastic foundation coefficient k_2 on the principal bending moment and maximum subsidence ω_0 of the main roof.

(1) The variation of the principal bending moment of the main roof and L_c with the elastic foundation coefficient k_2 of the backfill body

Figure 8 shows the variations of the principal bending moments of the main roof and L_c with the elastic foundation coefficient k_2 of the backfill body. The backfill body significantly reduced the value of the maximum principal bending moment of the main roof and improved the stability of the main roof. The principal bending moments (M_c, M_d and M_{zz}) continuously decreased with the increase in values of k_2. The value of the principal bending moment M_{zz} was highly sensitive to k_2, and comparatively the principal bending moment M_d was less sensitive. The principal bending moments M_c and M_d were finally almost equal with an increase in k_2. In this situation, the long side and short side of the main roof almost fractured simultaneously when the maximum principal bending moment of the main roof increased to the ultimate bending moment M_s. There should be very little decrease in the principal bending moments when k_2 increases to some extent.

The initial fracture position of the main roof was in the long side area, which is located in front of the coal wall with a M_c value greater than M_s. L_c continuously increased with the increase in k_2. Specifically, when M_c was greater than M_s, the larger the values of k_2, the further the distance between the fracture line of main roof and the coal wall (refer to Figure 8).

(2) Variation of the maximum subsidence ω_0 of the main roof with an elastic foundation coefficient k_2 of the backfill body

Figure 9 shows the variation of the maximum subsidence ω_0 of the main roof with an elastic foundation coefficient k_2 of the backfill body. The maximum subsidence ω_0 of the main roof appeared to decrease with the increase in k_2. The maximum subsidence should decrease very little when k_2 increases to some extent.

Using the above calculations and formulas, further study should be conducted to further describe the principal bending moments and the maximum subsidence ω_0 of main roof with an elastic foundation coefficient k_1 by varying the coal seam, thickness h of the main roof, Young's modulus E of the main roof, and Poisson's ratio μ of the main roof, in order to better and widely study fracture principles and subsidence of the main roof during backfill mining.

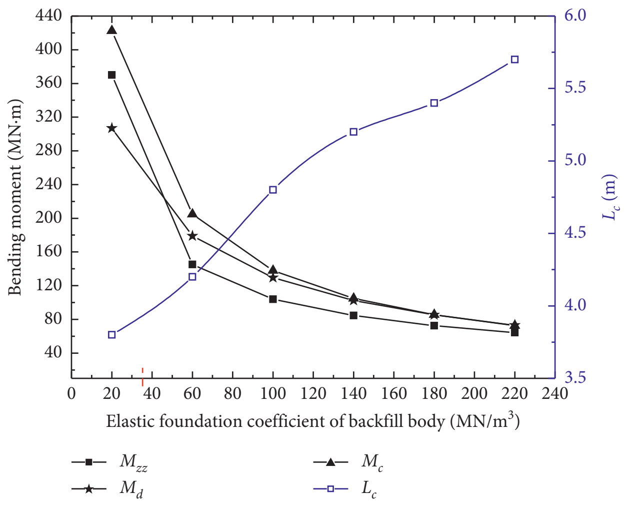


Figure 8: Change in the principal bending moments and L_c with elastic foundation coefficient k_2 of backfill body.

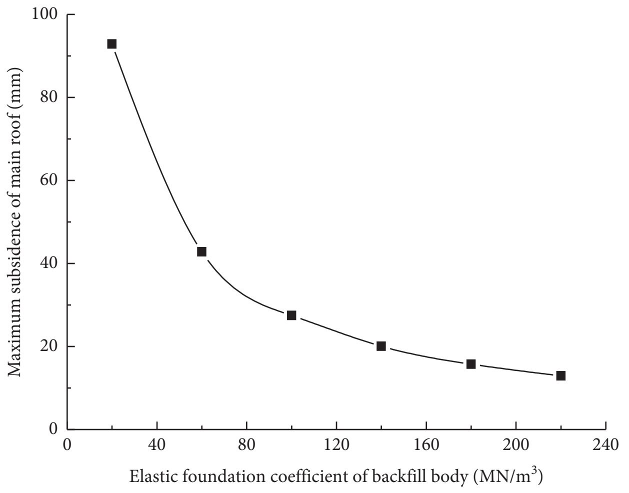


Figure 9: Change in the maximum subsidence ω_0 of the main roof with variable k_2.

4.3. Effect of the Elastic Foundation Coefficient k_1 of the Coal Seam in Backfill Mining. The elastic foundation coefficient k_1 of the coal seam mainly reflects the limitation of the displacement of the main roof. We then analyzed variations of the principal bending moments and the maximum subsidence ω_0 with an elastic foundation coefficient k_1.

(1) Variations in the principal bending moments and L_c with k_1.

Variations in the principal bending moments of the main roof and the distance to the coal wall L_c with an elastic foundation coefficient k_1 are presented in

Figure 10. The principal bending moments (M_c, M_d and M_{zz}) apparently increased with the increase in k_1, and comparatively the principal bending moment M_{zz} had a smaller growth rate. When the principal moment M_c was greater than the ultimate moment M_s, the larger the value of k_1 (the harder the coal seam), the closer the distance between the fracture line of the main roof and the coal wall (refer to Figure 10). Therefore, the influence of the boundary condition on the principal bending moment of main roof and L_c cannot be neglected. In other words, the influence of the boundary condition on the fracture position of main roof cannot be neglected.

(2) Variations in the maximum subsidence ω_0 of the main roof with an elastic foundation coefficient k_1 of coal seam.

The change in the maximum subsidence ω_0 with an elastic foundation coefficient k_1 of the coal seam is presented in Figure 11. The maximum subsidence continuously decreased with an increase in k_1, whereas the sensitivity of the maximum subsidence ω_0 decreased with an increase in k_1. The value of the maximum subsidence was reduced by approximately 10% with an increase in k_1 (varying from 1000 to 8000 MN/m^3).

4.4. Effect of the Thickness h of the Main Roof. The thickness of the main roof can vary from mine to mine, depending on various circumstances (geology, previous mining, etc.). The effects of various thicknesses h were analyzed on the maximum subsidence ω_0 and the principal bending moments in order to encompass a wide range of scenarios.

(1) Change in the principal bending moments and L_c with h

Figure 12 graphically presents the variations of each of the three principal bending moments and L_c with various thicknesses of the main roof h (4–12 m). The principal bending moments (M_c, M_d and M_{zz}) continuously increased with the increase in the roof thickness. Principal bending moment M_{zz} was highly sensitive to h, and comparatively the principal bending moment M_d was less sensitive. The principal bending moments M_c and M_d were almost equal when the thickness was relatively small (4–8 m). In such situation, the long side and short side area of the main roof almost fractured concurrently when the maximum principal bending moment of main roof increased to the ultimate bending moment M_s. L_c increased linearly with the increase in the roof thickness. Actually, the thicker the roof, the further the distance between the fracture line of main roof and the coal wall (refer to Figure 12).

(2) Changes in the maximum subsidence of the main roof with various thicknesses h

Figure 13 shows the change in the maximum subsidence ω_0 of the main roof with various thicknesses

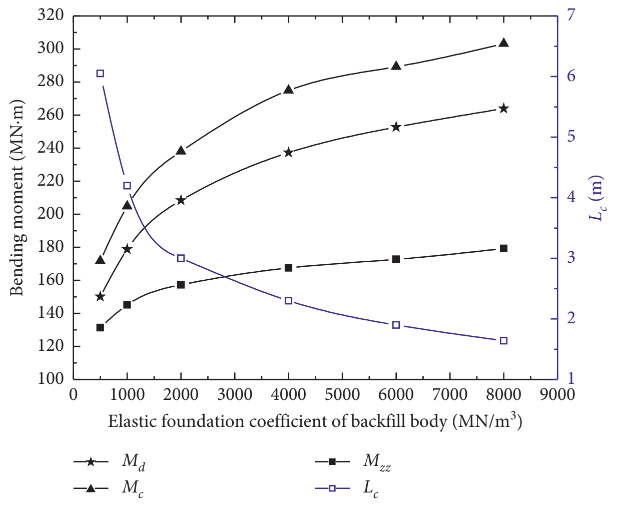


Figure 10: Change in the principal bending moments and L_c with k_1.

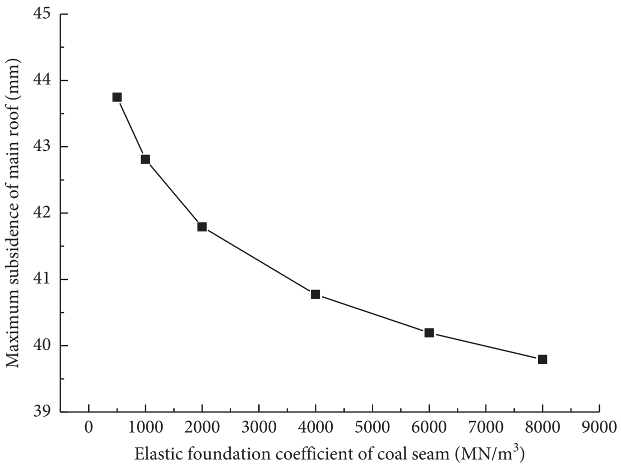


Figure 11: Change in the maximum subsidence ω_0 with k_1.

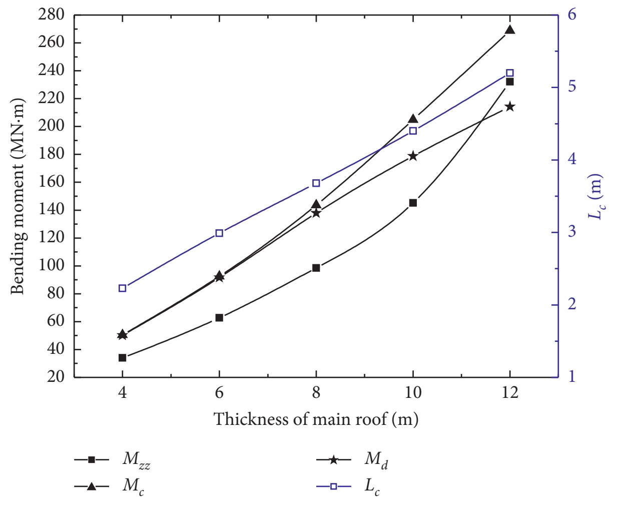


Figure 12: Change in the principal bending moments and L_c with h.

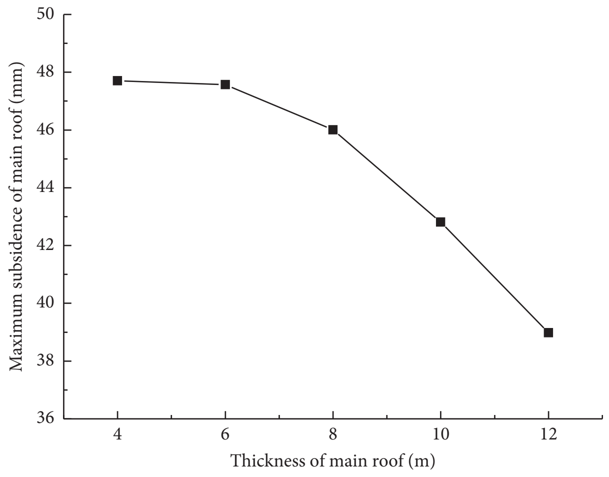


Figure 13: The maximum subsidence ω_0 of main roof with roof thickness h.

(4–12 m). The maximum subsidence of the main roof continuously decreased with an increase in roof thickness, and the sensitivity of the maximum subsidence increased with an increase in the roof thickness.

4.5. Variations of Young's Modulus E of the Main Roof in Backfill Mining. Young's modulus E of the main roof was then varied to analyze the effects on the principal bending moment and maximum subsidence ω_0.

(1) Changes in the principal bending moments and L_c with various E

Figure 14 shows the changes in the principal bending moments and L_c with various Young's modulus values E (10–75 GPa). The principal bending moments M_c and M_d were almost equal when E was relatively small (10 GPa). In this condition, the long side and short side area of the main roof almost fractured concurrently when the maximum principal bending moment of the main roof increased to M_s. When the value of Young's modulus reached 70 GPa, M_c and M_{zz} were nearly equal. In such situation, the long side and the middle area of the main roof almost fractured simultaneously when the principal bending moment was greater than M_s. The larger the E, the farther the L_c, and the farther the distance between the fracture line of main roof and the coal wall (refer to Figure 14). Based on the effect of Young's modulus on the main bending moment, we can conclude that the greater Young's modulus E value and the tensile strength must also be large within the main roof to ensure the main roof does not fracture.

(2) Change in maximum subsidence ω_0 with various Young's modulus E values of the main roof

Figure 15 shows the change in the maximum subsidence ω_0 with an increase in Young's modulus E of the main roof. Maximum subsidence of the main roof continuously decreased with an increase in E, and the sensitivity of ω_0 decreased with an increase in E.

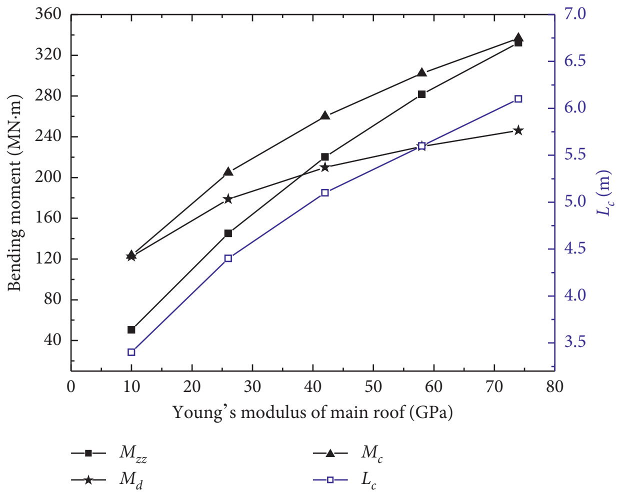


Figure 14: Change in the principal bending moments and L_c with E.

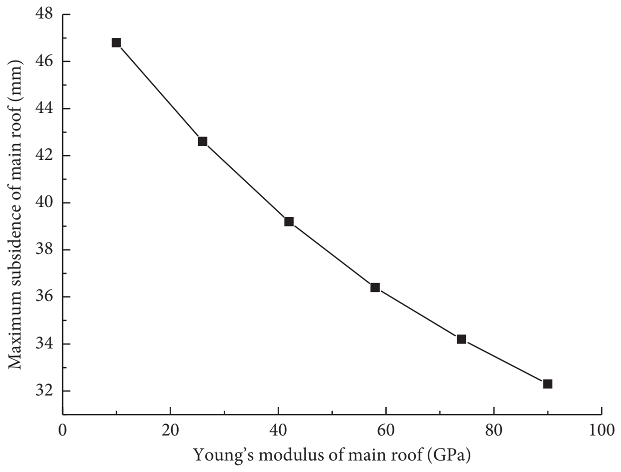


Figure 15: The change in the maximum subsidence ω_0 with various E.

4.6. Effects of Poisson's Ratio μ on the Main Roof in Backfill Mining. Variations of the principal moments and maximum subsidence ω_0 with Poisson's ratio μ of main roof are discussed in the following calculations. Generally, the value of Poisson's ratio varies from 0.1 to 0.3.

(1) Changes in the principal moments and L_c with various Poisson's ratio μ values

Figure 16 shows variations in principal moments and L_c with changes in Poisson's ratio μ (0.10–0.30). The principal moments (M_c, M_d and M_{zz}) gradually increased with the increase in μ, and L_c remained almost unchanged. Poisson's ratio μ had little effect on principal moments and L_c; therefore, Poisson's ratio

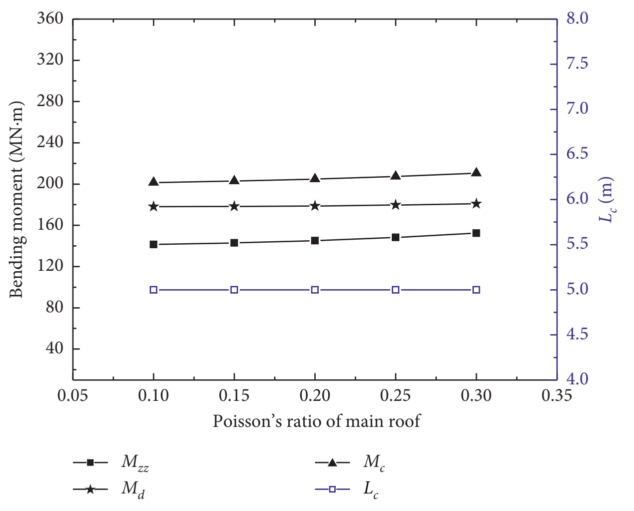


Figure 16: The variation of principal bending moments and L_c with μ.

would have little effect on the position of fracturing in the main roof.

(2) Variations of maximum subsidence ω_0 of the main roof with μ

Figure 17 shows how the maximum subsidence ω_0 of the main roof varies with μ (0.10–0.30). The maximum subsidence slowly decreased with an increase in Poisson's ratio. The maximum subsidence values ω_0 when $\mu = 0.1$ and $\mu = 0.3$ were almost equal. Overall, Poisson's ratio has little effect on the maximum subsidence of the main roof.

4.7. Effects of k_1, k_2, E in Backfill Mining. As observed through the above calculations, both the distribution of the principal bending moments of the main roof and the distance of the advancing coal seam L_c were influenced by k_1, k_2 and E. The value of the principal bending moments decreased as k_2 increased while the principal bending moments increased as k_1 increased; L_c decreased as k_1 increased while L_c increased as k_2 increased; the principal bending moments and L_c increased as E increased. These calculations only analyzed the variations of the principal bending moments with a single parameter. To better characterize backfill mining, multiple parameters were analyzed to determine the effects of the principal bending moments of the main roof and L_c.

(1) Variations of the principal moments and L_c with k_1, k_2, E

Figure 18 shows the variations of the principal bending moments and L_c with k_1, k_2, and E all taken into consideration. The bending moments of the main roof and L_c did not change when k_1, k_2 and E changed but the ratio of the three ($k_1/k_2/E$) remained constant, and the fracture position was unchanged.

(2) Changes in the maximum subsidence ω_0 with various k_1, k_2, E

Figure 19 shows the changes in the maximum subsidence ω_0 with various k_1, k_2 and E. The

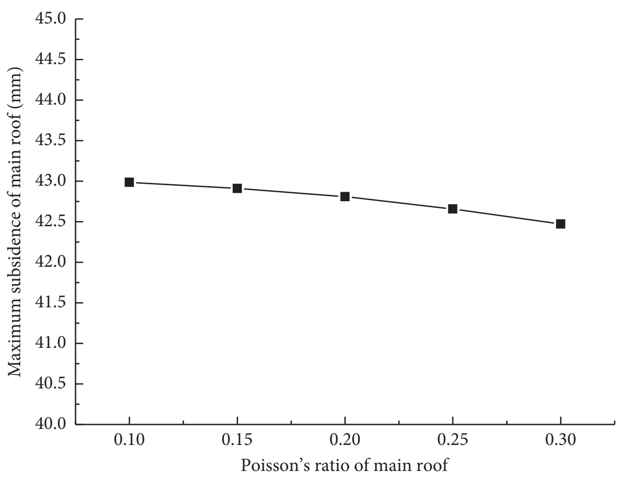


Figure 17: Variations of the maximum subsidence ω_0 with μ.

