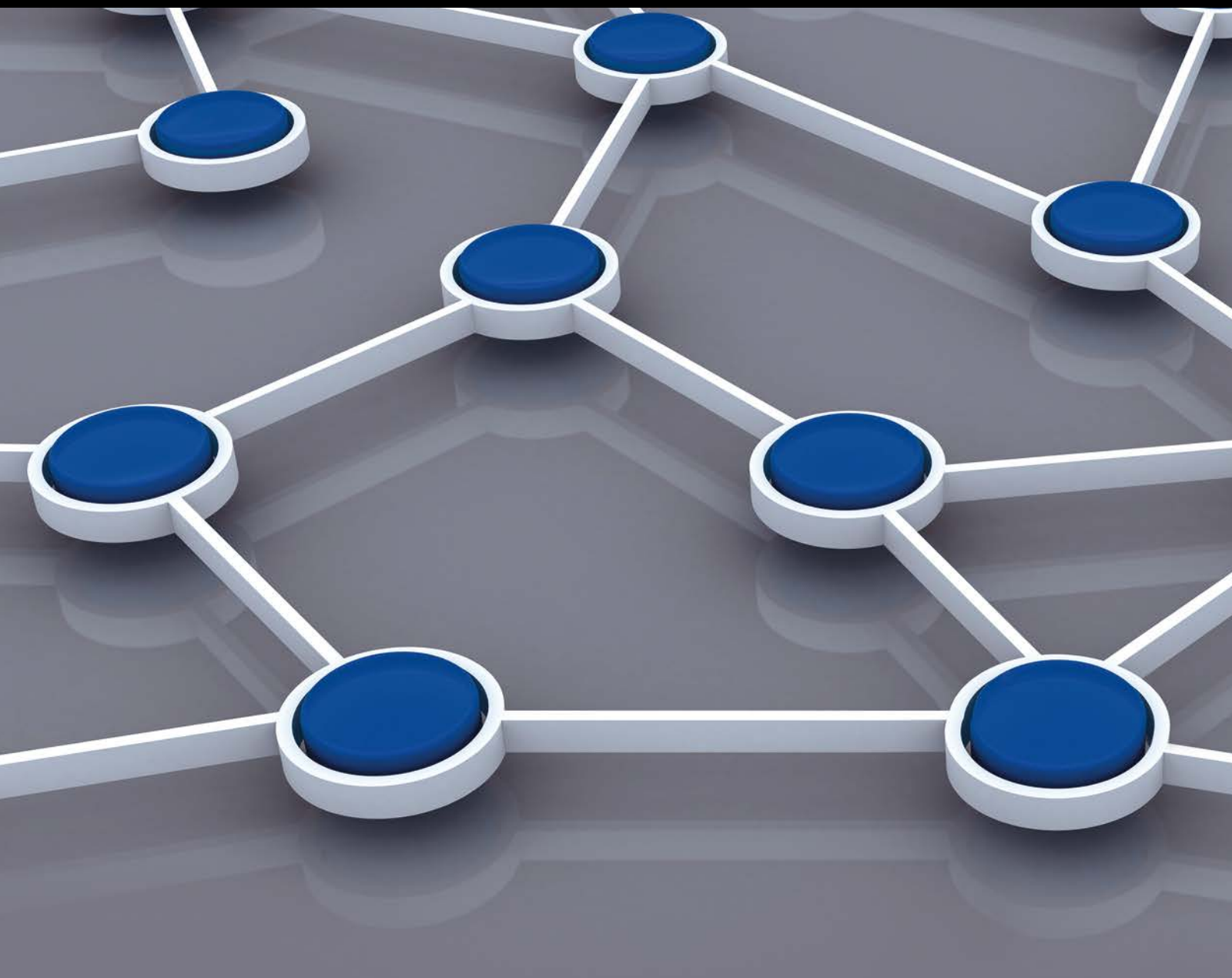
International Journal of Distributed Sensor Networks

Internet of Things, Linked Data, and Citizen Participation as Enablers of Smarter Cities

Guest Editors: Diego López-de-Ipiña, Liming Chen, Antonio Jara, Erik Mannens, and Yingshu Li



Internet of Things, Linked Data, and Citizen Participation as Enablers of Smarter Cities

International Journal of Distributed Sensor Networks

Internet of Things, Linked Data, and Citizen Participation as Enablers of Smarter Cities

Guest Editors: Diego López-de-Ipiña, Liming Chen, Antonio Jara, Erik Mannens, and Yingshu Li

Editorial Board

Contents

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2016, Article ID 2595847, 2 pages
http://dx.doi.org/10.1155/2016/2595847

Editorial

Internet of Things, Linked Data, and Citizen Participation as Enablers of Smarter Cities

Diego López-de-Ipiña,[1] Liming Chen,[2] Antonio Jara,[3] Erik Mannens,[4] and Yingshu Li[5]

[1] *Deusto Institute of Technology, DeustoTech, University of Deusto, 48007 Bilbao, Spain*
[2] *De Montfort University, The Gateway, Leicester LE1 9BH, UK*
[3] *University of Applied Sciences Western Switzerland (HES-SO), Sierre, Valais, Switzerland*
[4] *iMinds, Ghent University, AA Tower, Technologiepark 19, 9052 Zwijnaarde, Belgium*
[5] *Georgia State University, Atlanta, USA*

Correspondence should be addressed to Diego López-de-Ipiña; dipina@deusto.es

Received 5 May 2016; Accepted 5 May 2016

Web of Data and Internet of Things are enabling technologies that pave the way for next generation urban services. These services will play a crucial role in future interactions between the city and the citizens, giving them the impression of facing Smarter Cities, that is, cities that not only do manage their resources more efficiently but also are aware of the citizen needs. The aim of this special issue has been to bring together research results on the areas of Linked Data, Internet of Things, and smartphone-mediated interaction to assemble service ecosystems that may give place to Smarter Cities, that is, those that are actually aware of the real needs and demands of their citizens.

This special issue includes 6 articles covering the topics of *crowdsourcing*, that is, cooperation of individuals and IT systems to provide solutions where machine automation cannot reach *data correlation of big volumes of data* to enhance mobility in cities, *analysis of access logs to both social networks and IT systems* in order to predict the next location or the next data chunk to be requested by a user, and, finally, *novel AI techniques to explore interdependencies among different factor*s to help in the decision making process within a city.

The article entitled "CooperSense: A Cooperative and Selective Picture Forwarding Framework Based on Tree Fusion" by H. Chen et al. explores the topic of mobile crowd photographing for local sensing, allowing encountering participants to only exchange those pictures relevant to each other by applying a tree based selection mechanism.

The article "A Context-Driven Worker Selection Framework for Crowd-Sensing" by J. Wang et al. proposes a novel worker selection framework, called WSelector, to more precisely select appropriate workers by taking various contexts into account. To achieve this goal, it first provides programming time support to help task creators define constraints. Then its runtime system adopts a two-phase process to select workers who are not only qualified but also more likely to undertake a crowd-sensing task.

The article entitled "Urban Impedance Computing Based on Check-In Records" by Z. Yu et al. analyses the concept of urban impedance, that is, the travelling cost between the origin and destination locations as an important indicator of urban accessibility. For that, it combines check-in records obtained from mobile social networks and road networks data to calculate and adjust the various parameters of the model, including path length, number and angle of turns, number and direction of junctions, and population density.

The article "Twitter Can Predict Your Next Place of Visit" by A. Chauhan et al. proposes a predictor for users' next place of visit using their past tweets. For that, it computes the probabilities of visiting different types of places using a bank of binary classifiers and Markov models.

The article "Prefetching Scheme for Massive Spatiotemporal Data in a Smart City" by L. Xiong et al. explores access patterns to develop a prefetching scheme, which can effectively improve system I/O performance and reduce

user access latency. A prefetching scheme based on spatial-temporal attribute prediction, called STAP, is developed which maps the history of user access requests to the spatiotemporal attribute domain by analysing the characteristics of spatiotemporal data in a smart city. Notably, the STAP scheme mines the user access patterns and constructs a predictive function to predict the user's next access request.

Finally, the article "Factor Knowledge Mining Using the Techniques of AI Neural Networks and Self-Organizing Map" by P.-K. Wu and T.-C. Hsiao offers a hybrid technique combining artificial neural networks (ANN) and self-organizing maps (SOM) as a way to explore factor knowledge, namely NNSOM. This technique is applied to analyse the most important factor to organize a night market in Taiwan.

Overall, this special issue explores how to bring together machine and human intelligence in order to understand better data flows in the context of a city and support the decision making process within the cities of the future.