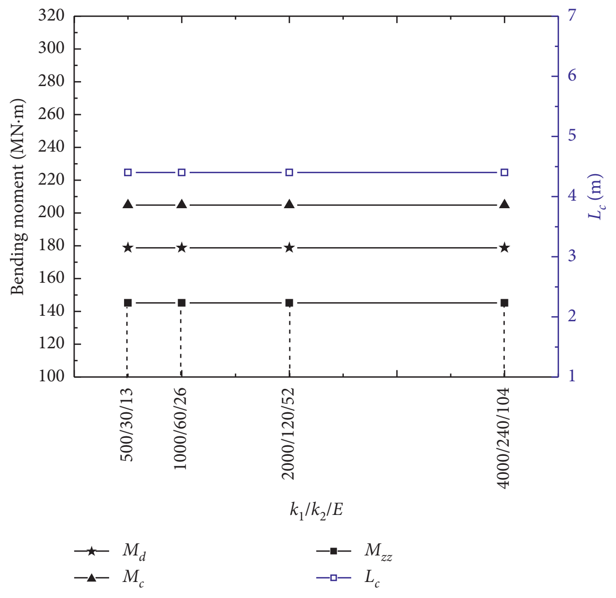


Figure 18: Variation of the principal bending moments with $k_1/k_2/E$.

maximum subsidence ω_0 of the main roof decreased gradually and the reduction extent significantly decreased when k_1, k_2 and E increased but with a constant ratio ($k_1/k_2/E$). The principal moments and fracture position of the main roof remained constant with the constant ratio ($k_1/k_2/E$), although the maximum subsidence ω_0 changed significantly.

4.8. Comparison of Model Conclusions. Under the condition of backfill mining, the mechanical model of thin plate was the most suitable to study the actual characteristics of overburden movement, while the rock beam model could only study the local mechanical characteristics in the mining area. Therefore, the model in this paper mainly studied the mechanical model of plate structure under the condition of

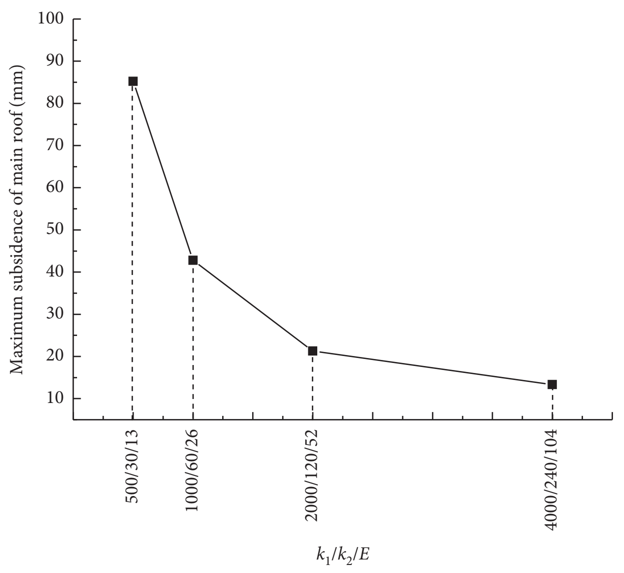


FIGURE 19: Variation of the maximum subsidence ω_0 with various *k_1, k_2 and E.*

backfill rather than the mechanical model of rock beam with defects.

In general, the thin plate mechanical model with fixed support boundary around the overburden in the backfill area was widely used [31], so the comparison object selected in this paper was the thin plate mechanical model with fixed support boundary of four sides.

The main differences were as follows:

(1) The elastic foundation coefficient k_2 of backfill body, the elastic foundation coefficient k_1 of the coal seam, the thickness h, and Young's modulus E of main roof could affect the values and positions of the main roof principal moments, the distance of principal moments advancing coal wall. However, the traditional thin plate model with fixed support boundary could not get these useful conclusions.

(2) In this paper, it was found that the fracture location of the main roof was deep into the coal body area, while the traditional model showed that the first fracture location of the main roof was on the long side of the mining area and along the coal wall, which was independent of the boundary conditions, backfill body parameters, thickness, and Young's modulus of the main roof.

(3) In particular, the weight relationship of the influencing factors of main roof fracture law was obtained; that is, when the ratio ($k_1/k_2/E$) was constant, the magnitude of the principal bending moments was constant, but the maximum subsidence changed significantly, which could not be obtained by the traditional plate structure model with fixed boundary.

It could be seen that the conclusion of the traditional model had great limitations. The model and the conclusions of this paper were more practical and helpful to guide the engineering practice, which effectively made up for the shortcomings of the traditional model.

5. Engineering Practice

Changes in the principal moments of the main roof, the distance of the M_c advancing coal wall L_c, and the maximum subsidence ω_0 with k_1, k_2, E, h, μ and the ratio of $k_1/k_2/E$ in backfill mining were examined to assist with forming the basis for engineering practice. To verify the theoretical calculations, the value of the elastic foundation coefficient k_2 in the I26 backfill working face of Xingdong mine was determined. Subsidence of main roof and support pressure were then measured during mining.

(1) Calculating the elastic foundation coefficient of the backfill body

The tensile strength of the main roof was measured at 12.5 MPa and the ultimate bending moment of the main roof was 208 MN m in the I26 backfill working face. When the elastic foundation coefficient of the backfill body was $k_2 = 60\,\text{MN/m}^3$, the maximum principal moment of main roof was $M_c = 199\,\text{MN/m}^3$, and the distance of the M_c advancing coal wall was $L_c = 4\,\text{m}$ (refer to Figure 8).

To control the ground deformation and protect the main roof from fracturing, the value of k_2 of the backfill body should be about 56–60 MN/m^3, while, using previous methods [31], the thin plate mechanical model with fixed support boundary, to calculate k_2, the k_2 would be 290 MN/m^3. Clearly, the result calculated by the previous method was much larger than that calculated by the present method. From Figure 9 we know that the maximum subsidence of main roof given by theoretical calculation should be 44.5 mm when the value of k_2 is 60 MN/m^3.

The I26 working face adopted the backfill bag of high-water material to backfill mining. In order to meet the production need of the working face and simultaneously ensure the convenience of backfilling the goaf, the size of the backfill bag was designed to be $15 \times 2.2 \times 4.8\,\text{m}$ (length × width × height). The high-water material used in the I26 working face was a quick-setting high-water material. The material was prepared from the mixture of materials A and B, which solidify after mixing for 10–20 minutes. The elastic foundation coefficient k_2 of the prepared high-water material was about 56–60 MN/m^3.

(2) Field observation

The final main roof subsidence was about 48 mm, which was consistent with the value of theoretical calculations presented here.

Mine pressure monitoring results showed that there was no periodic weighting during the mining of the I26 working face, which indicates that the main roof did not periodically fracture, and the borehole observation also indicated that the main roof did not fracture. Overall, it is feasible to determine the elastic foundation coefficient k_2 of the backfilling material using the proposed theoretical calculation. The result calculated by the previous method was much larger than that calculated by the present method. Trying to achieve the larger elastic foundation coefficient of the backfilling material is much more costly and excessive for the given situation. Thus, the thin plate model on the elastic foundation with an elastic foundation boundary based on the actual boundary of the surrounding rock can better guide practice and can bring better economic benefits.

6. Conclusions

It is very important to establish a mechanical model that is in line with the actual characteristics of the project to effectively analyze and solve the mining engineering problems.

In this paper, the thin plate model with elastic foundation boundary of overlying strata for backfill mining was established, and the calculation method was given. The results showed that the calculation method was feasible. Through this model, the fracturing law of the main roof was studied in detail, and new conclusions that the traditional model could not get were obtained, which made up the defects of the traditional model.

(1) The elastic foundation coefficient k_1 of the coal seam, the elastic foundation coefficient k_2 of backfill body, the thickness h, Young's modulus E, and Poisson's ratio μ of main roof could affect the values and positions of the main roof principal moments, the distance of principal moments advancing coal wall, and the maximum subsidence ω_0; the influence of k_2 on the principal moments and maximum subsidence ω_0 of the main roof was the largest, while the influence of μ was the smallest; the values and positions of each of the principal moments remained unchanged but ω_0 decreased significantly when k_1, k_2 and E increased and maintained the constant ratio ($k_1/k_2/E$).

(2) According to the variations in the principal moment, there were three types of initial fracturing of the main roof: fracturing in the long side ahead of the coal wall, fracturing in the long side ahead of the coal wall and the middle area of the main roof simultaneously, and fracturing in the long side ahead of the coal wall and short side ahead of the coal wall simultaneously, which was helpful to effectively monitor the fracturing position of the main roof in engineering practice.

(3) The elastic foundation coefficient k_2 of the backfill body calculated by the thin plate model on the elastic foundation with elastic foundation boundary was far less than the value of k_2 calculated by the thin plate model on an elastic foundation with fixed boundary. Using the I26 working face as verification of the calculation method, when the value of k_2 was 56–60 MN/m^3, the main roof did not undergo periodic weighting during mining. It provided a new and more reliable calculation method for determining the elastic foundation coefficient of the backfill body in order to protect the main roof from fracturing.

(4) The calculation model in this paper effectively made up for the defects and deficiencies of the traditional mechanical model, especially in the fracturing law of the main roof, fracturing conditions, and the weight relationship of the influencing factors. The conclusions promote the theoretical progress and had important reference significance for engineering practice.

Data Availability

All data contained in this study are available upon request from the corresponding author.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This research was funded by the China Postdoctoral Science Foundation (Grant no. 2019M650895), the National Natural Science Foundation of China (Grant no. 51974317), the National Natural Science Foundation of China (Grant no. 51504259), the Fundamental Research Funds for the Central Universities (Grant no. 2010QZ06), and the Yue Qi Young Scholar Project, China University of Mining and Technology, Beijing (Grant no. 800015Z1104).

References

[1] M. G. Qian, J. L. Xu, and X. X. Miao, "Green technique in coal mining," *Journal of China University of Mining & Technology*, vol. 32, no. 4, pp. 343–348, 2003.

[2] Q. Y. Wu, J. W. Pang, S. Z. Qi et al., "Impacts of coal mining subsidence on the surface landscape in Longkou city," *Environmental Earth Sciences*, vol. 59, no. 4, pp. 783–791, 2009.

[3] Y.-B. Jung, W.-K. Song, D.-S. Cheon, D.-K. Lee, and J.-Y. Park, "Simple method for the identification of subsidence susceptibility above underground coal mines in Korea," *Engineering Geology*, vol. 178, pp. 121–131, 2014.

[4] T. Unlu, H. Akcin, and O. Yilmaz, "An integrated approach for the prediction of subsidence for coal mining basins," *Engineering Geology*, vol. 166, pp. 186–203, 2013.

[5] A. H. M. Ng, L. L. Ge, Y. G. Yan et al., "Mapping accumulated mine subsidence using small stack of SAR differential interferograms in the Southern coalfield of New South Wales, Australia," *Engineering Geology*, vol. 115, no. 1-2, pp. 1–15, 2010.

[6] M. G. Karfakis and E. Topuz, "Post mining subsidence abatements in Wyoming abandoned coal mines," *Mining Science & Technology*, vol. 12, no. 3, pp. 233–240, 1991.

[7] J. Barbato, B. Hebblewhite, R. Mitra, and K. Mills, "Prediction of horizontal movement and strain at the surface due to longwall coal mining," *International Journal of Rock Mechanics and Mining Sciences*, vol. 84, pp. 105–118, 2016.

[8] L. Li, K. Wu, and D.-W. Zhou, "AutoCAD-based prediction of 3D dynamic ground movement for underground coal mining," *International Journal of Rock Mechanics and Mining Sciences*, vol. 71, pp. 194–203, 2014.

[9] K.-S. Woo, E. Eberhardt, D. Elmo, and D. Stead, "Empirical investigation and characterization of surface subsidence related to block cave mining," *International Journal of Rock Mechanics and Mining Sciences*, vol. 61, pp. 31–42, 2013.

[10] J. Ju and J. Xu, "Surface stepped subsidence related to top-coal caving longwall mining of extremely thick coal seam under shallow cover," *International Journal of Rock Mechanics and Mining Sciences*, vol. 78, pp. 27–35, 2015.

[11] B. Yu, J. Zhao, T. Kuang, and X. Meng, "In situ investigations into overburden failures of a super-thick coal seam for longwall top coal caving," *International Journal of Rock Mechanics and Mining Sciences*, vol. 78, pp. 155–162, 2015.

[12] H.-J. Oh and S. Lee, "Integration of ground subsidence hazard maps of abandoned coal mines in Samcheok, Korea," *International Journal of Coal Geology*, vol. 86, no. 1, pp. 58–72, 2011.

[13] E. Can, Ş. Kuşcu, and C. Mekik, "Determination of underground mining induced displacements using GPS observations in Zonguldak-Kozlu Hard Coal Basin," *International Journal of Coal Geology*, vol. 89, no. 1, pp. 62–69, 2012.

[14] Ş. Düzgün, C. Künzer, and C. Ö. Karacan, "Applications of remote sensing and GIS for monitoring of coal fires, mine subsidence, environmental impacts of coal-mine closure and reclamation," *International Journal of Coal Geology*, vol. 86, no. 1, pp. 1-2, 2011.

[15] M. Li, J. Zhang, N. Zhou, and Y. Huang, "Effect of particle size on the energy evolution of crushed waste rock in coal mines," *Rock Mechanics and Rock Engineering*, vol. 50, no. 5, pp. 1347–1354, 2017.

[16] Q. Zhang, J.-x. Zhang, Y. Tai, K. Fang, and W. Yin, "Horizontal roof gap of backfill hydraulic support," *Journal of Central South University*, vol. 22, no. 9, pp. 3544–3555, 2015.

[17] Q. Zhang, J. Zhang, Y. Huang, and F. Ju, "Backfilling technology and strata behaviors in fully mechanized coal mining working face," *International Journal of Mining Science and Technology*, vol. 22, no. 2, pp. 151–157, 2012.

[18] Q. Zhang, J. X. Zhang, T. Kang, Q. Sun, and W. K. Li, "Mining pressure monitoring and analysis in fully mechanized backfilling coal mining face-a case study in Zhai Zhen coal mine," *Journal of Central South University*, vol. 22, no. 5, pp. 1965–1972, 2015.

[19] J. Zhang, B. Li, N. Zhou, and Q. Zhang, "Application of solid backfilling to reduce hard-roof caving and longwall coal face burst potential," *International Journal of Rock Mechanics and Mining Sciences*, vol. 88, pp. 197–205, 2016.

[20] J. Li, J. Zhang, Y. Huang, Q. Zhang, and J. Xu, "An investigation of surface deformation after fully mechanized, solid back fill mining," *International Journal of Mining Science and Technology*, vol. 22, no. 4, pp. 453–457, 2012.

[21] M. Benzaazoua, M. Fall, and T. Belem, "A contribution to understanding the hardening process of cemented pastefill," *Minerals Engineering*, vol. 17, no. 2, pp. 141–152, 2004.

[22] Z. D. Cui and H. H. Sun, "The preparation and properties of coal gangue based similar paste like backfill material," *Journal of China Coal Society*, vol. 35, no. 6, pp. 896–899, 2010.

[23] X. Qin, P. Wang, L. Liu, M. Wang, and J. Xin, "Sensitivity analysis of microstructure parameters and mechanical strength during consolidation of cemented paste backfill," *Mathematical Problems in Engineering*, vol. 2018, Article ID 5170721, 9 pages, 2018.

[24] H. Q. Zhou, C. J. Hou, X. K. Sun et al., "Solid waste paste filling for none-village-relocation coal mining," *Journal of China University of Mining and Technology*, vol. 35, no. 3, pp. 403–408, 2006.

[25] Y. Ding, G. M. Feng, and C. Z. Wang, "Experimental research on basic properties of super high-water packing material," *Journal of China Coal Society*, vol. 36, no. 7, pp. 1087–1092, 2011.

[26] Z. P. Yan, T. Y. Qi, L. K. Zhang, and C. J. Hou, "Study of ZKD quick setting materials with high water content and technique of pump packing," *Journal of China Coal Society*, vol. 22, no. 3, pp. 48–53, 1997.

[27] G. M. Feng, C. D. Sun, C. Z. Wang et al., "Research on goaf filling methods with super high-water material," *Journal of China Coal Society*, vol. 35, no. 12, pp. 1963–1968, 2010.

[28] J. X. Zhang, M. Li, Y. L. Huang et al., "Interaction between backfilling body and overburden strata in fully mechanized backfilling mining face," *Disaster Advances*, vol. 6, no. S5, pp. 1–7, 2013.

[29] X. X. Miao, Y. L. Huang, F. Ju et al., "Strata movement theory of dense backfill mining," *Journal of China University of Mining and Technology*, vol. 41, no. 6, pp. 863–867, 2012.

[30] J. Chen, J. P. Du, W. S. Zhang, and J. X. Zhang, "An elastic model of overlying strata movement during coal mining with gangue back-filling," *Journal of China University of Mining and Technology*, vol. 41, no. 1, pp. 14–19, 2012.

[31] M. Li, J. X. Zhang, H. Q. Jiang, Y. L. Huang et al., "A thin plate on elastic foundation model of overlying strata for dense solid backfill mining," *Journal of China Coal Society*, vol. 39, no. 12, pp. 2369–2373, 2014.

[32] Y. L. Huang, J. X. Zhang, B. F. An, and Q. Zhang, "Overlying strata movement law in fully mechanized coal mining and backfilling longwall face by similar physical simulation," *Journal of Mining Science*, vol. 47, no. 5, pp. 618–627, 2011.

[33] S. R. Xie, D. D. Chen, Y. D Sun, M. M. Gao, Y. J. Sun, and W. Shi, "Analysis on thin plate model of basic roof at elastic foundation boundary (I): first breaking," *Journal of China Coal Society*, vol. 41, no. 6, pp. 1360–1368, 2016.

[34] D. D. Chen, S. R. Xie, L. H. Fu, M. M. Gao, and H. Z. Song, "First fracture of the thin plate of main roof with three sides elastic foundation boundary and one side coal pillar," *Journal of China Coal Society*, vol. 42, no. 10, pp. 2528–2536, 2017.

[35] Z. Wang, X. Liang, and G. Liu, "An analytical method for evaluating the dynamic response of plates subjected to underwater shock employing mindlin plate theory and laplace transforms," *Mathematical Problems in Engineering*, vol. 2013, pp. 1–11, 2013.

[36] I. Park, T. Kim, and U. Lee, "Frequency domain spectral element model for the vibration analysis of a thin plate with arbitrary boundary conditions," *Mathematical Problems in Engineering*, vol. 2016, pp. 1–20, 2016.

[37] D. D. Chen, L. H. Fu, S. R. Xie, and J. C. Zeng, "Time-space relationship between periodic fracture of plate structure of main roof and rebound in whole region with elastic foundation boundary," *Chinese Journal of Rock Mechanics and Engineering*, vol. 38, no. 6, pp. 1172–1187, 2019.

[38] D. D. Chen, S. R. Xie, L. H. Fu, J. C. Zeng, F. X. Xie, and Q. Cheng, "First fracturing of thin plate of main roof with elastic foundation boundary on both sides of the long side of goaf (coal pillars)," *Journal of China Coal Society*, vol. 43, no. 12, pp. 3273–3285, 2018.

[39] A. Kutlu, G. Meschke, and M. H. Omurtag, "A new mixed finite-element approach for the elastoplastic analysis of mindlin plates," *Journal of Engineering Mathematics*, vol. 99, no. 1, pp. 137–155, 2016.

Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 5712461, 16 pages
https://doi.org/10.1155/2020/5712461

Research Article

Combining Imbalance Learning Strategy and Multiclassifier Estimator for Bug Report Classification

Shikai Guo,[1,2] **Siwen Wang**,[2] **Miaomiao Wei**,[2] **Rong Chen**,[1] **Chen Guo**,[2] **and Hui Li**[1]

[1]*The School of Marine Electrical Engineering, Dalian Maritime University, Dalian 116026, China*
[2]*The College of Information Science and Technology, Dalian Maritime University, Dalian 116026, China*

Correspondence should be addressed to Chen Guo; dmuguoc@126.com

Received 23 September 2019; Revised 29 October 2019; Accepted 8 November 2019; Published 17 February 2020

Guest Editor: Weifeng Pan

Since a large number of bug reports are submitted to the bug repository every day, efficiently assigning bug reports to the correct developer is a considerable challenge. Because of the large differences between the different components of different projects, the current bug classification mainly relies on the components of the bug report to dispatch bug reports to the designated developer or developer community. Unfortunately, the component information of the bug report is filled in by default according to the bug submitter and the result is often incorrect. Thus, an automatic technology that can identify high-impact bug reports can help developers to be aware of them early, rectify them quickly, and minimize the damages they cause. In this paper, we propose a method based on the combination of imbalanced learning strategies such as random undersampling (RUS), random oversampling (ROS), synthetic minority oversampling technique (SMOTE), and AdaCost algorithms with multiclass classification methods, OVO and OVA, to solve bug reports component classification problem. We investigate the effectiveness of different combinations, i.e., variants, each of which includes a specific imbalance learning strategy and a specific classification algorithm. We mainly perform an analytical study on five open bug repositories (Eclipse, Mozilla, GCC, OpenOffice, and NetBeans). The results show that different variants have different performance for bug reports component identification and the best performance variants are combined with the imbalanced learning strategy RUS and the OVA method based on the SVM classifier.

1. Introduction

There are many studies on bug report predictions that have been performed to help reduce software quality issues [1–5]. Software quality requires a great deal of effort in the testing and debugging process. However, in many cases, the developer's resources and time are limited, so many bugs accumulate and are not fixed in the bug repository.

Anvik et al. reported their personal communication with a Mozilla triager who explains, "everyday, almost 300 bugs appear that need triaging, which is far too much for only the Mozilla programmers to handle" [6]. Therefore, it is especially important to find an effective method for improving the efficiency of bug allocation and resolution. Many developer-recommended methods have been proposed to solve bug classification problems by recommending suitable developers for bug reports to improve the efficiency of bug fixes. Xie et al. proposed a developer-recommended method based on a topic model, using historical bug-solving records to build topic models, simulating developers' interest and expertise in bug-solving activities, and providing a helpful developer recommendation list for new bug reports [7]. On this basis, Xia et al. proposed a bug-based analysis between reports and a developer-based analysis for recommending a suitable developer list for new bug reports by calculating the relevance score [8]. Many researchers have proposed bug prediction techniques to prioritize software testing and debugging that can identify flawed components for developers and conduct considerable research on defect prediction. These techniques predict the allocation model based on features such as code lines, code complexity, and the number of modified files [9–11]. Yang et al. proposed using deep learning techniques to predict changes in bug reports, extracting a set of expression features from initial variation features through deep confidence network algorithms, and constructing machine learning classifiers based on these expression features [12].

However, bug classification still has many problems and faces many challenges. Large-scale and low-quality bug data in bug repositories can prevent the usage of automatic bug classification techniques. Since software bug data are free-form text data, it is necessary to generate well-processed bug data to facilitate the application [13–15]. Tamrawi et al. proposed a caching model based on fuzzy sets and knowledge based on the professional repair of developers. The fuzzy set is defined as the relevant technical terms related to the bug report activities that developers have previously participated in. Developers are ranked by calculating the score between technical terms and bug reports [16]. Alenezi et al. proposed an efficient bug classification method based on the term selection method and a naive Bayesian classifier to construct a prediction model. This method improves the efficiency of bug classification by reducing the dimensions of the terms [17]. Some researchers have proposed automatic bug-dispatching techniques, such as support-vector machine (SVM) [18], *k*-nearest neighbor algorithm (KNN) [19], and naive Bayes multinomial (NBM) [20], to ensure that bug reports are assigned to the appropriate developers to improve the accuracy of bug allocation. Xia et al. used different combinations of imbalanced learning strategies and text classification algorithms to identify high-impact bug reports. The problem of class imbalance in bug reports is solved through imbalancing the processing of data.

Because of the large differences in component categories in different open source projects in the bug repository, it is clear from the analysis that managers rely on the component categories of bug reports to dispatch bug reports to a specific developer or developer group. The multiclass classification method OVO does not require retraining all classifiers when adding samples and only needs to retrain the classifiers associated with the added samples. The multiclass classification method OVA only needs to train the same number of classifiers, and the training time is relatively fast. In this paper, we propose an automatic bug reports component classification method that combines the imbalanced learning strategies such as random undersampling (RUS), random oversampling (ROS), synthetic minority oversampling technique (SMOTE), and AdaCost algorithms with the multiclass classification methods, OVO and OVA, to solve the automatic bug reports component classification problem. According to the mechanism of bug report classification, each bug report component is assigned to a specific developer or several developers. We recommend the appropriate developers to implement bug report classification through bug component classification. Since different classification algorithms have different sensitivities to different categories, we explore the effectiveness of different variants to find the optimal variant [21], i.e., each variant contains an imbalanced learning strategy (ILS) and a classification method. To verify the validity of our proposed model, we conduct experiments on five open source datasets: Mozilla, GCC, NetBeans, OpenOffice, and Eclipse.

The contributions of this article are as follows:

(i) We propose a new model that combines imbalanced learning strategies and multiclass classification methods to implement bug reports component classification

(ii) We improve bug reports component classification performance through different combinations of imbalanced learning strategies and classification algorithms, i.e., variants.

(iii) The validity of our proposed model is verified by experiments for the Mozilla, GCC, NetBeans, OpenOffice, and Eclipse projects, and the accuracy, precision, recall, and F1 are used as the evaluation metrics.

The remainder of this paper is organized as follows. The background knowledge and motivations are discussed in Section 2. The design of our approach is discussed in Section 3. The experimental design and results are presented in Section 4, and the conclusions are discussed in Section 5.

2. Background Knowledge and Motivations

2.1. Background Knowledge. Since the number of daily bugs is large and properly assigned and the human triager has difficulty grasping all the knowledge about bugs [22–25], it is time consuming and error prone for humans to manually classify bugs. Existing work uses text-based classification methods to assist in bug classification, for example, [26–29]. Existing work uses a text-based classification approach to assist in preventing misclassification in recommending the correct developer. In such an approach, the summary and description of the bug report are extracted as textual content and the developer who can fix the bug is marked as a label for classification. Then, the appropriate developer is predicted for the new bug report. Since the number of bug reports submitted to the bug repository is very large, during the bug classification process, developers resolve as many bug reports with a high degree of impact and severity as possible. Severity has become a key factor in determining the priority for bug fixes. A number of prediction methods for bug reporting severity labels have been proposed.

Xuan et al. used the modified REP algorithm and *K*-nearest neighbor algorithm to predict the severity of bug reports and fixer recommendations [30]. This method uses a topic model to find the topic to which each bug belongs, introduces topics to enhance similarity function REP, and uses a *K*-nearest neighbor algorithm to search historical bug reports similar to the new bug report. Based on features extracted from the nearest neighbor of the new bug severity prediction, fixer recommendations are realized. Zhang et al. proposed a new automated method, SEVERIS, which helps test engineers assign severity levels to bug reports [31]. Menzies and Marcus compared the text classification algorithms such as naive Bayes, naive Bayes multinomial, *K*-nearest neighbor, and support-vector machine to determine which particular algorithm is most suitable for bug reporting severity level prediction. The results show that the naive Bayes multinomial algorithm has the highest classification accuracy [32]. Lamkanfi et al. proposed a new method of utilizing information retrieval by analyzing the severity label assigned by historical bug reports. Based on the document similarity function of BM25, the severity label is predicted for the new bug report [33]. Bug reports have serious imbalances. Tian et al. proposed a new sampling technique, CR-SMOTE, to enhance the classification of bug reports with real imbalanced severity distributions [34]. This method uses the RSMOTE sampling method combined with the ELM algorithm to achieve bug severity prediction. In subsequent research work, Chen proposed a fuzzy integral fusion multi-RSMOTE method to solve the problem of data distribution imbalance for the randomness problem of RSMOTE [35, 36]. In order to address the imbalance of the dataset, Guo et al. proposed an enhanced oversampling approach called CR-SMOTE to enhance the classification of bug reports with a realistically imbalanced severity distribution [37]. Pan et al. proposed an approach to empirically investigate the static and evolving topological properties enclosed in the weighted software networks by using weighted k-core decomposition [1]. In this study, Pan et al. explored the structural properties of the multilayer software network at the class level by progressively merging layers together, where each coupling type such as inheritance, implements, and method call defines a specific layer [38]. Jiang et al. proposed the ROSF method combining both information retrieval and supervised learning and recommend top-k code snippets for a given free-form query based on two stages [39, 40]. On this basis, Pan et al. proposed a novel approach to identify key class candidates in object-oriented software [41–43]. Chai et al. proposed an approach to cluster mashup services and determine the cluster number based on a genetic algorithm [44, 45]. To reduce the time developers spent analyzing bug reports, Jiang et al. used crowdsourced data to infer and summarized the valid attributes of bug reports [14]. To improve the quality of detection bug reports, Chen et al. proposed a new framework called the test report augmentation framework (TRAF) to help developers better understand and fix bug reports [14, 46–48].

2.2. Motivations. We find that actual data always contain noise and redundancy [49–51]. Noise data can mislead data analysis techniques, while redundant data can increase the cost of data processing [52]. In the bug repository, all bug reports are filled by developers in natural language. As the scale grows, low-quality bugs are accumulated in the bug repository. Such large-scale and low-quality bug data may undermine the effectiveness of bug fixes [53, 54]. Figure 1 shows the distribution of components in the top 10,000 bug reports for the five datasets. Because of the large number of component categories, it is difficult to achieve accurate bug report classification based on current classification methods. Table 1 shows that each developer performs an average of 2-3 components with a single component allocation. In Table 1, the second column represents an average of how each

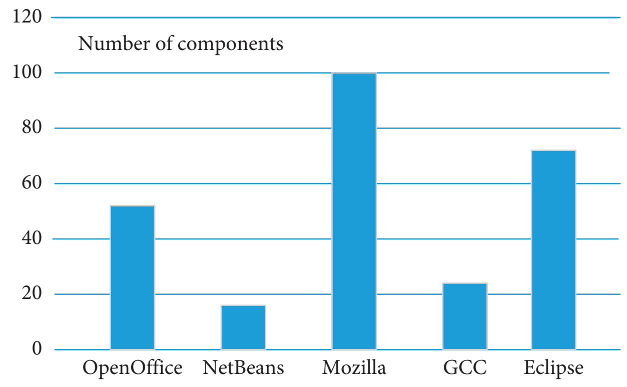


Figure 1: The number of different components in the top 10,000 bug reports for the OpenOffice, NetBeans, Mozilla, GCC, and Eclipse projects.

Table 1: Average of how each component is handled by each developer for the OpenOffice, NetBeans, Mozilla, GCC, and Eclipse projects.

Date Set	Developer/ per_component	Component/ per_developer
OpenOffice	19.42	3.29
NetBeans	19.56	2.84
Mozilla	16.59	2.52
GCC	29.39	2.86
Eclipse	6.79	1.24

component can be solved by several developers. The third column represents how many components each developer can solve on average. Therefore, the rational allocation of components directly affects the classification efficiency of bug reports. The current component information is filled in by the bug reporter and is the default option and cannot be used directly. Therefore, focus should be on the allocation of the bug reports component.

Taking the OpenOffice dataset as an example, 27 components in the dataset are assigned to 1 to 5 developers, 3 components are assigned to 6 to 10 developers, 7 components are assigned to 11 to 20 developers, 6 components are assigned to 21–30 developers, and 10 components are assigned to more than 30 developers. Each component is assigned to approximately 20 developers. On average, each developer handles three components. Therefore, we can more accurately assign a more appropriate developer to the bug report by identifying the bug report component.

3. Methodology

In this section, we present the overall model for bug component allocation problems and detail the algorithms in the overall framework.

3.1. Overview. Inspired by the motivation in Section 2, we propose a new method by combining an imbalance learning strategy (ILS) with a multiclass classification method for bug component classification problems. The model consists of the following three parts. (1) Data preprocessing: we use a text categorization technique to convert each bug report into a word vector based on the vector space model, which is mentioned in [34, 41]. (2) Imbalanced processing of data: because of the serious category imbalance in the dataset, we use four imbalanced learning strategies, RUS, ROS, SMOTE, and AdaCost, to process the data and obtain a balanced dataset to make the classification results more accurate. (3) Multiclass classification of data: we use the multiclass classification methods, OVO and OVA, for the balanced dataset to classify the bug component and solve the bug component classification problem. Since different classification algorithms are sensitive to different categories, we analyze the effectiveness of different variants on the classification of bug reports components, that is, an imbalanced learning strategy combined with a specific classification algorithm. Figure 2 shows the overall framework of our model.

3.2. Imbalanced Learning Strategy. Currently, the strategy for solving the problem of class imbalance of data is mainly divided into two directions. One direction starts with the data training set and reduces the class imbalance of the dataset by changing the sample distribution of the training set. The other direction starts with the learning algorithm according to the algorithm when solving the problem of class imbalance and modifies the algorithm to improve its efficiency. Many data sampling techniques have been introduced in the literature [30, 41, 55]. In our study, we choose RUS, ROS, and SMOTE methods based on changing datasets,and the AdaCost algorithm based on cost sensitivity.

3.2.1. Random Undersampling Method. The random undersampling (RUS) method directly undersamples most of the samples in the training set, that is, it removes some samples in the majority class so that the number of positive and negative examples is close and then learns.

That is, some samples are randomly selected from the majority class S_{maj} to form a sample set E, and then the sample set E is removed from S_{maj} to obtain a new dataset $S_{\text{new}} = S_{\text{maj}} - E$.

3.2.2. Random Oversampling Method. The random oversampling (ROS) method directly oversamples a small number of samples in the training set, that is, it increases some minority samples so that the number of positive and negative examples is close and then learns.

That is, some samples are randomly selected from the minority class S_{min}, and then the sample set E is generated by copying the selected samples, and they are added to S_{min} and the original dataset is expanded to obtain a new minority class set $S_{\text{new}} = S_{\text{min}} + E$.

3.2.3. Synthetic Minority Oversampling Technique. The basic idea of the SMOTE method is to randomly select a sample $\underset{i}{\hat{x}}$ from its nearest neighbors for each minority class sample x_i ($\underset{i}{\hat{x}}$ is a sample of minority classes) and then randomly select a point on the line between x_i and $\underset{i}{\hat{x}}$ as a newly synthesized sample of minority classes.

The specific method for synthesizing new minority samples by the SMOTE method is as follows:

(i) For each sample x_i in minority classes, the distance from x_i to all the samples in the sample set S_{min} of minority classes is calculated according to the Euclidean distance, and its k-nearest neighbors are obtained.

(ii) A sampling ratio is set according to the sample imbalance ratio to determine the sampling magnification N. For each minority sample x_i, several samples are selected randomly from its k-nearest neighbors, assuming that $\underset{i}{\hat{x}}$ is selected.

(iii) For each randomly selected nearest neighbor $\underset{i}{\hat{x}}$, a new sample is constructed with x_i according to the following formula:

$$x_{\text{new}} = x_i + \text{rand}(0,1) \times \left(\underset{i}{\hat{x}} - x_i\right). \tag{1}$$

3.2.4. AdaCost Algorithm. The AdaCost algorithm learns a classifier by iteration and updates the weight of the sample according to the performance of the current classifier. The weight update strategy greatly increases the weight of the costly misclassification sample, and the weight of the correct classification sample is appropriately reduced so that the weight reduction is relatively small. The overall idea is that the cost of the high sample weight is greatly reduced. The sample weights are updated according to the following formula [56]:

$$D_{t+1}(i) = \frac{D_t(i)\exp\left(-\alpha_t h_t(x_i) y_i \beta_i\right)}{Z_t}. \tag{2}$$

$\beta_+ = -0.5C_i + 0.5$, β_+ indicates the value of β in the case where the sample is correctly classified. $\beta_- = 0.5C_i + 0.5$, β_- indicates the value of β in the case where the sample is misclassified.

3.3. Multiclass Classification Method

3.3.1. OVO Method. Assuming there are m categories, the method creates a binomial classifier for the two categories and obtains $k = m * (\text{m} - 1)/2$ classifiers. When classifying new data, the k classifiers are sequentially used for classification. Each classification is equivalent to one vote, and each of the classification results is equivalent to which class is voted for. After classifying using all k classifiers, it is equivalent to k-th voting and the class with the most votes is selected as the final classification result. The following is a description of the structure of the algorithm.

In line 1, y is initialized to null. Lines 2–4 indicate that a classifier is designed between any two samples and classifiers need to be designed. Lines 5–7 indicate that the classification results are obtained. Lines 8–13 represent the voting strategy, and if the sample belongs to the class, one is added. Line 14

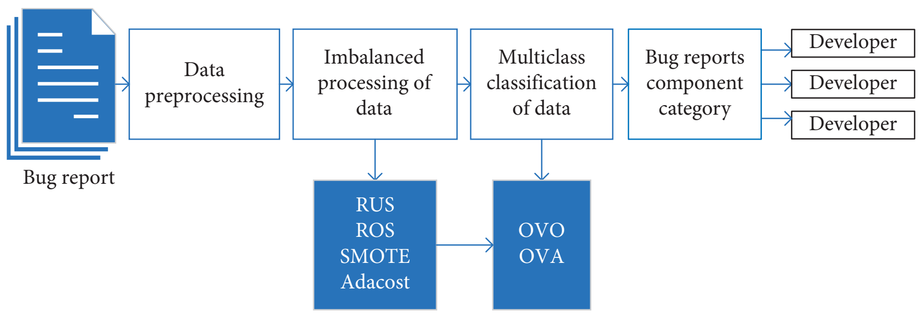

Bug report
Data preprocessing
Imbalanced processing of data
Multiclass classification of data
Bug reports component category
Developer
Developer
Developer
RUS
ROS
SMOTE
Adacost
OVO
OVA

Figure 2: The overall framework of our model.

indicates that the class with the most votes lasting is the class of the unknown sample. Line 17 indicates that the class score with the most votes is returned.

3.3.2. OVA Method. Assuming there are n categories, the method establishes n binomial classifiers; each classifier classifies one of the categories and the remaining categories. When making predictions, we use the n binomial classifiers to classify the data and obtain the probability that the data belong to the current class, and one of the categories with the highest probability is selected as the final prediction result. The following is a description of the structure of the algorithm.

In line 1, y is initialized to null. Lines 2–4 indicate that there are n groups of classifications, i.e., n classifiers. Lines 5–11 indicate that each group of classification results $h_\theta^{(i)}(x)$ is obtained, and the value with the highest probability is selected as the prediction result. Line 12 indicates that the maximum value of the classification value is returned.

4. Experimental Design

4.1. Experimental Datasets. To demonstrate the effectiveness of the proposed approach, we carry out a series of experiments on bug repositories of five large open source projects, namely, Mozilla [57], GCC [58], NetBeans [59], OpenOffice, [60] and Eclipse [61]. By analyzing the components of the bug datasets in these five open source projects, we found that there were category imbalances in all five projects. Table 2 shows the different components and their numbers of bug reports in these five large open source projects. From Table 2, we found that the maximum number of community components reached 10,473, while the number of incubator components was only 19 for the Eclipse project.