Diego López-de-Ipiña
Liming Chen
Antonio Jara
Erik Mannens
Yingshu Li

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2016, Article ID 6968014, 13 pages
http://dx.doi.org/10.1155/2016/6968014

Research Article

CooperSense: A Cooperative and Selective Picture Forwarding Framework Based on Tree Fusion

Huihui Chen,[1,2] Bin Guo,[1] and Zhiwen Yu[1]

[1]*Northwestern Polytechnical University, Xi'an 710129, China*
[2]*Luoyang Institute of Science and Technology, Luoyang 471023, China*

Correspondence should be addressed to Huihui Chen; chenhuihui.cn@gmail.com

Received 14 December 2015; Accepted 29 March 2016

Academic Editor: Diego López-de-Ipiña

Mobile crowd photographing has become a major crowd sensing paradigm, which allows people to use cameras on smart devices for local sensing. In MCP, pictures taken by different people in close proximity or time period can be highly similar and different MCP tasks have diverse constraints or needs to deal with such duplicate data. In order to save the network cost and improve the transmitting efficiency, pictures will be preselected by mobile clients and then uploaded to the server in an opportunistic manner. In this paper, CooperSense, a multitask MCP framework for cooperative and selective picture forwarding, was designed. Based on the sensing context of pictures and task constraints, CooperSense structures sequenced pictures into a hierarchical context tree. When two participants encounter, their mobile clients will just exchange their context trees and at the same time automatically accomplish forwarding high-quality pictures to each other via a tree fusion mechanism. Via virtual or real pruning and grafting, mobile clients learn which picture should be sent to the encounter and which one should be abandoned. Our experimental results indicate that the transmission and storage cost of CooperSense are much lower comparing with the traditional Epidemic Routing (ER) method, while their efficiency is almost the same.

1. Introduction

People using cameras to get pictures is a typical application of mobile crowd sensing. Nowadays, widespread smart devices are driving more and more people to use cameras as visual sensors to relook at this world. It raised a particular type of participatory sensing paradigm: Mobile Crowd Photographing (MCP) [1, 2].

MCP applications collect uploaded pictures and send them to the backend server for further collective knowledge mining. Typical MCP achievements include monitoring the pollution of creeks [3], reposting and sharing fliers distributed in urban areas [4], and gathering pictures of buildings for 3D city modeling [5]. These MCP applications assume that transmission is available anytime and anywhere. However, communication will become unavailable and the internet will become inaccessible in some cases, such as an emergency. For example, 36 cellular base stations were out of service due to the power cut caused by the Tianjin explosion accident in 2015 [6]. Then some MCP applications were developed to report scenes of emergency situations (e.g., disasters or fire) [7–9]. Also, because pictures are of large size (around 200 KB for a useable zoomed-out picture to 3 MB for a full-size picture taken by an iPhone 6S), they cost much network flow and money (about 0.29 RMB for 1 MB on average in China) when the commercial network is connected (e.g., 3/4G). Therefore, some researchers studied opportunistically forwarding pictures [10] with ER (Epidemic Routing) [11] based on some free and high-speed transmissions like Bluetooth, WiFi, and so on. Researches on Delay Tolerant Network (DTN) [9, 12] and Mobile Opportunistic Network (MON) [13, 14] are typical examples.

MON and DTN utilize cooperative Work Nodes (WN) to store, carry, and forward pictures, which has been mentioned by Uddin et al. [7], Weinsberg et al. [12], Jiang et al. [15], Zhao et al. [16], and Yu et al. [17]. Their works leverage the cooperative sensing scheme to eliminate sampling redundancy so as to save energy. Cooperative WNs refer to those participants who join in to collect or forward pictures in this paper.

In MCP, it is possible that participants perform the same task and collect similar pictures. As a result, not all WN-carried pictures are valuable. Then what kind of pictures are valuable? Different applications give different answers. For instance, in the disaster area after an earthquake, pictures of buildings taken from different directions and in different places might be more valuable than those taken with the same spatial context [7]. Valuable pictures refer to those about-to-receive ones that are different from carried ones. Therefore, in order to select valuable pictures, MCP applications will keep additional context information on saved pictures. However, existing picture dissemination schemes in MON/DTN are not sufficient since the utilization of photographing contexts has not been effectively explored.

Based on requirements of forwarding pictures through an opportunistic way, five important constraints for the solution are summarized as follows.