Therefore, we need to reduce the datasets; we read the first 1,000 rows of data in five open source projects, and Table 3 shows the word frequency in the five datasets and the total number of bug reports for the original dataset. To more accurately identify the category of bug reports, we removed the number of bug report columns with word frequencies less than 5 and bug reports with a category of less than 50. Table 4 shows the categories of the processed datasets and the number of columns after word frequency reduction.

4.2. Evaluation Metrics. We use accuracy, precision, recall, and F1 as our evaluation metrics. These metrics are commonly used measures for evaluating classification performance [62].

The number of true positives (TP) is the number instances that are correctly divided into positive cases, that is, the number of instances that are actually positive examples and are classified into positive examples by the classifier. The number of false positives (FP) is the number of instances that are incorrectly divided into positive examples, that is, the number of instances that are actually negative but are classified as positive by the classifier. The number of false negatives (FN) is the number of instances that are incorrectly divided into negative examples, that is, the number of instances that are actually positive but are classified as negative by the classifier. The number of true negatives (TN) is the number of instances that are correctly divided into negative examples, that is, the number of instances that are actually negative and are classified as negative by the classifier. Based on the values of TP, FP, FN, and TN, the accuracy, precision, recall, and F1 are calculated as follows.

4.2.1. Accuracy. Accuracy is the number of correctly classified samples divided by the total number of samples. Generally, the higher the correct rate, the better the classifier. We formally define the accuracy as follows:

$$\text{Accurancy} = \frac{TP + TN}{TP + FN + FP + TN}. \tag{3}$$

4.2.2. Precision. Precision is the ratio that is actually divided into positive examples. We formally define the precision as follows:

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{4}$$

4.2.3. Recall. Recall is the proportion of positive cases that are classified as positive examples. Mathematically, recall is defined as

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{5}$$

```
Input:
  $class\ \ y,\ sample\ \ i,\ sample\ \ j,\ learning\ \ algorithm\ \ H\ \ number\ \ of\ \ training\ \ T$
Output:
  class_score
(1)  $y \longleftarrow \varnothing$.
(2)  For $i$ from 1 to $m$ do
(3)    For $j$ from 2 to $m$ do
(4)      Create a binomial classifier with samples of class i and class j, and for all samples, obtain classifiers
(5)      Train the classifier to obtain the classification result.
(6)    $h_\theta^{(i)}(x) = H(y = i \mid x; \theta)$
(7)    $h_\theta^{(j)}(x) = H(y = j \mid x; \theta)$
(8)  Voting strategy:
(9)    if $h_\theta^{(i)}(x) > h_\theta^{(j)}(x)$
(10)     $y_i$ increase one vote.
(11)   Else
(12)     $y_j$ increase one vote.
(13)   End if
(14)       $class\_score = \max_{i=1,\dots,m}\{\sum_{1\le j\ne i\le m, t=1} T_{y_i}, \sum_{1\le j\ne i\le m, t=1} T_{y_i}\}$
(15)     End for
(16)   End for
(17) Return $class\_score$
```

Algorithm 1: OVO method.

```
Input:
$class\ \ y,\ sample\ \ i,\ sample\ \ j,\ learning\ \ algorithm\ \ H$
Output:
H(x)
(1)  $y \longleftarrow \varnothing$.
(2)  For all $y$ from 1 to $n$ do
(3)    Choosing one class and lumping all the others into a single second class, obtain $n$ classifiers.
(4)    Repeat the previous step.
(5)    Train $n$ classifiers $h_\theta^{(i)}(x)$ for each class $i$ to predict the probability that $y = i$.
(6)    $h_\theta^{(1)}(x) = H(y = 1 \mid x; \theta)$
(7)    $h_\theta^{(2,3,\dots,n)}(x) = H(y = 2, 3, \dots, n \mid x; \theta)$
(8)    Record the probability value of the sample belonging to the current class.
(9)    Repeat steps 6, 7, and 8 until all the $n$ classifiers have been trained, and separately record the probability values of each sample
     belonging to the current class.
(10)   Pick the class $i$ that maximizes.
(11) End for
(12) Return $H(x) = \max_i (h_\theta^{(i)}(x))$
```

Algorithm 2: OVA method.

Table 2: Distribution of bug reports components in five open source projects.

Projects	The largest number of component names	Maximum number of components	The smallest number of component names	Minimum number of components	Total number of bug reports (row ∗ column)
Mozilla	Cloud services	4243	Composer	15	(18793 ∗ 35365)
GCC	gcc	13254	Classpath	710	(13964 ∗ 52405)
NetBeans	Java	11593	Groovy	341	(19451 ∗ 40445)
OpenOffice	Impress	2209	Xml	1	(23481 ∗ 50359)
Eclipse	Community	10473	Incubator	19	(40938 ∗ 66029)

Table 3: Word frequency distribution of five datasets and total number of bug reports.

Project	Number of columns with word frequency 1	Number of columns with word frequency 2	Number of columns with word frequency 3	Number of columns with word frequency 4	Number of columns with word frequency below 5	Number of columns with word frequency greater than or equal to 5	The total number of training sets for bug reports
Mozilla	14822	2714	1310	786	19632	4406	(9048 ∗ 24038)
GCC	12438	2627	1287	816	17168	4638	(4851 ∗ 21806)
NetBeans	14121	3507	1616	874	20118	6780	(8538 ∗ 26898)
OpenOffice	14714	2887	1278	832	19711	4591	(9922 ∗ 24302)
Eclipse	15032	3232	1532	853	20649	4854	(9992 ∗ 25503)

Table 4: Category distribution of processed datasets and the number of columns after word frequency reduction.

Project	Maximum number of categories for the bug component	Minimum number of categories for the bug component	Number of columns with word frequency greater than 5
Mozilla	1089	52	4406
GCC	958	86	4638
NetBeans	2170	73	6780
OpenOffice	2770	82	4591
Eclipse	1064	52	4854

Table 5: Experimental results of the OVO and OVA methods based on the NBM, KNN, and SVM classifiers for the Mozilla dataset.

Project	Classifier	Evaluation metrics	OVO method	OVA method
Mozilla	NBM	Accuracy	0.5066	0.5066
		Precision	0.5336	0.5336
		Recall	0.5066	0.5066
		F1	0.5074	0.5074
	KNN	Accuracy	Memory overflow	0.2674
		Precision	Memory overflow	0.3785
		Recall	Memory overflow	0.2674
		F1	Memory overflow	0.2814
	SVM	Accuracy	0.3569	0.5928
		Precision	0.704	0.6015
		Recall	0.3569	0.5928
		F1	0.3626	0.5787

Table 6: Experimental results of the OVO and OVA methods based on the NBM, KNN, and SVM classifiers for the GCC dataset.

Project	Classifier	Evaluation metrics	OVO method	OVA method
GCC	NBM	Accuracy	0.6343	0.6179
		Precision	0.6462	0.6385
		Recall	0.6343	0.6179
		F1	0.6389	0.6207
	KNN	Accuracy	0.4356	0.4356
		Precision	0.461	0.4417
		Recall	0.4356	0.4356
		F1	0.3817	0.4239
	SVM	Accuracy	0.5705	0.6539
		Precision	0.6723	0.6685
		Recall	0.5705	0.6539
		F1	0.4977	0.6153

Table 7: Experimental results of the OVO and OVA methods based on the NBM, KNN, and SVM classifiers for the Eclipse dataset.

Project	Classifier	Evaluation metrics	OVO method	OVA method
Eclipse	NBM	Accuracy	0.7332	0.7044
		Precision	0.7706	0.7532
		Recall	0.7332	0.7044
		F1	0.7429	0.7185
	KNN	Accuracy	0.4947	0.5292
		Precision	0.6949	0.6382
		Recall	0.4947	0.5292
		F1	0.5045	0.5462
	SVM	Accuracy	0.6417	0.7892
		Precision	0.7836	0.8033
		Recall	0.6417	0.7892
		F1	0.616	0.7781

Table 8: Experimental results of the OVO and OVA methods based on the NBM, KNN, and SVM classifiers for the OpenOffice dataset.

Project	Classifier	Evaluation metrics	OVO method	OVA method
OpenOffice	NBM	Accuracy	0.425	0.407
		Precision	0.5173	0.5019
		Recall	0.425	0.407
		F1	0.4438	0.4226
	KNN	Accuracy	0.3598	0.3747
		Precision	0.4967	0.3996
		Recall	0.3598	0.3747
		F1	0.3584	0.37
	SVM	Accuracy	0.4255	0.5205
		Precision	0.6025	0.5321
		Recall	0.4255	0.5205
		F1	0.3497	0.4964

Table 9: Experimental results of the OVO and OVA methods based on the NBM, KNN, and SVM classifiers for the NetBeans dataset.

Project	Classifier	Evaluation metrics	OVO method	OVA method
NetBeans	NBM	Accuracy	0.5024	0.4941
		Precision	0.5341	0.5117
		Recall	0.5024	0.4941
		F1	0.5097	0.4973
	KNN	Accuracy	0.3939	0.354
		Precision	0.4594	0.4077
		Recall	0.3939	0.354
		F1	0.3834	0.3508
	SVM	Accuracy	0.4614	0.616
		Precision	0.6537	0.626
		Recall	0.4614	0.616
		F1	0.4432	0.6051

4.2.4. F-Measure. *F*-measure is also known as *F*-score. When the precision (*P*) and recall (*R*) indicators sometimes appear contradictory, you need to use the indicator to consider both of them. *F*-measure is the weighted harmonic average of precision and recall.

$$F - \text{measure} = \frac{(\alpha^2 + 1) * P * R}{\alpha^2 * (P + R)}. \quad (6)$$

When $a = 1$, it is the most common F1, i.e.,

$$F1 = \frac{2 * P * R}{P + R}. \quad (7)$$

5. Experimental Results

In this section, the experimental results are discussed in relation to the specific research questions.

5.1. RQ1: Which Classification Method Has Better Classification Effect Based on NBM, KNN, and SVM Classifiers' OVO and OVA Methods? To answer this question, we use the OVO and OVA multiclass classification methods based on NBM, KNN, and SVM classifiers, which together contain six variants (i.e., OVO method-based NBM classifier, OVO method-based KNN classifier, OVO method-based SVM classifier, OVA method-based NBM classifier, OVA method-based KNN classifier, and OVA method-based SVM classifier) and record the experimental results. We use accuracy, precision, recall, and F1 as evaluation criteria. Tables 5–9 show our experimental results for the Mozilla, GCC, NetBeans, OpenOffice, and Eclipse datasets.

From Tables 5–9, it can be seen that the OVA multiclass classification method based on the SVM classifier improves the effect most obviously for the five datasets. The experimental results of the OVA method based on the SVM classifier improved by 0.5928, 0.6015, 0.5928, and 0.5787, respectively, for the Mozilla dataset. The experimental results of the OVA method based on the SVM classifier improved by 0.6539, 0.6685, 0.6539, and 0.6153, respectively, for the GCC dataset. The experimental results of the OVA method based on the SVM classifier improved by 0.7892, 0.8033, 0.7892, and 0.7781, respectively, for the Eclipse dataset. The experimental results of the OVA method based on the SVM classifier improved by 0.5205, 0.5321, 0.5205, and 0.4964, respectively, for the OpenOffice dataset. The experimental results of the OVA method based on the SVM classifier improved by 0.6160, 0.6260, 0.6160, and 0.6051, respectively, for the NetBeans dataset. Therefore, for the Mozilla, GCC, NetBeans, OpenOffice, and Eclipse datasets, the OVA method based on the SVM classifier has greater efficiency in solving bug reports component classification problems.

5.2. RQ2: What Is the Impact of Imbalanced Learning Strategies on the Multiclass Classification OVO Method in Solving Bug Reports Component Allocation Problems? Specifically, the question explores whether an imbalanced learning strategy has an impact on the OVO classification method. To answer this question, we use the imbalanced learning strategies RUS, ROS, SMOTE, and AdaCost algorithms to process the Mozilla, GCC, Eclipse, OpenOffice, and NetBeans datasets and then use the multiclass classification method OVO based on SVM, KNN, and NBM classifiers to

Table 10: The results of our classification using the ILS (RUS, ROS, SMOTE, and AdaCost) and OVO method based on SVM, KNN, and NBM classifiers for the Mozilla dataset.

Project	ILS	Classifier	Evaluation metrics	OVO method
Mozilla	RUS	SVM	Accuracy	0.2
			Precision	0.525
			Recall	0.2
			F1	0.2294
		KNN	Accuracy	0.0918
			Precision	0.2724
			Recall	0.0918
			F1	0.1042
		NBM	Accuracy	0.4408
			Precision	0.4611
			Recall	0.4408
			F1	0.4372
	ROS	SVM	Accuracy	0.9288
			Precision	0.9415
			Recall	0.9288
			F1	0.9316
		NBM	Accuracy	0.894
			Precision	0.893
			Recall	0.894
			F1	0.8906
	SMOTE	SVM	Accuracy	0.3541
			Precision	0.5533
			Recall	0.3541
			F1	0.3817
		NBM	Accuracy	0.4745
			Precision	0.4907
			Recall	0.4745
			F1	0.4759
	AdaCost	SVM	Accuracy	0.5392
			Precision	0.6477
			Recall	0.5392
			F1	0.534

Table 11: The results of our classification using the ILS (RUS, ROS, SMOTE, and AdaCost) and OVO method based on SVM, KNN, and NBM classifiers for the GCC dataset.

Project	ILS	Classifier	Evaluation metrics	OVO method
GCC	RUS	SVM	Accuracy	0.2905
			Precision	0.4873
			Recall	0.2905
			F1	0.265
		KNN	Accuracy	0.1623
			Precision	0.3543
			Recall	0.1623
			F1	0.0903
		NBM	Accuracy	0.5213
			Precision	0.5024
			Recall	0.5213
			F1	0.5023
	ROS	SVM	Accuracy	0.9016
			Precision	0.9035
			Recall	0.9016
			F1	0.9009
		NBM	Accuracy	0.8601
			Precision	0.8589
			Recall	0.8601
			F1	0.8589
	SMOTE	SVM	Accuracy	0.5036
			Precision	0.5702
			Recall	0.5036
			F1	0.4864
		NBM	Accuracy	0.6271
			Precision	0.6244
			Recall	0.6271
			F1	0.622
	AdaCost	SVM	Accuracy	0.7167
			Precision	0.7127
			Recall	0.7167
			F1	0.682

solve the problem of bug reports component classification. We combine ILS with OVO method based on SVM, KNN, and NBM classifiers and an imbalanced learning strategy combined with a classification method for the five datasets. We use accuracy, precision, recall, and F1 as evaluation criteria.

In addition, first, when we use RUS combined with the OVO method based on SVM, KNN, and NBM classifiers, it is found that RUS combined with the OVO method based on the KNN classifier is the least effective compared with other combinations. Therefore, we use RUS and SMOTE combined with the OVO method based on SVM and NBM classifiers to carry out experiments for the five datasets. According to RQ1, we learned that the OVO and OVA methods based on the SVM classifier work best before the data are imbalanced. Therefore, we only use the algorithm-based AdaCost to combine the OVO method based on the SVM classifier to experiment and observe the experimental results. Tables 10–14 show the results of our experiments using ILS combined with the OVO method for the Mozilla, GCC, NetBeans, OpenOffice, and Eclipse datasets.

From Tables 10–14, it can be seen that the combination of the imbalanced leaning strategy RUS and the OVO method based on SVM and NBM classifiers has higher efficiency in solving the bug reports component classification problem compared to the OVO method based on the KNN classifier for the Mozilla, GCC, NetBeans, OpenOffice, and Eclipse datasets. The combination of imbalanced leaning strategy ROS and the OVO method based on SVM and NBM classifiers has the largest improvement in solving the bug reports component classification problem compared with other combinations for the Mozilla, GCC, NetBeans, OpenOffice, and Eclipse datasets. From Table 10, for the Mozilla dataset, the combination of the ROS and OVO method based on the SVM classifier has the greatest improvement, increasing by 0.9288, 0.9415, 0.9288, and 0.9316, respectively. A combination of the ROS and the OVO method based on the NBM classifier also greatly improved compared with other combinations, increasing by 0.8940, 0.8930, 0.8940, and 0.8906, respectively. The combination of the RUS and OVO method based on the KNN classifier had the lowest improvement, 0.0918, 0.2724, 0.0918, and 0.1042, respectively. The combination of the AdaCost and OVO method based on the SVM classifier had higher improvement compared with the combination of the SMOTE and OVO methods based on the SVM and NBM classifiers,

Table 12: The results of our classification using the ILS (RUS, ROS, SMOTE, and AdaCost) and OVO method based on SVM, KNN, and NBM classifiers for the Eclipse dataset.

Project	ILS	Classifier	Evaluation metrics	OVO method
Eclipse	RUS	SVM	Accuracy	0.1923
			Precision	0.572
			Recall	0.1923
			F1	0.2094
		KNN	Accuracy	0.1713
			Precision	0.4548
			Recall	0.1713
			F1	0.1713
		NBM	Accuracy	0.5734
			Precision	0.6175
			Recall	0.5734
			F1	0.5719
	ROS	SVM	Accuracy	0.9288
			Precision	0.9415
			Recall	0.9288
			F1	0.9316
		NBM	Accuracy	0.917
			Precision	0.9181
			Recall	0.917
			F1	0.9154
	SMOTE	SVM	Accuracy	0.5842
			Precision	0.6729
			Recall	0.5842
			F1	0.5919
		NBM	Accuracy	0.7144
			Precision	0.7288
			Recall	0.7144
			F1	0.7147
	AdaCost	SVM	Accuracy	0.6689
			Precision	0.7966
			Recall	0.6689
			F1	0.6652

Table 13: The results of our classification using the ILS (RUS, ROS, SMOTE, and AdaCost) and OVO method based on SVM, KNN, and NBM classifiers for the OpenOffice dataset.

Project	ILS	Classifier	Evaluation metrics	OVO method
OpenOffice	RUS	SVM	Accuracy	0.1538
			Precision	0.3992
			Recall	0.1538
			F1	0.1635
		KNN	Accuracy	0.1052
			Precision	0.1896
			Recall	0.1052
			F1	0.0958
		NBM	Accuracy	0.4736
			Precision	0.5044
			Recall	0.4736
			F1	0.475
	ROS	SVM	Accuracy	0.9204
			Precision	0.9346
			Recall	0.9204
			F1	0.9225
		NBM	Accuracy	0.8489
			Precision	0.8496
			Recall	0.8489
			F1	0.8437
	SMOTE	SVM	Accuracy	0.4075
			Precision	0.4693
			Recall	0.4075
			F1	0.3844
		NBM	Accuracy	0.4455
			Precision	0.469
			Recall	0.4455
			F1	0.4515
	AdaCost	SVM	Accuracy	0.4856
			Precision	0.528
			Recall	0.4856
			F1	0.4439

increasing by 0.5392, 0.6477, 0.5392, and 0.5340, respectively.