(1) Self-Driven Delivery. Under the submitting permission from participants, the self-driven picture delivery is more applicable comparing with the manual transmission. Therefore, opportunistic forwarding is applicable.

(2) Energy-Saving. Energy limitation has been stated in all mobile applications, and thus our method must keep balance between the energy-saving and performance.

(3) Coverage Assurance. Selecting representative high-quality data and then forwarding or uploading them is a more effective way to decrease the traffic and storage consumption, but only on the condition that the selection method guarantees the coverage of the selected pictures from those raw ones.

(4) Low Delay. People out of the observation area need pictures on the scene to know about the situation, so the receiving delay of these pictures must be as minor as possible.

(5) Useless Epidemic Termination. Whenever WNs encounter, they exchanges their pictures via an epidemic way. Once a picture is uploaded to the data center, those duplications of this picture carried by WNs are useless and thus should not be forwarded any more. In this case, useless epidemic must be terminated as soon as possible.

In order to address these constraints mentioned above, we will use data fusion or aggregation in our picture forwarding method. Picture data fusion in MCP is different from numeric data fusion (e.g., an average value of temperature) [18, 19] or splicing pictures in wireless sensor network [20], and it usually relies on a data selection scheme [7, 9, 15]. In other words, based on some selection constraints, the WN will only forward valuable pictures selected from a picture set union of this WN and others' picture collections.

CooperSense is a novel cooperative and selective picture forwarding framework for varied MCP tasks. It leverages the cooperation of opportunistically encountered participants to decrease the delay of uploading pictures and utilizes epidemic picture routing with fusion to eliminate sampling redundancy. Specifically, the following contributions are mainly made:

(1) Proposing a generic framework based on a cooperative data fusion to address the problem of how to select and forward high-utility picture for MCP tasks; we, especially, are the first to consider useless epidemic termination for energy-saving in its cooperative picture forwarding framework.

(2) Developing a novel and efficient tree model to structure the information of picture collection, named picture tree (PicTree), which satisfies various MCP task needs and constraints for data fusion; photographing contexts as locality, timestamp, and shooting angle are used to build this tree structure.

(3) Proposing a PicTree fusion based method for picture selection; without image processing, PicTree fusion can efficiently select valuable pictures and abandon useless ones.

We have developed a visualized simulator to simulate and analyze forwarding behaviors of WNs, including PicTree fusion based and Epidemic Routing based [11] (ER-based) picture forwarding schemes. Results show that CooperSense saves over 50% traffic and storage space comparing with the traditional ER-based method.

The rest of the paper will be organized as follows. Section 2 outlines the related work on common data fusion and picture selection and fusion methods for MCP. Section 3 presents the CooperSense framework and data models followed by PicTree-based picture set fusion method in Section 4. We will present and discuss experimental results in Section 5, and we conclude the paper and speculate about the future work in Section 6.

2. Related Works

CooperSense well addressed problems in the cooperative picture forwarding procedure in MON/DTN, which utilizes a data fusion method and a data selection method. In this section, we will introduce some related works about these two methods.

2.1. Data Fusion. Studies focusing on the issue of data forwarding in MON/DTN during a disaster/emergency either introduces DTN architecture to an efficient disaster response network [21] or eliminates redundancies in disaster related content [7] for fast message delivery out of the disaster zone. The performance of simple routing protocols (e.g., flooding) for data forwarding is generally poor [10]. Many generic opportunistic forwarding protocols, such as Epidemic Routing (ER) and its variations, for example, Epidemic Routing with Fusion (ERF) [16], have been proposed to route data among mobile nodes.

Picture data cannot be fused as numeric data can. For the visual sensor network, Chow et al. proposed the method to process and combine overlapping portions of pictures in neighbor sensors so as to reduce the energy spent on image transmission [20]. But, in MCP, battery capacity of mobile devices is limited. In order to save the power used on image processing, the representative pictures should be simply selected in the first place by mobile devices as a fusion result. Picture set fusion is regarded as picture selection, and we will introduce some related works in the following.