From Table 11, for the GCC dataset, the combination of the ROS and OVO method based on the SVM classifier had the largest improvement, increasing by 0.9016, 0.9035, 0.9016, and 0.9009, respectively. A combination of the ROS and OVO method based on the NBM classifier also greatly improved compared with other combinations, increasing by 0.8601, 0.8589, 0.8601, and 0.8589, respectively. The combination of the RUS and OVO method based on the KNN classifier has the lowest improvement, 0.1623, 0.3543, 0.1623, and 0.0903, respectively. The combination of the AdaCost and OVO method based on the SVM classifier had higher improvement compared to the combination of the SMOTE and OVO method, which increased by 0.7167, 0.7127, 0.7167, and 0.6820, respectively. The combination of the SMOTE and OVO method based on the NBM classifier was more efficient than the combination of the SMOTE and OVO method based on the SVM classifier, with elevations of 0.6271, 0.6244, 0.6271, and 0.6220, respectively. From Table 12, for the Eclipse dataset, the combination of the ROS and OVO method based on the SVM classifier had the greatest improvement, increasing by 0.9288, 0.9415, 0.9288, and 0.9316, respectively. A combination of the ROS and OVO method based on the NBM classifier also greatly improved compared with other combinations, increasing by 0.9170, 0.9181, 0.9170, and 0.9154, respectively. The combination of the RUS and OVO method based on the KNN classifier had the lowest improvement, 0.1713, 0.4548, 0.1713, and 0.1713, respectively. The combination of the SMOTE and OVO method based on the NBM classifier had higher improvement compared with the combination of the SMOTE, AdaCost, and OVO method based on the SVM classifier, which increased by 0.7144, 0.7288, 0.7144, and 0.7147, respectively. From Table 13, for the OpenOffice dataset, the combination of the ROS and OVO method based on the SVM classifier had the greatest improvement, increasing by 0.9204, 0.9346, 0.9204, and 0.9225, respectively. A combination of the ROS and OVO method based on the NBM classifier also greatly improved compared with other combinations, increasing by 0.8489, 0.8496, 0.8489, and 0.8437, respectively. The combination of the RUS and OVO method based on the KNN classifier had the lowest improvement, 0.1052, 0.1896, 0.1052, and 0.0958,

Table 14: The results of our classification using the ILS (RUS, ROS, SMOTE, and AdaCost) and OVO method based on SVM, KNN, and NBM classifiers for the NetBeans dataset.

Project	ILS	Classifier	Evaluation metrics	OVO method
NetBeans	RUS	SVM	Accuracy	0.1729
			Precision	0.2464
			Recall	0.1729
			F1	0.1484
		KNN	Accuracy	0.1518
			Precision	0.3168
			Recall	0.1518
			F1	0.1242
		NBM	Accuracy	0.4008
			Precision	0.4098
			Recall	0.4008
			F1	0.3966
	ROS	SVM	Accuracy	0.9043
			Precision	0.9125
			Recall	0.9043
			F1	0.9058
		NBM	Accuracy	0.8244
			Precision	0.8204
			Recall	0.8244
			F1	0.8175
	SMOTE	SVM	Accuracy	0.4432
			Precision	0.54
			Recall	0.4432
			F1	0.4366
		NBM	Accuracy	0.5019
			Precision	0.5089
			Recall	0.5019
			F1	0.5028
	AdaCost	SVM	Accuracy	0.5645
			Precision	0.6295
			Recall	0.5645
			F1	0.5584

Table 15: The results of our classification using the ILS (RUS, ROS, SMOTE, and AdaCost) and OVA method based on SVM, KNN, and NBM classifiers for the Mozilla dataset.

Project	ILS	Classifier	Evaluation metrics	OVA method
Mozilla	RUS	SVM	Accuracy	0.4612
			Precision	0.493
			Recall	0.4612
			F1	0.4503
		KNN	Accuracy	0.1346
			Precision	0.3396
			Recall	0.1346
			F1	0.156
		NBM	Accuracy	0.3693
			Precision	0.3876
			Recall	0.3693
			F1	0.3612
	ROS	SVM	Accuracy	0.946
			Precision	0.9483
			Recall	0.946
			F1	0.945
		NBM	Accuracy	0.7424
			Precision	0.7683
			Recall	0.7424
			F1	0.742
	SMOTE	SVM	Accuracy	0.4834
			Precision	0.5162
			Recall	0.4834
			F1	0.4901
		NBM	Accuracy	0.4237
			Precision	0.4424
			Recall	0.4237
			F1	0.4188
	AdaCost	SVM	Accuracy	0.4701
			Precision	0.58
			Recall	0.4701
			F1	0.4549

respectively. The combination of the AdaCost and OVO method based on the SVM classifier had higher improvement compared with the combination of the SMOTE and OVO method based on the SVM and NBM classifiers, increasing by 0.4856, 0.5280, 0.4856, and 0.4439, respectively. From Table 14, for the NetBeans dataset, the combination of the ROS and OVO method based on the SVM classifier had the greatest improvement, increasing by 0.9043, 0.9125, 0.9043, and 0.9058, respectively. A combination of the ROS and OVO method based on the NBM classifier also greatly improved compared with other combinations, increasing by 0.8244, 0.8204, 0.8244, and 0.8175, respectively. The combination of the RUS and OVO method based on the KNN classifier had the lowest improvement, 0.1518, 0.3168, 0.1518, and 0.1242, respectively. The combination of the AdaCost and OVO method based on the SVM classifier had higher improvement compared with the combination of the SMOTE and OVO method based on the SVM and NBM classifiers, which increased by 0.5645, 0.6295, 0.5645, and 0.5584, respectively.

Therefore, the combination of the imbalanced leaning strategy RUS and the OVO method based on SVM and NBM classifiers has higher efficiency in solving the bug reports component classification problem compared with other combinations for the Mozilla, GCC, NetBeans, OpenOffice, and Eclipse datasets.

5.3. RQ3: How Much Improvement Does the Classification of Bug Reports Component Have in Combination with the Imbalanced Learning Strategies and the OVA Method? To answer this question, we use the imbalanced learning strategies RUSUS, ROS, SMOTE, and AdaCost algorithms to process the Mozilla, GCC, Eclipse, OpenOffice, and NetBeans datasets and then use the multiclass classification method OVA based on SVM, KNN, and NBM classifiers to solve the problem of bug reports component classification. We combine ILS with OVA method based on SVM, KNN, and NBM classifiers and an imbalanced learning strategy with a classification method for the five datasets. Then, we use accuracy, precision, recall, and F1 as evaluation criteria and we build the classifier with reference to the combination of RQ2. Tables 15–19 show the results of our experiments using ILS combined with the OVA method for the Mozilla, GCC, NetBeans, OpenOffice, and Eclipse datasets.

Table 16: The results of our classification using the ILS (RUS, ROS, SMOTE, and AdaCost) and OVA method based on SVM, KNN, and NBM classifiers for the GCC dataset.

Project	ILS	Classifier	Evaluation metrics	OVA method
GCC	RUS	SVM	Accuracy	0.5042
			Precision	0.4818
			Recall	0.5042
			F1	0.4784
		KNN	Accuracy	0.2222
			Precision	0.4332
			Recall	0.2222
			F1	0.1807
		NBM	Accuracy	0.4786
			Precision	0.4687
			Recall	0.4786
			F1	0.4668
	ROS	SVM	Accuracy	0.919
			Precision	0.9198
			Recall	0.919
			F1	0.9185
		NBM	Accuracy	0.826
			Precision	0.832
			Recall	0.826
			F1	0.8235
	SMOTE	SVM	Accuracy	0.5777
			Precision	0.6044
			Recall	0.5777
			F1	0.5556
		NBM	Accuracy	0.6096
			Precision	0.6186
			Recall	0.6096
			F1	0.6114
	AdaCost	SVM	Accuracy	0.7116
			Precision	0.695
			Recall	0.7116
			F1	0.6899

Table 17: The results of our classification using the ILS (RUS, ROS, SMOTE, and AdaCost) and OVA method based on SVM, KNN, and NBM classifiers for the Eclipse dataset.

Project	ILS	Classifier	Evaluation metrics	OVA method
Eclipse	RUS	SVM	Accuracy	0.5909
			Precision	0.6701
			Recall	0.5909
			F1	0.5906
		KNN	Accuracy	0.2762
			Precision	0.6386
			Recall	0.2762
			F1	0.2967
		NBM	Accuracy	0.5419
			Precision	0.6012
			Recall	0.5419
			F1	0.5404
	ROS	SVM	Accuracy	0.946
			Precision	0.9483
			Recall	0.946
			F1	0.945
		NBM	Accuracy	0.8705
			Precision	0.8745
			Recall	0.8705
			F1	0.8694
	SMOTE	SVM	Accuracy	0.6956
			Precision	0.7412
			Recall	0.6956
			F1	0.7
		NBM	Accuracy	0.6966
			Precision	0.7157
			Recall	0.6966
			F1	0.7018
	AdaCost	SVM	Accuracy	0.7651
			Precision	0.8155
			Recall	0.7651
			F1	0.7673

From Table 15, for the Mozilla dataset, the combination of the ROS and OVA method based on the SVM classifier had the greatest improvement, increasing by 0.9460, 0.9483, 0.9460, and 0.9450, respectively. A combination of the ROS and OVA method based on the NBM classifier also greatly improved compared with other combinations, increasing by 0.7424, 0.7683, 0.7424, and 0.7420, respectively. The combination of the RUS and OVA method based on the KNN classifier had the lowest improvement, 0.01346, 0.3396, 0.1346, and 0.1569, respectively. The combination of the SMOTE and OVA method based on the SVM classifier had higher improvement compared with the combination of the SMOTE and OVA method based on the NBM classifier, and the combination of the AdaCost and OVA method based on the SVM classifier increased by 0.4834, 0.5162, 0.4834, and 0.4901, respectively. From Table 16, for the GCC dataset, the combination of the ROS and OVA method based on the SVM classifier had the largest improvement, increasing by 0.9190, 0.9198, 0.9190 and 0.9185, respectively. A combination of the ROS and OVA method based on the NBM classifier also greatly improved compared to other combinations, increasing by 0.8260, 0.8320, 0.8260, and 0.8235, respectively. The combination of the RUS and OVA method based on the KNN classifier has the lowest improvement, 0.2222, 0.4332, 0.2222, and 0.1807, respectively. The combination of the AdaCost and OVA method based on the SVM classifier had higher improvement compared with the combination of the SMOTE and OVA methods based on the SVM and NBM classifiers, which increased by 0.7116, 0.6950, 0.7116, and 0.6899, respectively. The combination of the SMOTE and OVA methods based on the NBM classifier was more efficient than the combination of the SMOTE and OVA method based on the SVM classifier, with elevations of 0.6069, 0.6186, 0.6096, and 0.6114, respectively. From Table 17, for the Eclipse dataset, the combination of the ROS and OVA method based on the SVM classifier had the greatest improvement, increasing by 0.9460, 0.9483, 0.9460, and 0.9450, respectively. A combination of the ROS and OVA method based on the NBM classifier also greatly improved compared with other combinations, increasing by 0.8705, 0.8745, 0.8705, and 0.8694, respectively. The combination of the RUS and OVA method based on the KNN classifier had the lowest accuracy value, recall value, and F1 value of 0.2762, 0.2762, and 0.2967, respectively. The combination of the AdaCost and OVA method based on the SVM classifier had higher improvement compared with the

Table 18: The results of our classification using the ILS (RUS, ROS, SMOTE, and AdaCost) and OVA method based on SVM, KNN, and NBM classifiers for the OpenOffice dataset.

Project	ILS	Classifier	Evaluation metrics	OVA method
OpenOffice	RUS	SVM	Accuracy	0.4291
			Precision	0.4885
			Recall	0.4291
			F1	0.4325
		KNN	Accuracy	0.1497
			Precision	0.3197
			Recall	0.1497
			F1	0.1534
		NBM	Accuracy	0.4291
			Precision	0.438
			Recall	0.4291
			F1	0.4256
	ROS	SVM	Accuracy	0.93
			Precision	0.9335
			Recall	0.93
			F1	0.9265
		NBM	Accuracy	0.7723
			Precision	0.7838
			Recall	0.7723
			F1	0.7647
	SMOTE	SVM	Accuracy	0.4414
			Precision	0.4764
			Recall	0.4414
			F1	0.4305
		NBM	Accuracy	0.4204
			Precision	0.4438
			Recall	0.4204
			F1	0.4222
	AdaCost	SVM	Accuracy	0.4378
			Precision	0.4714
			Recall	0.4378
			F1	0.3749

Table 19: The results of our classification using the ILS (RUS, ROS, SMOTE, and AdaCost) and OVA method based on SVM, KNN, and NBM classifiers for the NetBeans dataset.

Project	ILS	Classifier	Evaluation metrics	OVA method
NetBeans	RUS	SVM	Accuracy	0.4683
			Precision	0.4701
			Recall	0.4683
			F1	0.4586
		KNN	Accuracy	0.1476
			Precision	0.181
			Recall	0.1476
			F1	0.117
		NBM	Accuracy	0.3713
			Precision	0.3804
			Recall	0.3713
			F1	0.3699
	ROS	SVM	Accuracy	0.9197
			Precision	0.9178
			Recall	0.9197
			F1	0.9172
		NBM	Accuracy	0.7381
			Precision	0.7463
			Recall	0.7381
			F1	0.735
	SMOTE	SVM	Accuracy	0.5329
			Precision	0.568
			Recall	0.5329
			F1	0.5239
		NBM	Accuracy	0.4775
			Precision	0.4809
			Recall	0.4775
			F1	0.4723
	AdaCost	SVM	Accuracy	0.5346
			Precision	0.5869
			Recall	0.5346
			F1	0.5217

combination of the SMOTE and OVA method based on the SVM and NBM classifiers, which increased by 0.7651, 0.8155, 0.7651, and 0.7673, respectively. The combination of the SMOTE and OVA method based on the NBM classifier had higher accuracy value, recall rate, and F1 value compared with the combination of the SMOTE and OVA method based on the SVM classifier, with elevations of 0.6966, 0.6966, and 0.7018, respectively. From Table 18, for the OpenOffice dataset, the combination of the ROS and OVA method based on the SVM classifier had the greatest improvement, increasing by 0.9300, 0.9335, 0.9300, and 0.9265, respectively. A combination of the ROS and OVA method based on the NBM classifier also greatly improved compared with other combinations, increasing by 0.7723, 0.7838, 0.7723, and 0.7647, respectively. The combination of the RUS and OVA method based on the KNN classifier had the lowest improvement, 0.1497, 0.3197, 0.1497, and 0.1534, respectively. The combination of the SMOTE and OVA method based on the SVM classifier had higher improvement compared with the combination of the SMOTE and OVA method based on the NBM classifier, and the combination of the AdaCost and OVA methods based on the SVM classifier increased by 0.4414, 0.4764, 0.4414, and 0.4305, respectively. From Table 19, for the NetBeans dataset, the combination of the ROS and OVA method based on the SVM classifier had the greatest improvement, increasing by 0.9197, 0.9178, 0.9197, and 0.9172, respectively. A combination of the ROS and OVA method based on the NBM classifier also greatly improved compared with other combinations, increasing by 0.7381, 0.7463, 0.7381, and 0.7350, respectively. The combination of the RUS and OVA method based on the KNN classifier had the lowest improvement, 0.1476, 0.1810, 0.1476, and 0.1170, respectively. The combination of the AdaCost and OVA method based on the SVM classifier had the highest accuracy value, the precision rate value, and the recall rate value compared with the combination of the SMOTE and OVA method based on the SVM and NBM classifiers which increased by 0.5346, 0.5869, and 0.5346, respectively.

Therefore, the combination of the imbalanced leaning strategy RUS and the OVA method based on SVM and NBM classifiers had higher efficiency in solving the bug reports component classification problem compared with other combinations for the Mozilla, GCC, NetBeans, OpenOffice, and Eclipse datasets.

6. Conclusion

In this article, we propose a new method by combining imbalanced learning technologies with multiclass classification methods to implement bug reports component classification problems. We use four imbalanced processing strategies, RUS, ROS, SMOTE, and AdaCost, to process the data and obtain a balanced dataset. Then, we use the multiclass classification methods, OVO and OVA, based on NBM, KNN, and SVM classifiers for the balanced dataset to classify the bug reports component and solve the bug reports component classification problem. We explored the optimal performance of bug reports component classifications by different combinations of imbalanced learning strategies and classification algorithms. We can better solve the problem of bug reports classification by using the bug component classification to determine the appropriate developer for the bug report. In our work, we could not only reduce the word dimension of the original training set that improves the quality of the training set but also improve the classification performance for bug reports.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (nos. 61902050, 61672122, 61602077, 61771087, 51879027, 51579024, and 71831002), Program for Innovative Research Team in University of Ministry of Education of China (no. IRT 17R13), Fundamental Research Funds for the Central Universities (nos. 3132019501 and 3132019502), and Next-Generation Internet Innovation Project of CERNET (no. NGII20190627).

References

[1] W. Pan, B. Li, J. Liu, Y. Ma, and Bo Hu, "Analyzing the structure of Java software systems by weighted k-core decomposition," *Future Generation Computer Systems*, vol. 83, pp. 431–444, 2018.

[2] F. Rahman and P. Devanbu, "How, and why, process metrics are better," in *Proceedings of the 2013 35th International Conference on Software Engineering (ICSE)*, pp. 432–441, IEEE, San Francisco, CA, USA, May 2013.

[3] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in *Proceedings of the 2013 35th International Conference on Software Engineering (ICSE)*, pp. 382–391, IEEE, San Francisco, CA, USA, May 2013.

[4] Y. Kamei, E. Shihab, B. Adams et al., "A large-scale empirical study of just-in-time quality assurance," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 757–773, 2012.

[5] S. Guo, R. Chen, M. Wei et al., "Ensemble data reduction techniques and Multi-RSMOTE via fuzzy integral for bug report classification," *IEEE Access*, vol. 6, pp. 45934–45950, 2018.

[6] J. Anvik, L. Hiew, and G. C. Murphy, "Coping with an open bug repository," in *Proceedings of the 2005 OOPSLA Workshop on Eclipse Technology Exchange*, pp. 35–39, ACM, San Diego, CA, USA, 2005.

[7] X. Xie, W. Zhang, Y. Yang et al., "Dretom," in *Proceedings of the 8th International Conference on Predictive Models in Software Engineering*, pp. 19–28, ACM, Paris, France, 2012.

[8] X. Xia, D. Lo, X. Wang et al., "Accurate developer recommendation for bug resolution," in *Proceedings of the 2013 20th Working Conference on Reverse Engineering (WCRE)*, pp. 72–81, IEEE, Koblenz, Germany, October 2013.

[9] T. Menzies, Z. Milton, B. Turhan et al., "Defect prediction from static code features: current results, limitations, new approaches," *Automated Software Engineering*, vol. 17, no. 4, pp. 375–407, 2010.

[10] T. Jiang, L. Tan, and S. Kim, "Personalized defect prediction," in *Proceedings of the 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 279–289, IEEE, Silicon Valley, CA, USA, November 2013.

[11] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: a benchmark and an extensive comparison," *Empirical Software Engineering*, vol. 17, no. 4-5, pp. 531–577, 2012.

[12] X. Yang, D. Lo, X. Xia et al., "Deep learning for just-in-time defect prediction," in *Proceedings of the 2015 IEEE International Conference on Software Quality, Reliability and Security*, pp. 17–26, IEEE, Vancouver, Canada, August 2015.

[13] X. Zhu and X. Wu, "Cost-constrained data acquisition for intelligent data preparation," *IEEE Transactions on Knowledge & Data Engineering*, vol. 17, no. 11, pp. 1542–1556, 2005.

[14] H. Jiang, X. Li, Z. Ren, J. Xuan, and Z. Jin, "Toward better summarizing bug reports with crowdsourcing elicited attributes," *IEEE Transactions on Reliability*, vol. 68, no. 1, pp. 2–22, 2019.

[15] W. Fan, S. J. Stolfo, J. Zhang et al., "AdaCost: misclassification cost-sensitive boosting," in *Proceedings of the ICML*, vol. 99, pp. 97–105, Bled, Slovenia, 1999.

[16] A. Tamrawi, T. T. Nguyen, J. M. Al-Kofahi et al., "Fuzzy set and cache-based approach for bug triaging," in *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, pp. 365–375, ACM, 2011.

[17] M. Alenezi, K. Magel, and S. Banitaan, "Efficient bug triaging using text mining," *JSW*, vol. 8, no. 9, pp. 2185–2190, 2013.

[18] C. Kang, Y. Huo, L. Xin et al., "Feature selection and tumor classification for microarray data using relaxed lasso and generalized multi-class support vector machine," *Journal of Theoretical Biology*, vol. 463, pp. 77–91, 2019.

[19] S. Guney and A. Atasoy, "Multiclass classification of n-butanol concentrations with *k*-nearest neighbor algorithm and support vector machine in an electronic nose," *Sensors and Actuators B: Chemical*, vol. 166, pp. 721–725, 2012.

[20] S. K. Ajagekar and V. Jadhav, "Automated approach for DDOS attacks detection based on naive Bayes multinomial classifier," in *Proceedings of the 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 1–5, IEEE, Tirunelveli, India, May 2018.

[21] W. Deng, J. Xu, and H. Zhao, "An improved ant colony optimization algorithm based on hybrid strategies for

scheduling problem," *IEEE Access*, vol. 7, pp. 20281–20292, 2019.

[22] S. Guo, Y. Liu, R. Chen, X. Sun, and X. Wang, "Improved SMOTE algorithm to deal with imbalanced activity classes in smart homes," *Neural Processing Letters*, vol. 50, no. 2, pp. 1503–1526, 2019.

[23] G. Murphy and D. Cubranic, "Automatic bug triage using text categorization," in *Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering*, Reading, MA, USA, 2004.

[24] W. Pan and C. Chai, "Measuring software stability based on complex networks in software," *Cluster Computing*, vol. 22, no. 2, pp. 2589–2598, 2019.

[25] Y. Liu, X. Wang, Z. Zhai, R. Chen, B. Zhang, and Y. Jiang, "Timely daily activity recognition from headmost sensor events," *ISA Transactions*, vol. 94, pp. 379–390, 2019.

[26] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?," in *Proceedings of the 28th International Conference on Software Engineering*, pp. 361–370, ACM, Shanghai, China, 2006.

[27] C. Sun, D. Lo, X. Wang et al., "A discriminative model approach for accurate duplicate bug report retrieval," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-ICSE '10*, pp. 45–54, 2010.

[28] G. Jeong, S. Kim, and T. Zimmermann, "Improving bug triage with bug tossing graphs," in *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pp. 111–120, ACM, 2009.

[29] W. Deng, H. Zhao, L. Zou, G. Li, X. Yang, and D. Wu, "A novel collaborative optimization algorithm in solving complex optimization problems," *Soft Computing*, vol. 21, no. 15, pp. 4387–4398, 2017.

[30] J. Xuan, H. Jiang, Z. Ren et al., "Automatic bug triage using semi-supervised text classification," 2017, https://arxiv.org/abs/1704.04769.

[31] T. Zhang, J. Chen, G. Yang, B. Lee, and X. Luo, "Towards more accurate severity prediction and fixer recommendation of software bugs," *Journal of Systems and Software*, vol. 117, pp. 166–184, 2016.

[32] T. Menzies and A. Marcus, "Automated severity assessment of software defect reports," in *Proceedings of the 2008 IEEE International Conference on Software Maintenance*, pp. 346–355, IEEE, Beijing, China, September-October 2008.

[33] A. Lamkanfi, S. Demeyer, Q. D. Soetens et al., "Comparing mining algorithms for predicting the severity of a reported bug," in *Proceedings of the 2011 15th European Conference on Software Maintenance and Reengineering*, pp. 249–258, IEEE, Oldenburg, Germany, March 2011.

[34] Y. Tian, D. Lo, and C. Sun, "Information retrieval based nearest neighbor classification for fine-grained bug severity prediction," in *Proceedings of the 2012 19th Working Conference on Reverse Engineering*, pp. 215–224, IEEE, Kingston, Canada, October 2012.

[35] R. Chen, S. Guo, X. Wang et al., "Fusion of multi-RSMOTE with fuzzy integral to classify bug reports with an imbalanced distribution," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 12, pp. 2406–2420, 2019.

[36] H. Li, G. Gao, R. Chen, X. Ge, S. Guo, and L.-Y. Hao, "The influence ranking for testers in bug tracking systems," *International Journal of Software Engineering and Knowledge Engineering*, vol. 29, no. 1, pp. 93–113, 2019.

[37] S. Guo, R. Chen, H. Li, T. Zhang, and Y. Liu, "Identify severity bug report with distribution imbalance by CR-SMOTE and ELM," *International Journal of Software Engineering and Knowledge Engineering*, vol. 29, no. 02, pp. 139–175, 2019.

[38] W. Pan, B. Hu, J. Dong, K. Liu, and B. Jiang, "Structural properties of multilayer software networks: a case study in Tomcat," *Advances in Complex Systems*, vol. 21, no. 2, Article ID 1850004, 2018.

[39] H. Jiang, L. Nie, Z. Sun et al., "ROSF: leveraging information retrieval and supervised learning for recommending code snippets," *IEEE Transactions on Services Computing*, vol. 12, no. 1, pp. 34–46, 2019.

[40] J. Guo, Y. Mu, M. Xiong, Y. Liu, and J. Gu, "Activity feature solving based on TF-IDF for activity recognition in smart homes," *Complexity*, vol. 2019, Article ID 5245373, 10 pages, 2019.

[41] W. Pan, B. Song, K. Li, and K. Zhang, "Identifying key classes in object-oriented software using generalizedk-core decomposition," *Future Generation Computer Systems*, vol. 81, pp. 188–202, 2018.

[42] N. Bettenburg, S. Just, A. SchrÃter et al., "What makes a good bug report?," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 308–318, ACM, Atlanta, GA, USA, June 2008.

[43] W. Pan, H. Ming, C. K. Chang, Z. Yang, and D.-K. Kim, "ElementRank: ranking java software classes and packages using multilayer complex network-based approach," *IEEE Transactions on Software Engineering (IEEE TSE)*, 2019.

[44] W. Pan and C. Chai, "Structure-aware mashup service clustering for cloud-based Internet of things using genetic algorithm based clustering algorithm," *Future Generation Computer Systems*, vol. 87, pp. 267–277, 2018.

[45] W. Pan, J. Dong, K. Liu, and J. Wang, "Topology and topic-aware service clustering," *International Journal of Web Services Research*, vol. 15, no. 3, pp. 18–37, 2018.

[46] X. Chen, H. Jiang, Z. Chen, T. He, and L. Nie, "Automatic test report augmentation to assist crowdsourced testing," *Frontiers of Computer Science*, vol. 13, no. 5, pp. 943–959, 2019.

[47] J. Xuan, H. Jiang, Z. Ren et al., "Developer prioritization in bug repositories," in *Proceedings of the 2012 34th International Conference on Software Engineering (ICSE)*, pp. 25–35, IEEE, Zurich, Switzerland, June 2012.

[48] J. Ai, Z. Su, Y. Li, and C. Wu, "Link prediction based on a spatial distribution model with fuzzy link importance," *Physica A: Statistical Mechanics and its Applications*, vol. 527, Article ID 121155, 2019.

[49] W. Deng, H. Zhao, X. Yang, J. Xiong, M. Sun, and B. Li, "Study on an improved adaptive PSO algorithm for solving multi-objective gate assignment," *Applied Soft Computing*, vol. 59, pp. 288–302, 2017.

[50] G. Lang, Q. Li, and L. Guo, "Discernibility matrix simplification with new attribute dependency functions for incomplete information systems," *Knowledge and Information Systems*, vol. 37, no. 3, pp. 611–638, 2013.

[51] H. Zhao, R. Yao, L. Xu, Y. Yuan, G. Li, and W. Deng, "Study on a novel fault damage degree identification method using high-order differential mathematical morphology gradient spectrum entropy," *Entropy*, vol. 20, no. 9, p. 682, 2018.

[52] W. Pan, H. Jiang, H. Ming, C. Chai, Bi Chen, and H. Li, "Characterizing software stability via change propagation simulation," *Complexity*, vol. 2019, Article ID 9414162, 17 pages, 2019.

[53] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[54] S. Kim, K. Pan, and E. E. Whitehead Jr., “Memories of bug fixes,” in *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 35–45, ACM, Portland, OR, USA, November 2006.

[55] E. Shihab, A. Mockus, Y. Kamei et al., “High-impact defects: a study of breakage and surprise defects,” in *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, pp. 300–310, ACM, Szeged, Hungary, September 2011.

[56] H. Zhao, J. Zheng, J. Xu, and W. Deng, “Fault diagnosis method based on principal component analysis and broad learning system,” *IEEE Access*, vol. 7, pp. 99263–99272, 2019.

[57] Y. Xiang, W. Pan, H. Jiang, Y. Zhu, and H. Li, “Measuring software modularity based on software networks,” *Entropy*, vol. 21, no. 4, p. 344, 2019.

[58] http://Mozilla.apache.org/, March, 2019.

[59] http://GCC.apache.org/, March, 2019.

[60] http://NetBeans.apache.org/, March, 2019.

[61] http://OpenOffice.apache.org/, March, 2019.

[62] http://Eclipse.apache.org/, March, 2019.

Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 1495676, 11 pages
https://doi.org/10.1155/2019/1495676

Research Article

Clustering Services Based on Community Detection in Service Networks

Shiyuan Zhou[1,2] **and Yinglin Wang**[1]

[1]*School of Information Management and Engineering, Shanghai University of Finance and Economics, Shanghai 200433, China*
[2]*Jiaxing University, Jiaxing 314001, China*

Correspondence should be addressed to Yinglin Wang; yinlwang@zoho.com.cn

Received 26 September 2019; Accepted 15 November 2019; Published 2 December 2019

Guest Editor: Chunlai Chai

Service-oriented computing has become a promising way to develop software by composing existing services on the Internet. However, with the increasing number of services on the Internet, how to match requirements and services becomes a difficult problem. Service clustering has been regarded as one of the effective ways to improve service matching. Related work shows that structure-related similarity metrics perform better than semantic-related similarity metrics in clustering services. Therefore, it is of great importance to propose much more useful structure-related similarity metrics to improve the performance of service clustering approaches. However, in the existing work, this kind of work is very rare. In this paper, we propose a SCAS (service clustering approach using structural metrics) to group services into different clusters. SCAS proposes a novel metric *A2S* (atomic service similarity) to characterize the atomic service similarity as a whole, which is a linear combination of *C2S* (composite-sharing similarity) and *A3S* (atomic-service-sharing similarity). Then, SCAS applies a guided community detection algorithm to group atomic services into clusters. Experimental results on a real-world data set show that our SCAS performs better than the existing approaches. Our *A2S* metric is promising in improving the performance of service clustering approaches.

1. Introduction

Web service is a new web application mode that has been widely distributed and invoked on the whole Internet [1, 2]. With the rapid development of SaaS (software-as-a-service) and SOA (service-oriented architecture) technologies, web services on the Internet are showing a trend of rapid growth [1, 2]. The traditional way to develop software has moved to service-oriented development. More and more software systems are developed by composing existing services on the Internet, and it has become a promising way to develop software. There are a large number of web services with different types on the Internet, which makes the matching of requirements and services become a complex process that needs to be solved through a process composed of many steps such as clustering, selection, and evaluation of services [3]. Thus, in the initial stage of service matching, how to cluster services has become an urgent problem for the selection of service set that can meet the requirements of users.

Service clustering has been widely regarded as one of the effective ways to improve service matching [1]. In the literature, many different clustering approaches have been proposed to group services into clusters. Among them, the most widely used approaches are based on the mining of features of WSDL documents [4–6]. These approaches first extract key features such as WSDL description, WSDL type, WSDL port, and web service name from the WSDL document. Then, based on these extracted features, they calculate the service similarity between services by using methods like cosine similarity, so as to organize services into clusters. However, due to the fact that WSDL documents usually contain few descriptions, these approaches usually fail to achieve satisfactory clustering results. Worse still, these approaches usually ignore the semantic association between services. Therefore, many other approaches based on the

LDA topic model and its extensions have been proposed [1, 2]. They first extract the hidden topics in web services, and then use low-dimensional topic vectors to encode the functional attributes of web services. Finally, they calculate the similarity between services so as to organize services into different clusters. The experimental results show that these topic model-based approaches can achieve good results. However, the WSDL documents for web services are usually short in length, containing very little semantic (or functional) information that can be used by LDA-based clustering approaches. Thus, the semantic-related similarity metrics seem to not perform better than structure-related similarity metrics in clustering web services [1]. Therefore, it is of great importance to propose much more useful structure-related similarity metrics to promote the performance of service clustering approaches.

The purpose of this work is to propose some novel structure-related similarity metrics so as to combine them together to quantify the similarity between services as a whole. It is expected that the novel similarity metrics can be used to improve the performance of service clustering approaches. In this paper, we propose a SCAS (service clustering approach using structural metrics) approach to group services into different clusters. First, SCAS uses an ASAN (atomic service affiliation network) to represent composite services, atomic services, and their relationships. Based on the ASAN, we propose our first structural metric, *C2S* (composite-sharing similarity) which is a similarity owing to the sharing of common composite services. Second, by projecting the ASAN onto the atomic services, we build an ASDN (atomic service dependency network) to denote atomic services and their relationships. Based on ASDN, we propose our second structural metric, *A3S* (atomic-service-sharing similarity) which is a similarity owing to the sharing of common atomic services. Third, we propose the final structural metric, *A2S* (atomic service similarity) to characterize the atomic service similarity as a whole, which is a linear combination of *C2S* and *A3S*. Finally, we propose a SSN (service similarity network) and a community detection algorithm, GUIDA (Guided community detection algorithm), to find the community structures in the SSN. The communities correspond to the atomic service clusters. GUIDA only takes the similarity between atomic services as its input and can determine the number of clusters for the atomic service set automatically. Our experimental study is carried out on a real-world service data set collected from a famous service directory ProgrammableWeb (PWeb) (https://www.programmableweb.com). The comparative studies show that our approach performs better than *C2S*-based approaches and semantic-based approaches.

The main contributions of this paper can be summarized as follows:

(i) The introduction of a novel structural metric to characterize the service similarity, which combines two structure similarity metrics simultaneously, i.e., one is proposed to characterize the composite-sharing similarity and the other one is used to characterize atomic-service-sharing similarity. Our metric is very different from the existing structural similarity metrics which usually only character the composite-sharing similarity, ignoring the atomic-service-sharing similarity.

(ii) The application of a parameter-free community detection algorithm to detect the communities and group services into clusters. Our approach only takes the similarity between services as the input and can determine the number of clusters in the service set automatically. It is very different from the existing approaches which needs to set the number of clusters beforehand to cluster services.

(iii) A real-world data set is built to validate our approach empirically, which is very different from those approaches using a simulation way to validate their approaches.

The reminder of this paper is organized as follows. Section 2 reviews the related work. Section 3 presents our SCAS approach, with the focus on service networks, similarity metrics, and the GUIDA algorithm used to group services into clusters. Section 4 presents the empirical validation of our SCAS approach. We draw conclusions in Section 5.

2. Related Work

As mentioned above, service clustering is one of the effective technologies to help the matching of requirements and services. Its main goal is to group services into different clusters according to the similarity of their functions. Generally, services in the same cluster are similar to each other while services in different clusters are different. Till now, lots of service clustering approaches have been proposed. According to the information that they used to characterize service similarity, they can be roughly divided into two categories: function-based service clustering approaches [1, 7–11] and nonfunction-based clustering approaches [12–16].

Function-based service clustering approaches employed the functional information of services (e.g., description documents, tags) to group services into clusters. Chen et al. [7, 8] proposed an improved service clustering approach which integrates WSDL documents and service tags to improve clustering accuracy. They think that tags represent the functional characteristics of services, which can be combined with WSDL documents to much more accurately determine the functional category of services. Shi et al. [9] first organized words in the service description of all services according to semantics using the Word2Vec, and then they proposed an improved clustering approach by considering the auxiliary function of the words which belong to the same cluster with the words in the service description document. Liu et al. [10] first trained a preliminary SVM classifier based on a small set of manually labeled samples. Then, they recommended potential labels for other services that are not manually labeled based on the SVM classifier. Finally, services are clustered based on these labeled samples. Cao et al. [11] extracted the hidden topic information of mashup services based on the LDA topic model, and then clustered the mashup services based on the LDA topic model. Chen et al. [8] proposed an enhanced probabilistic topic model (WT-LDA), which can model Web service description documents and service tag information simultaneously so as to group services into clusters. Shi et al. [9] proposed a Mashup-API-Tag probabilistic topic model to model the service description, label information, and the composite relationship information among services, so as to improve the accuracy of topic information extraction. Pan and Chai [1] proposed a novel mashup service clustering approach based on a structural similarity and a genetic algorithm-based clustering algorithm. Their approach can group mashup services into clusters effectively and determine the number of clusters automatically.

Nonfunction-based clustering approaches employed the nonfunctional information of services (e.g., QoS and physical locations) to group services into clusters. Liu et al. [12] proposed a service clustering algorithm based on an ontology which contains the information of service name, performance, interface, and QoS attributes. Zhou et al. [13] used a genetic algorithm to group services into clusters by using the QoS information of services. To avoid converging to a local optimum, they introduced the concept of entropy to measure and improve the population diversity of their genetic algorithm. Chen et al. [14] proposed a service clustering algorithm based on the historical QoS data with similar physical characteristics to ensure that services in the same cluster have similar physical environment characteristics. These clustering algorithms only consider nonfunctional attributes, thus usually have a relatively small value of execution complexity. However, these algorithms usually do not have good scalability because nonfunctional attributes of services are often difficult to obtain and unstable.

Generally, the structure-related similarity metrics perform better than semantic-related similarity metrics in grouping services into clusters since the WSDL documents and service descriptions do not always carry a lot of function-related information. However, to the best of our knowledge, there is only one structure-related similarity metric, *C2S* (composite-sharing similarity) [3]. Therefore, it is of great importance to propose much more useful structure-related similarity metrics to promote the performance of service clustering approaches. In this work, we proposed an atomic-service-sharing similarity metric. Our metric is different from *C2S*. Our metric is based on the sharing relationship of atomic services while *C2S* is based on the sharing relationship of composites.

3. SCAS Approach

In this paper, SCAS approach is proposed to group services into different clusters. It is composed of four steps: (1) SCAS uses an ASAN to represent composite service, atomic services, and their relationships and proposes the *C2S* metric. (2) SCAS builds an ASDN to denote atomic services and their relationships and proposes the *A3S* metric. (3) SCAS proposes the *A2S* metric to characterize the atomic service similarity as a whole. (4) SCAS applies a community detection algorithm to find the service clusters in the set of atomic services. The workflow of the SCAS approach is shown in Figure 1.

3.1. Services Profile Crawling. To group services into clusters, the first work that we should do is to craw the profile of services to be clustered. Generally, the profile of services are usually registered in a registration center where people can share their own developed services with other people over the world. Furthermore, people can find the right services to meet their requirements by searching and browsing in the registration center.

To the best of our knowledge, there are many famous service registration centers such as ProgrammableWeb (PWeb) (https://www.programmableweb.com), myExperiment (https://www.myexperiment.org/home), and Biocatalogue (https://www.biocatalogue.org). To carry out the service clustering work, we will crawl the profile of services stored in these registration centers. The information includes service name, the lower level services each composite service used, and the category each service belongs to. The information will be stored in the local database as to reduce the noises and finally build the data set for SCAS to perform the service clustering task.