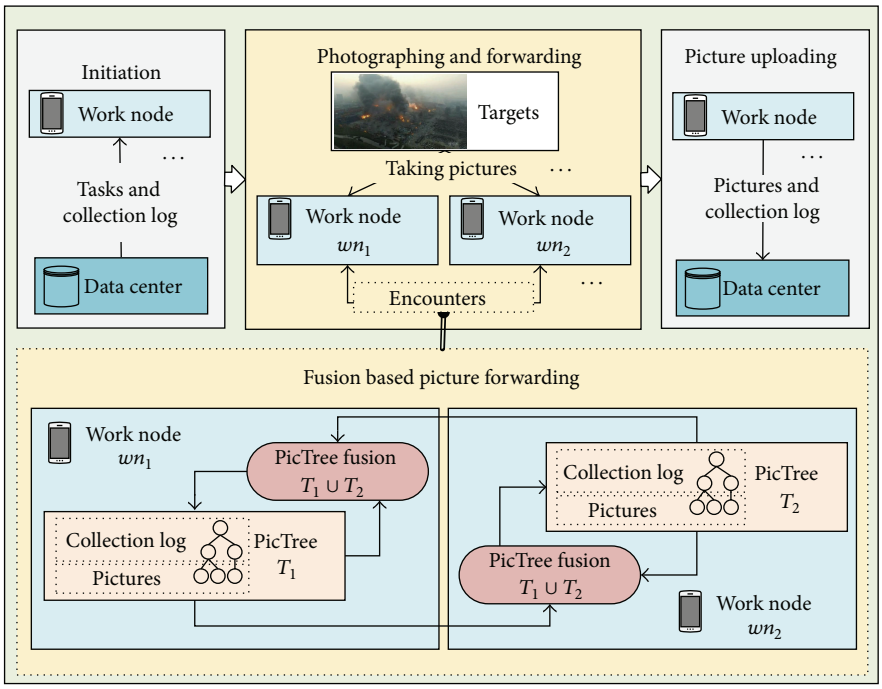


Figure 1: The framework of CooperSense contains three stages.

2.2. Picture Selection. MCP applications can measure similarity of pictures based on different features due to their various knowledge mining purposes. Available features include both visual features and photographing contexts [2], such as the shooting direction, the timestamp, the location, and the ambient light. For example, FlierMeet [4] collects crowdsourced photos of fliers for cross-space public information reposting, tagging, and sharing, and it only selects one picture of every flier based on the visual feature. PhotoCity [5] leverages a game-based incentive way to collect photos of buildings in cities for 3D city modeling, and it selects pictures of buildings taken from different directions. Photonet [7] and the work in [9] are disaster response applications, by using which a group of survivors and first responders may be able to survey the damage and send images to a rescue center without the access to functional communication infrastructure. These two applications focus on collecting visually different pictures based on the location and visual features. Therefore, in order to satisfy various situations of MCP applications, CooperSense will measure similarity of pictures based on the location, the shooting direction, and the timestamp for picture selection.

In order to save traffic and decrease the communication overhead, some MCP applications use a preselection way to forward valuable pictures, for example, PhotoNet [7] and MediaScope [15]. Preselection means that valuable pictures are selected by mobile clients before they are uploaded. However, those applications are application-specific and cannot address varied needs/constraints (e.g., spatiotemporal contexts, single or multiple shooting angles to a sensing target). COUPON [16] addresses data fusion in MON, but it is a generic and formal solution and needs further improvement for MCP. Therefore, CooperSense can address data fusion issue for MCP in MON and utilizes a data fusion for the preselection process. For MCP, data fusion is actually merging picture sets of different WNs.

2.3. Picture Set Fusion. For MCP, most applications will focus on how to efficiently assess similarity of pictures and select diverse pictures. For example, WNs of PhotoNet [7] a priori forward other pictures that can increase content-diversity of others' picture sets. The work in [9] detects critical content (e.g., fires, road blocks) on the images through image processing, and those detected images with critical content can be forwarded faster than those without critical content in order to quickly address critical events. Mediascope [15] also selects different pictures based on their locations. Although they tried to decrease duplicate picture transmission through picture set fusion, the comparing algorithm to select pictures is queue-based, which is time-consuming. In order to improve the efficiency, CooperSense uses a faster tree-based algorithm to aggregate picture sets, which will be introduced in the following sections.

3. CooperSense Framework

3.1. An Overview. CooperSense aims to satisfy the sensing requirements of various MCP tasks. It uses an efficient and effective picture set fusion method and an overhead-saving selective forwarding method based on the picture set fusion. In this section, we will present three data models and the framework of CooperSense.

As shown in Figure 1, CooperSense has three-stage workflow: (1) the *initiation stage*, (2) the *photographing and forwarding stage*, and (3) the *picture uploading stage*. We will introduce the mechanism via the workflow at the WN side.