3.2. Service Network Construction. Complex network theory has been widely used in software engineering to quantify software structure [17, 18], identify key classes [19, 20], and measure software quality [21–23]. In this work, to quantify the service similarity as a whole, we also apply the complex network theory and build three service networks, i.e., ASAN (atomic service affiliation network), ASDN (atomic service dependency network), and SSN (service similarity network), which are defined as follows.

3.2.1. Atomic Service Affiliation Network

Definition 1. ASAN is an affiliation network that can be denoted formally as ASAN = (V_c, V_a, E), where V_c denotes the node set of composite service, V_a denotes the node set of atomic services, and E denotes the unweighted undirected edge set between every pair of composite service and atomic service if the composite service uses the atomic service, i.e., if $v_i \in V_c$ uses $v_j \in V_a$, then it follows that $\{v_i, v_j\} \in E$. ASAN is essentially a two-mode graph. As a two-mode graph, any edge can only exist between the different node sets V_c and V_a, i.e., $V_c \cap V_a = \varnothing$. To be specific, if $(v_i, v_j) \in E$, then it follows that $v_i \in V_c$ and $v_j \in V_a$. E will be associated with a $|V_c| \times |V_a|$ adjacency matrix ψ to encode the "use" relationships between a specific pair of composite service and atomic service. The entry of ψ, ψ_{ij}, is defined as

$$\psi_{ij} = \begin{cases} 1, & \{v_i, v_j\} \in E, \\ 0, & \text{otherwise}. \end{cases} \tag{1}$$

We use a simple example shown in Figure 2 to show the idea to build the ASAN. The example is chosen from our

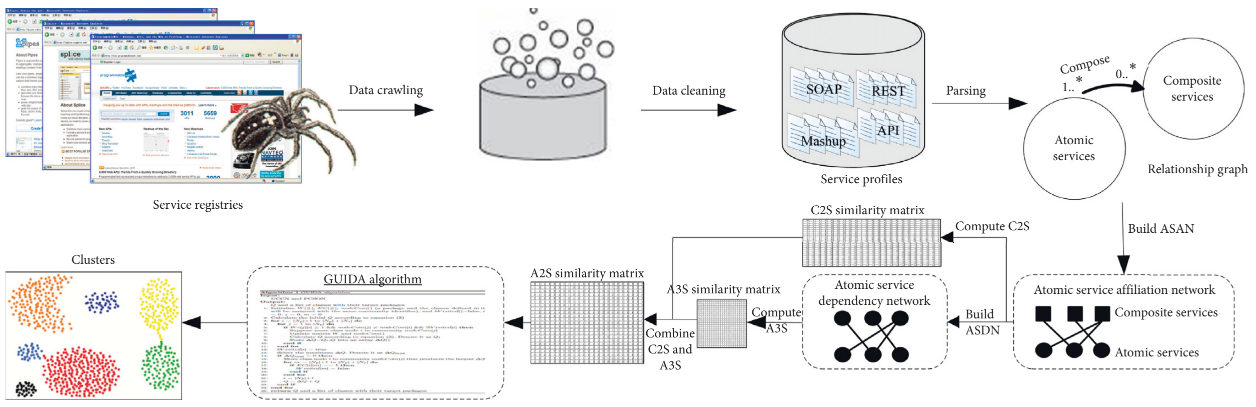


Figure 1: The workflow of SCAS. The workflow is adapted from that of [1, 2].

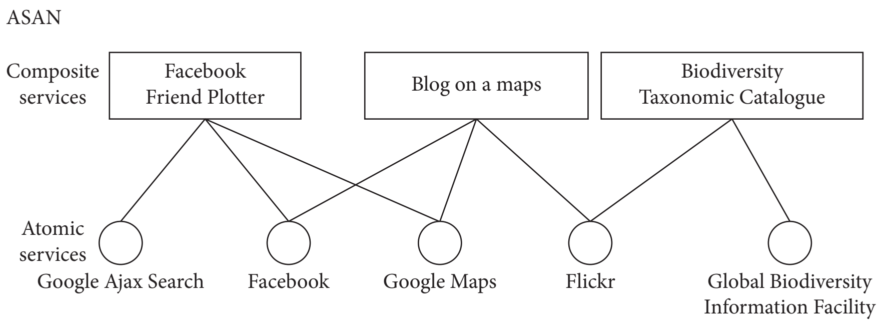


Figure 2: Illustration of ASAN.

data set used in Section 4 and is borrowed from [1, 2]. As shown in the example, the composite service (i.e., mashup "Facebook Friend Plotter (https://www.programmableweb.com/mashup/facebook-friend-plotter)") consists of three atomic services (i.e., APIs "Google Ajax Search (https://www.programmableweb.com/api/google-ajax-search)," "Facebook (https://www.programmableweb.com/api/facebook)," and "Google Maps (https://www.programmableweb.com/api/google-maps)"). Thus, in the corresponding ASAN, there exist three edges between composite service "Facebook Friend Plotter" and the other three atomic services "Google Ajax Search," "Facebook," and "Google Maps." By taking a similar way, we can establish other edges in the ASAN.

3.2.2. Atomic Service Dependency Network. ASAN describes the macro composition information between composite services and atomic services, and it also implicitly reflects the composition potential between the atomic services that exist in the same composite services. In other words, if two atomic services $v_j \in V_a$ and $v_k \in V_a$ are used together in ≥1 composite service(s), it follows that $v_j \in V_a$ and $v_k \in V_a$ have the probability to be composed together to build a composite service. Thus, in this work, we will build an ASDN (atomic service dependency network) to capture atomic services and their composition potential.

Definition 2. ASDN can be formally denoted as ASDN $= (V_a, E_a)$, where V_a denotes the node set of atomic services (V_a is the same as that in ASAN) and E_a is the edge set between atomic services, signifying the composition potential between the atomic services, i.e., if $(v_j, v_k) \in E_a$, and then it follows that v_j and v_k have the probability to be composed together. Each edge will be assigned a weight to denote the number of composite services that the two atomic services commonly participate in. E_a will be associated with a $|V_a| \times |V_a|$ adjacency matrix $\psi^{\mathbf{a}}$ to encode the composition relations between a specific pair of atomic services. The entry of $\psi^{\mathbf{a}}$, ψ^{a}_{ij}, is defined as

$$\psi^{a}_{ij} = \begin{cases} w, & \{v_i, v_j\} \in E_a, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

where w is the weight assigned to the edge $\{v_i, v_j\} \in E_a$, denoting the number of composite services that the two atomic services commonly participate in.

We also use a simple example shown in Figure 3 to show the idea to build the ASDN. ASDN is constructed from the example shown in Figure 2. As is shown in the example, since the composite service "Facebook Friend Plotter" uses the function of atomic services "Google Ajax Search" and "Google Maps," an edge between the node of "Google Ajax Search" and the node of "Google Maps" will be established in Figure 3. We establish all other edges in Figure 3 by taking a similar way.

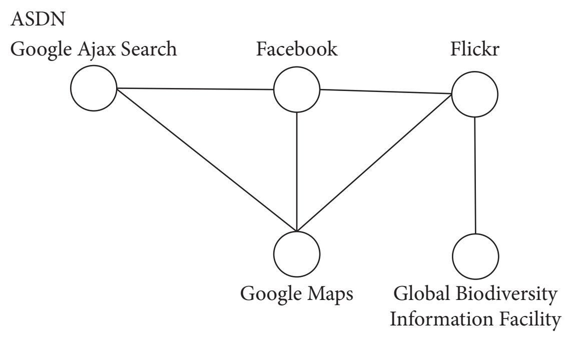


Figure 3: Illustration of ASDN.

In this work, ASDN is built by using the one-mode projection of ASAN on atomic services. Specifically, if two atomic services are used together in ≥1 composite service(s), then it follows that an edge will be established between the two atomic services in ASDN.

3.2.3. Service Similarity Network. We use SSN to denote services and their similarity relationships. Thus, SSN can be defined as follows:

Definition 3. SSN can be formally denoted as $\mathrm{SSN} = (V_{\mathrm{a}}, E_s)$, where V_{a} denotes the node set of atomic services (V_{a} is the same as that in ASAN and ASDN) and E_s is the edge set between atomic services, signifying the similarity relationships between the atomic services, i.e., if $(v_j, v_k) \in E_s$, and then it follows that v_j and v_k are similar to each other. Each edge will be assigned a weight to denote the similarity between the two atomic services. E_s will be associated with a $|V_{\mathrm{a}}| \times |V_{\mathrm{a}}|$ adjacency matrix ψ^{s} to encode the similarity relations between a specific pair of atomic services. The entry of ψ^{s} (ψ^{s}_{ij}) is defined as

$$\psi^{s}_{ij} = \begin{cases} A2S(s,t), & \{s,t\} \in E_s, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

where $A2S(s,t)$ is the similarity between atomic services s and t, which will be computed in Section 3.3.

3.3. Service Similarity Metrics. To group services into clusters, we usually need to quantify service similarity as a whole. In this work, service similarity is measured from the perspective of structure. As mentioned above, in this work, we quantify service similarity using two structure-related metrics, $C2S$ (composite-sharing similarity) and $A3S$ (atomic-service-sharing similarity). The service similarity as a whole is measured as the integration of $C2S$ and $A3S$.

If the composite-sharing similarity between two atomic services s and t is denoted as $C2S(s,t)$, then it can be defined as

$$C2S(s,t) = \frac{|N_s \cap N_t|}{|N_s \cup N_t|}, \tag{4}$$

where N_s and N_t denote the number of composite services that atomic services s and t are used, respectively.

If the atomic-service-sharing similarity between two atomic services s and t is denoted as $A3S(s,t)$, then it can be defined as

$$A3S(u,v) = \begin{cases} \dfrac{\psi^{a}_{st}}{\sum_{k=1}^{n} \psi^{a}_{st}}, & \sum\limits_{k=1}^{|V_{\mathrm{a}}|} \psi^{a}_{kt} \neq 0, \\ 0, & \text{otherwise}. \end{cases} \tag{5}$$

Obviously, $A3S(s,t)$ is the ratio of ψ^{a}_{st} relative to the total number of weights on the edges that link to v_t; otherwise, $A3S(s,t)$ equals 0.

Furthermore, the atomic-service-sharing similarity between atomic services s and t is denoted as the maximum value of $A3S(s,t)$ and $A3S(t,s)$, i.e.,

$$A3S(s,t) = A3S(t,s) = \max(A3S(s,t), A3S(t,s)). \tag{6}$$

Then, the total similarity between atomic services s and t, $A2S(s,t)$, can be defined as

$$A2S(s,t) = \lambda \times C2S(s,t) + (1-\lambda) \times A3S(s,t), \tag{7}$$

where λ is the weight assigned to the component of equation (7).

In this work, λ is determined by CV (coefficient of variation) [24], which refers to a statistical metric that is used to measure the distribution of data points in a data series around the mean. CV is a helpful statistic in comparing the degree of variation from one data series to the other. CV is computed by deriving the ratio between the standard deviation and the mean. Thus, CV is defined as

$$\mathrm{CV} = \frac{\mathrm{std}}{\mathrm{mean}}, \tag{8}$$

where std and mean are the standard deviation and mean of the sample, respectively.

In this work, we will first compute $C2S(s,t)$ (or $A3S(s,t)$) for all pairs of atomic services s and t ($s, t \in V_{\mathrm{a}}$). Then, we compute the std and mean for the $C2S(s,t)$ (or $A3S(s,t)$) set so as to obtain the CV for $C2S$ (or $A3S$). If we use CV_{c2s} (or CV_{a3s}) to denote the CV for $C2S$ (or $A3S$), then it follows that

$$\lambda = \frac{\mathrm{CV}_{c2s}}{\mathrm{CV}_{c2s} + \mathrm{CV}_{a3s}}. \tag{9}$$

3.4. Community Detection Algorithm

3.4.1. Modularity Q. In the literature, many different community detection algorithms have been proposed to organize nodes in a network into communities. Among them, a popular way used is to optimize a Q index [21, 25]. Q is short for modularity, which is used to quantify the quality of a specific partition of a network into communities. Q is based on the measurement of density of edges within communities compared with edges between communities. In this work, we also applied Q to measure the quality of a specific clustering solution of atomic services. For SSN, a weighted undirected network, Q can be defined as

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta\left(c_i, c_j\right), \tag{10}$$

where m denotes the sum of the weight on all edges in the SSN, A_{ij} denotes the weight assigned to edge $\{i, j\}$ or $(\{j, i\})$, k_i and k_j are the sum of the weight on the edges linking to nodes i and j, respectively, c_i and c_j denote the communities into which nodes i and j are organized, respectively, and δ returns 1 when c_i equals c_j, 0, otherwise.

In the process of community detection in SSN, once an atomic service is moved from one community to another community, Q will be recalculated to decide whether to accept or reject this movement.

3.4.2. Community Detection Algorithm Flow. In this work, a Guided community detection algorithm (GUIDA) is introduced to perform the community detection task in SSN. GUIDA is originally proposed to refactor the class structure [22, 23]. The "guide" means GUIDA is different from the traditional community detection algorithms using a guidance way. Its "guide" reflects mainly in two aspects:

(i) Guide the initial division: composite services consist of a set of related atomic services. Thus, composite services can be seen as the initial communities of atomic services. There is no need to start the community detection process using a random way by assigning each atomic service to a random community. In this work, we will guide the initial division by grouping the atomic services in the same composite service to a same community. However, this will make some atomic services belong to different communities since some atomic services may participate in the composition of ≥1 composite service(s). Thus, in this work, we first assign atomic services participating in the composition of ≥1 composite service(s) to different communities. Then, for the atomic services participating in the composition of only one composite service, they will be assigned to the community of the atomic services that they are most similar to that in the SSN. Finally, for the remainder of the atomic services, we will assign them to a random community.

(ii) Guide the atomic-service-moving process: GUIDA groups atomic services into communities by a series of atomic-service-moving operations. In this work, the atomic-service-moving operation can only happen at atomic services linking to other atomic services in ASDN and belonging to different communities.

In Algorithm 1, we show the flow of GUIDA, where array depend $[\cdot][\cdot]$ stores $\psi^{\mathbf{a}}$ of ASDN; array sim $[\cdot][\cdot]$ stores $\psi^{\mathbf{s}}$ of SSN; deg $[\cdot]$ is an array storing the degree of atomic services in ASAN, e.g., atomic service i has degree deg $[i]$; community $[\cdot]$ is an array storing the community identifiers for all atomic services, e.g., atomic service i belongs to the community with identifier community $[i]$; bInit$[\cdot]$ is an array with the Boolean type denoting whether atomic service i has been initialized or not; and bVisited$[\cdot]$ is an array with the Boolean type denoting whether node i has been visited or not.

As is shown in Algorithm 1, the most dominant steps of GUIDA are step 32 to step 53 with the loop. Thus, the computational complexity of GUIDA is O $(|V_{\mathrm{a}}|^2)$.

4. Empirical Study

In this section, we performed experiments to investigate the effectiveness of our SCAS approach. For the experiment environment, all the experiments were carried out on a PC at 2.6 GHz with 8 GB of RAM.

4.1. Research Questions. We performed experiments with the aim to address the following two research questions (RQs):

(i) RQ1: Does our SCAS approach perform better than other *C2S*-based approaches?

We integrate *C2S* and *A3S* together to quantify the similarity between atomic services and use a guided community detection algorithm to group atomic services into clusters. Thus, we wish to know whether our SCAS approach performs better than approaches that only use *C2S* to quantify the similarity between atomic services.

(ii) RQ2: Does our SCAS approach perform better than other semantic-based approaches?

There are many research works on clustering services (see Section 2). Thus, we wish to know whether our SCAS approach performs better than some of these approaches, especially those based on semantic similarities.

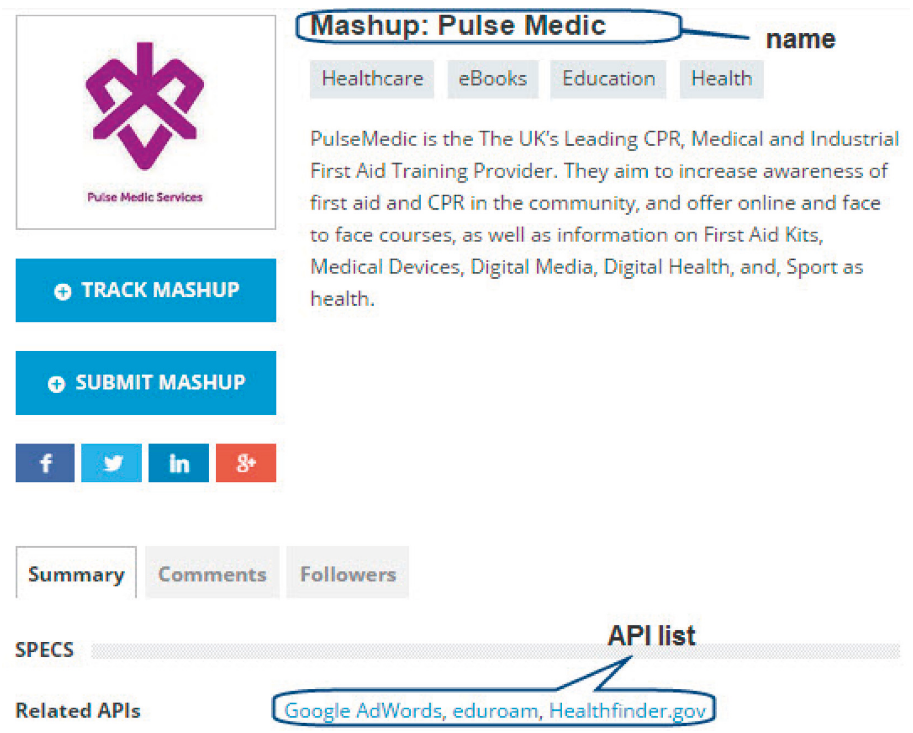
4.2. Objects of Study. As mentioned above, PWeb is a famous service repository and widely used as a benchmark data set for experiments of service clustering approaches [1]. PWeb provides the profile of APIs and mashups, including their names, descriptions, tags, and providers. Every mashup is composed of ≥1 API(s) which will be listed in the profile of the mashup. Mashups can be regarded as composite services and APIs can be regarded as atomic services. Thus, PWeb meets the requirement of our SCAS approach and can be used as our object of study.

We crawled the profile of mashups stored in the PWeb till December 14, 2011. Specifically, we crawled the name of mashups and the APIs that mashups used. Figure 4 illustrates the information that we crawled for a mashup.

Specifically, we crawled the name of the mashup (i.e., "Pulse Medic") and the APIs it used (i.e., "Google AdWords," "Healthfinder.gov," and "eduroam"). Finally,

```
Input:
      ASAN, ASDN, and SSN
Output:
      $Q$ and all the communities
 (1) Initialize depend [·][·], sim [·][·], community [·], bInit[·] = false, and bVisited[·] = false
 (2) Calculate the degree of each atomic service in ASAN and store them in deg[ ]
 (3) for $i = 1$ to $|V_a|$ do
 (4)   $cnt = 0$;
 (5)   if deg [$i > 1$ then]
 (6)     community [·] = $cnt$++;
 (7)     bInit [$i$] = true;
 (8)   end if
 (9) end for
(10) for $i = 1$ to $|V_a|$ do
(11)   if deg[$i == 1$ && !bInit [$i$] then]
(12)     MAX = −1;
(13)     max Index = −1;
(14)     for $j = 1$ to $|V_a|$ do
(15)       if sim[$i$][$j$] > MAX then
(16)         MAX = sim[$i$][$j$];
(17)         max Index = $j$;
(18)       end if
(19)     end for
(20)     if MAX != −1 || $maxIndex$ != −1 then
(21)       community [·] = community [·];
(22)       bInit [$i$] = true;
(23)       $cnt$++;
(24)     end if
(25)   end if
(26)   if $cnt == |V_a|$ then
(27)     break;
(28)   end if
(29)   $i = 1$;
(30) end for
(31) Calculate the initial $Q$ according to equation (10)
(32) for $i = 1$ to $|V_a|$ do
(33)   for $j = 1$ to $|V_a|$ && $j! = i$ do
(34)     if depend[$j$ [$i$] $\geq 1$ && community[$j$] $\neq$ community[$i$] && !bVisited[$i$]
         then
(35)       Suppose we move atomic service $i$ to community community[$j$]
(36)       Update community[·]
(37)       Calculate $Q$ according to equation (10). Denote it as $Q_t$
(38)       Store $\Delta Q = Q_t - Q$ into an array $\Delta Q$[ ]
(39)     end if
(40)   end for
(41)   bVisited [$i$] = true
(42)   Select the maximum $\Delta Q$. Denote it as $\Delta Q_{max}$
(43)   if $\Delta Q_{max} > 0$ then
(44)     Move atomic service $i$ to community community[$j$] that produces the largest $\Delta Q$
(45)     for $m = 1$ to $|V_a|$ do
(46)       if depend[$i$ [$m$] == 1 then]
(47)         bVisited[$m$] = false
(48)       end if
(49)     end for
(50)       $i = 1$
(51)       $Q = \Delta Q + Q$
(52)   end if
(53) end for
(54) return $Q$ and all the communities
```

Algorithm 1: GUIDA algorithm.


Mashup: Pulse Medic

Healthcare eBooks Education Health

PulseMedic is the The UK's Leading CPR, Medical and Industrial First Aid Training Provider. They aim to increase awareness of first aid and CPR in the community, and offer online and face to face courses, as well as information on First Aid Kits, Medical Devices, Digital Media, Digital Health, and, Sport as health.

SPECS

Related APIs Google AdWords, eduroam, Healthfinder.gov

Figure 4: The information that we crawled for a mashup.

our data set for experiments contains 6,362 mashups and 982 APIs.

4.3. Baseline Approaches. The purpose of this work is to improve the performance of service clustering approaches that are based on structure-related metrics. Thus, our SCAS will be compared with the approach using *C2S*, which is named as CSCA (*C2S*-based service clustering approach). CSCA uses *C2S* to quantify the similarity between atomic services and uses GUIDA to group atomic services into clusters.

We cannot make a thorough comparison with the approaches that we have reviewed in Section 2. The main reason is that we cannot implement their approaches since their work does not report the details of their approaches. But, we try our best to compare our approach with the approaches that we reviewed in Section 2. In this work, we compare our approach with the following two approaches in the related work, which are originally proposed to cluster Web services.

(i) TCluster: it uses the information of tags to quantify the semantic similarity between APIs by using LDA and further applies k-means to organize APIs into clusters.

(ii) DTCluster: DTCluster is very similar to TCluster. The only difference is DTCluster also utilizes the information of the description.

4.4. Experiment Process and Results. We follow the main procedures shown in Figure 1 to parse the data set, build the ASAN, ASDN, and SSN, compute *C2S*, *A3S*, and *A2S*, and apply GUIDA group APIs into clusters.

Figures 5(a) and 5(b) show the ASAN and ASDN we constructed from PWeb, respectively. The position of the nodes in ASAN and ASDN is all computed using the circular algorithm in Pajek (Pajek: http://vlado.fmf.uni-lj.si/pub/networks/pajek/).

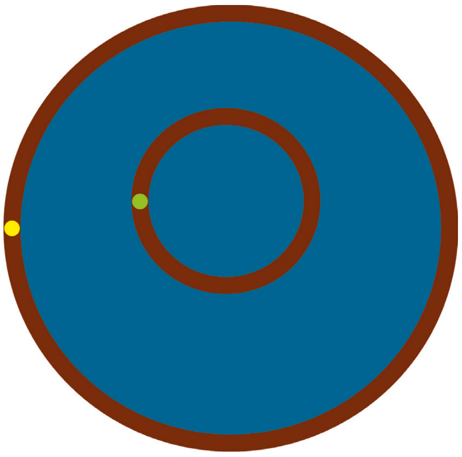
Based on ASAN and ASDN, we compute the *C2S* and *A3S* similarity metrics between APIs so as to compute the *A2S* to quantify API similarity as a whole. Then, we can use GUIDA to group APIs into clusters. In the experiment, GUIDA returns the communities in the SSN when it terminates at $Q = 0.521102$.

4.5. Analysis of the Results. In this section, we analyze the obtained service clustering results with the aim to answer the research questions that we have presented in Section 4.1. To compare SCAS with CSCA, we should measure the quality of their clustering solutions.

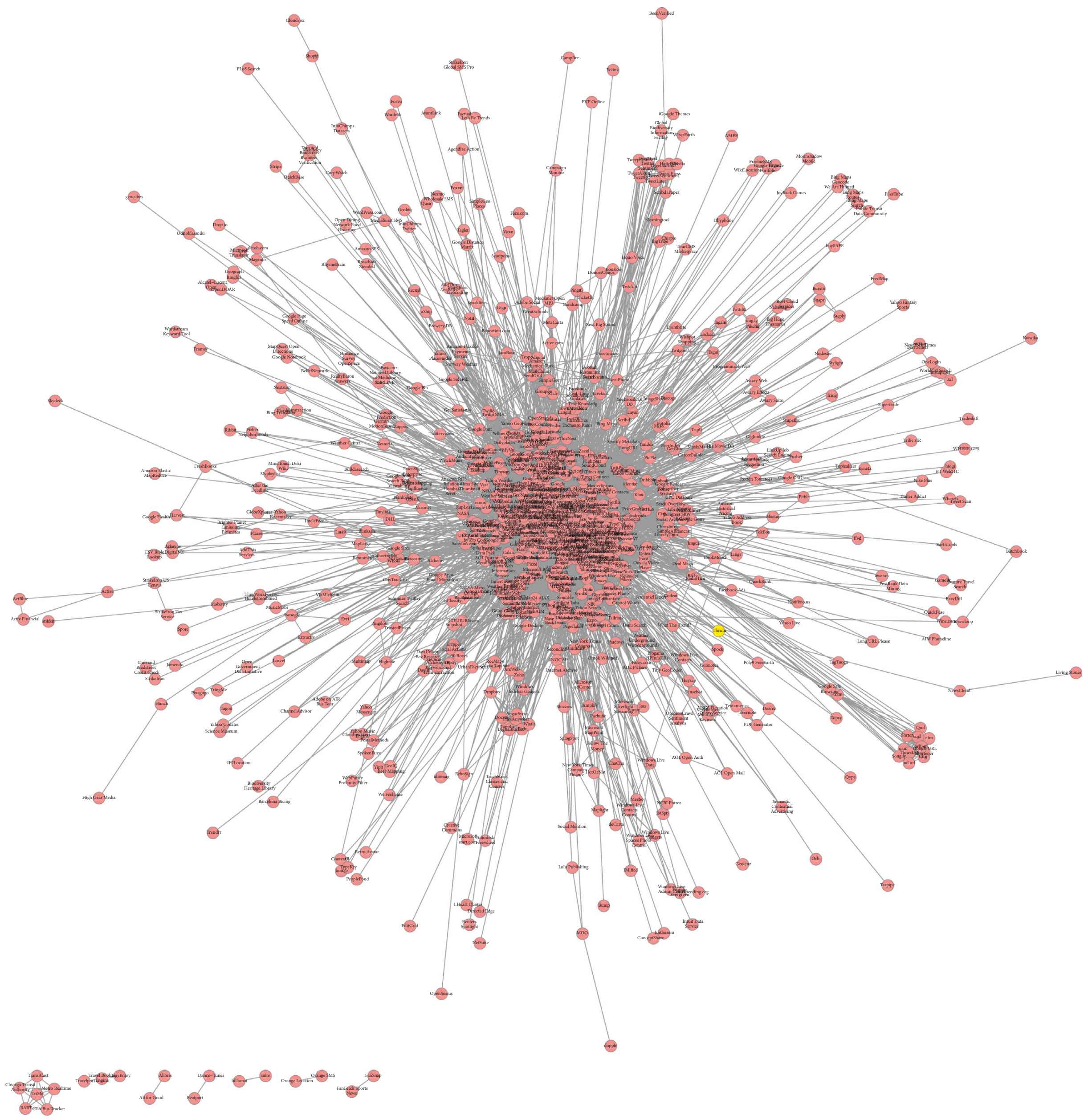
In this work, we apply a set of criteria that are widely used in information retrieval, i.e., Precision, Recall, and *F*-Measure (i.e., F_1 and F_5) [1]. To compute the value of these criteria, we use the classification system in PWeb as the standard clustering results, i.e., APIs in the same category are viewed as APIs in the same cluster. These criteria can be defined as follows:

(i) *True Positive (TP)*. A *TP* decision assigns two similar atomic services to the same cluster

(ii) *False Positive (FP)*. A *FP* decision assigns two dissimilar atomic services to the same cluster



(a)



(b)

FIGURE 5: (a) ASAN and (b) ASDN constructed from PWeb.

(iii) *False Negative (FN)*. A *FN* decision assigns two similar atomic services to different clusters

(iv) *True Negative (TN)*. A *TN* decision assigns two dissimilar atomic services to different clusters

$$\begin{aligned}\text{Precision} &= \frac{TP}{TP+FP},\\ \text{Recall} &= \frac{TP}{TP+FN},\\ F_1 &= \frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}},\\ F_5 &= \frac{26*\text{Precision}*\text{Recall}}{25*\text{Precision}+\text{Recall}}.\end{aligned} \tag{11}$$

Generally, a larger value of Precision, Recall, and *F*-Measure indicates a better service clustering solution [17–19].

4.5.1. Does Our SCAS Approach Perform Better than Other C2S-Based Approaches? SCAS combined *C2S* and *A3S* to quantify the similarity between atomic services while CSCA applied *C2S*. The only difference between SCAS and CSCA is the similarity metrics that they used. Thus, by comparing the results obtained by SCAS with those of CSCA, we can know whether SCAS performs better than CSCA.

Figure 6 shows the performance comparisons of the two approaches, SCAS and CSCA, when applied to the data set. Obviously, SCAS outperforms CSCA with respect to *Precision*, *Recall*, F_1, and F_5. It indicates that the combination of *C2S* and *A3S* can improve the performance of a service clustering approach.

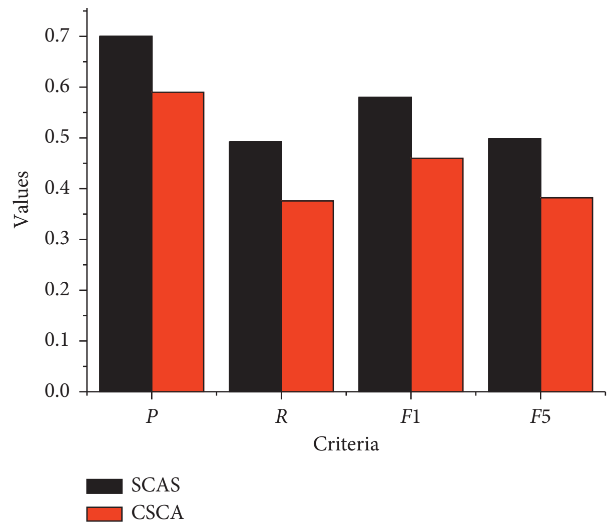


FIGURE 6: Performance comparison of SCAS and CSCA. P, *Precision*; R, *Recall*; *F*1 and *F*5 denote F_1 and F_5, respectively.

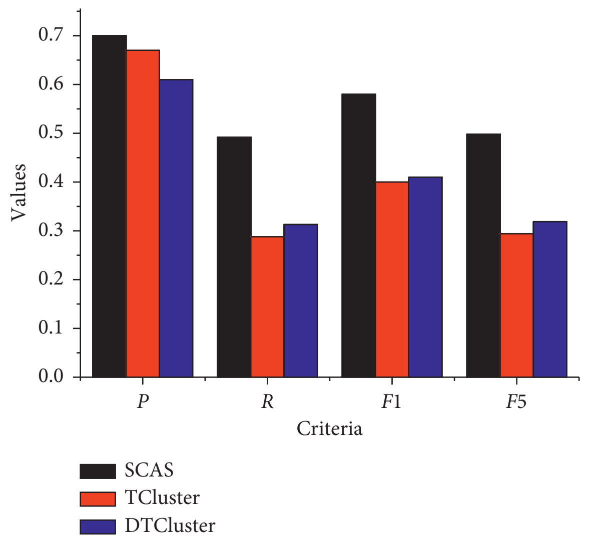


FIGURE 7: Performance comparison of SCAS and TCluster (or DTCluster). P, Precision; *R*, Recall; *F*1 and *F*5 denote F_1 and F_5, respectively.

4.5.2. Does Our SCAS Approach Perform Better than Other Semantic-Based Approaches? As mentioned in Section 2, there are many research works on clustering services. In this section, we performed experiments to check whether our SCAS approach performs better than some of these approaches which are based on semantic similarities, i.e., TCluster and DTCluster. As mentioned above, the only difference between SCAS and TCluster (or DTCluster) is the former applied structure-based similarity metrics while the latter applied semantic-based similarity metrics. Thus, by comparing the results obtained by SCAS with those of TCluster (or DTCluster), we can know whether structure-based similarities are better than semantic-based similarities in service clustering.

Figure 7 shows the performance comparisons of the three approaches, SCAS, TCluster, and DTCluster, when applied to the data set. Obviously, SCAS outperforms TCluster and DTCluster with respect to Precision, Recall, F_1, and F_5. It indicates that structure-based similarity metrics are better than semantic-based similarities in service clustering.

5. Conclusions

In this work, we proposed a SCAS approach to group services into different clusters. It proposed an improved structure-related metric, *A2S*, to quantify service similarity and proposed a guided community detection algorithm to organize services into clusters. Comparative studies with other related approaches on a real-world data set show that SCAS performs better than some of the existing approaches.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 61375053).

References

[1] W. Pan and C. Chai, "Structure-aware Mashup service clustering for cloud-based Internet of Things using genetic algorithm based clustering algorithm," *Future Generation Computer Systems*, vol. 87, pp. 267–277, 2018.

[2] W. Pan, J. Dong, K. Liu, and J. Wang, "Topology and topic-aware service clustering," *International Journal of Web Services Research*, vol. 15, no. 3, pp. 18–37, 2018.

[3] B. Jiang, L. Y. Ye, W. F. Pan, and J. L. Wang, "Service clustering based on the functional semantics of requirements," *Chinese Journal of Computer*, vol. 41, no. 6, pp. 1255–1266, 2018.

[4] W. Liu and W. Wong, "Web service clustering using text mining techniques," *International Journal of Agent-Oriented Software Engineering*, vol. 3, no. 1, pp. 6–26, 2009.

[5] L.-J. Zhang and J. Zhang, "Service oriented solution modeling and variation propagation analysis based on architectural building blocks," *International Journal of Web Services Research*, vol. 10, no. 4, pp. 39–61, 2013.

[6] K. Elgazzar, A. E. Hassan, and P. Martine, "Clustering WSDL documents to bootstrap the discovery of Web services," in *Proceedings of the 8th IEEE International Conference on Web Services (ICWS 2010)*, pp. 147–154, Miami, FL, USA, July 2010.

[7] L. Chen, L. K. Hu, Z. B. Zheng et al., "WTcluster: Utilizing tags for Web services clustering," in *Proceedings of the 9th International Conference on Service-Oriented Computing (ICSOC 2011)*, pp. 204–218, Paphos, Cyprus, December 2011.

[8] L. Chen, Y. Wang, Q. Yu, Z. Zheng, and J. Wu, "WT-LDA: User tagging augmented LDA for Web service clustering," in *Proceedings of the 11th International Conference on Service-Oriented Computing (ICSOC 2013)*, pp. 162–218, Berlin, Germany, December 2013.

[9] M. Shi, J. X. Liu, D. Zhou, M. D. Tang, and B. Q. Cao, "WE-LDA: A word embeddings augmented LDA model for Web services clustering," in *Proceedings of the 15th IEEE International Conference on Web Services (ICWS 2017)*, pp. 9–16, Honolulu, HI, USA, June 2017.

[10] X. Liu, S. Agarwal, C. Ding, and Q. Yu, "An LDA-SVM active learning framework for Web service classification," in *Proceedings of the 14th IEEE International Conference on Web Services (ICWS 2016)*, pp. 49–56, San Francisco, CA, USA, July 2016.

[11] B. Q. Cao, "Mashup service clustering based on an integration of service content and network via exploiting a two-level topic model," in *Proceedings of the 14th IEEE International Conference on Web Services (ICWS 2016)*, pp. 212–219, San Francisco, USA, July 2016.

[12] J. X. Liu, K. Q. He, J. Wang, and D. Ning, "A clustering method for web service discovery," in *Proceedings of the International Conference on Service Computing (SCC 2011)*, pp. 729-730, Washington, DC, USA, July 2011.

[13] J. Zhou and S. Li, "Semantic web service discovery approach using service clustering," in *Proceedings of the International Conference on Information Engineering and Computer Science (ICIECS 2009)*, pp. 1–5, Wuhan, China, December 2009.

[14] F. Chen, S. Yuan, and B. Mu, "User-QoS-based Web service clustering for QoS prediction," in *Proceedings of the 13rd IEEE International Conference on Web Services (ICWS 2015)*, pp. 583–590, New York, USA, July 2015.

[15] N. Zhang, J. Wang, and Y. T. Ma, "Mining domain knowledge on service goals from textual service descriptions," *IEEE Transactions on Services Computing*, 2017.

[16] R. B., Xiong, J. Wang, N. Zhang, and Y. T. Ma, "Deep hybrid collaborative filtering for Web service recommendation," *Expert Systems with Applications*, vol. 110, pp. 191–205, 2018.

[17] W. Pan, B. Li, J. Liu, Y. Ma, and B. Hu, "Analyzing the structure of Java software systems by weighted *K*-core decomposition," *Future Generation Computer Systems*, vol. 83, pp. 431–444, 2018.

[18] W. Pan, B. Hu, J. Dong, K. Liu, and B. Jiang, "Structural properties of multilayer software networks: a case study in Tomcat," *Advances in Complex Systems*, vol. 21, no. 2, Article ID 1850004, 2018.

[19] W. Pan, B. Song, K. Li, and K. Zhang, "Identifying key classes in object-oriented software using generalized *k*-core decomposition," *Future Generation Computer Systems*, vol. 81, pp. 188–202, 2018.

[20] W. F. Pan, H. Ming, C. K. Chang, Z. J. Yang, and D.-K. Kim, "ElementRank: ranking Java software classes and packages using multilayer complex network-based approach," *IEEE Transactions on Software Engineering*, 2019.

[21] Y. Xiang, W. Pan, H. Jiang, Y. Zhu, and H. Li, "Measuring software modularity based on software networks," *Entropy*, vol. 21, no. 4, p. 344, 2019.

[22] W. Pan and C. Chai, "Measuring software stability based on complex networks in software," *Cluster Computing*, vol. 22, no. S2, pp. 2589–2598, 2019.

[23] W. F. Pan, H. B. Jiang, H. Ming, C. L. Chai, B. Chen, and H. Li, "Characterizing software stability via change propagation simulation," *Complexity*, vol. 2019, Article ID 9414162, 17 pages, 2019.

[24] J. T. McClave, P. G. Benson, and T. Sincich, *Statistics for Business and Economics*, Prentice Hall, Upper Saddle River, NJ, USA, 2008.

[25] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review*, vol. 69, no. 6, Article ID 066133, 2004.