Firstly, in the *initiation* stage, when a person is entering the observed area, he/she will download task requirements to his/her smart device, namely, the WN. Since some tasks may be long-term, the WN can download requirements of both new tasks and old unexpired tasks. Meanwhile, in order to terminate the epidemic of useless pictures in time, the *collection log* of unexpired old tasks is also downloaded. The collection log is a PicTree structure and contains the information of carried and uploaded pictures. Both MCP server and WNs have collection logs. According to the collection log contributed by other WNs, any WN can delete useless pictures and forward useful pictures to other WNs in the next stage, that is, the photographing and forwarding stage. Here, useless pictures refer to WN-carried duplicates of uploaded pictures, so deleting them means terminating useless epidemic.

Secondly, in the *photographing and forwarding* stage, people take pictures according to tasks' requirements, and WNs automatically forward valuable pictures to other WNs when they encounter. For instance, assuming that the WN wn_1 encounters a WN wn_2, then the forwarding consists of two steps: (i) wn_1 sends its collection log to wn_2 and wn_2 merges wn_1's collection log with its own, named PicTree fusion; (ii) wn_2 can select pictures during the PicTree fusion procedure and send them back to wn_1, named PicTree fusion based forwarding. Also, using this two-step procedure, wn_2 sends pictures to wn_1.

Finally, in the *picture uploading* stage, the WN leaves the observed area and uploads picture collections when the data center becomes accessible. Also, uploading picture is based on PicTree fusion to save traffic.

3.2. Data Modeling

3.2.1. Task Model. An MCP task is denoted by TSK: $\langle tid, whn, whr, ntr\rangle$. *tid* is the identifier of the task. *whn* denotes the start time and the end time of the task. *whr* is a set of geographic coordinates and denotes where the targets are. *ntr* denotes the required number of pictures, and it is used to determine whether to stop an unexpired task or not. By the way, the task posted to participants is described in natural language and TSK is its formalization.

Since pictures taken in close proximity or time period may be highly similar, the implicit task constraints are defined for the picture selection in most MCP applications, for example, PhotoNet [7], Smartphoto [22], and Fliermeet [4]. The task constraint is denoted by TSKC: $\langle tid, c\_loc, c\_tm, c\_agl\rangle$. If two pictures are different, *c_loc* is the minimum geographic distance of their locations, *c_tm* denotes the minimum time span of their picture-taking time points, and *c_agl* is the minimum angle of their shooting directions.

3.2.2. Picture Model. A picture captured by a smart device is denoted with PIC: $\langle wid, tid, loc, tm, agl, img\rangle$. *wid* is the identifier of the WN, *tid* is the identifier of the task that the picture is taken for, *loc* and *tm* denote the coordinate and the timestamp of the photographing, respectively, *agl* denotes the angle between the shooting direction and the north pole, and *img* is the image file.

Table 1: Layers and tree nodes' structures.

Layer	Attribute of the picture	Symbol	Attributes of the tree node
Task	*tid*	L^t	$\langle vLink, tid\rangle$
Location	*loc*	L^l	$\langle vLink, loc, c\_loc\rangle$
Date and time	*tm*	L^d	$\langle vLink, ts, te, c\_tm\rangle$
Shooting angle	*agl*	L^a	$\langle vLink, agl, c\_agl\rangle$
Image	*img*	L^i	$\langle vLink, img\rangle$

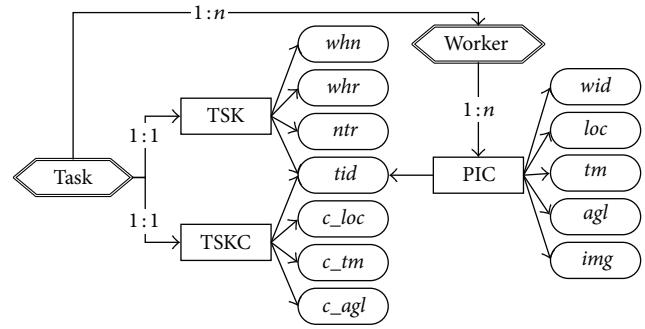


Figure 2: Data model relationship between pictures and a task.

3.2.3. PicTree Model. The picture set carried by a WN is denoted by P^{wid}, and its corresponding PicTree-based collection log is a PTree structure [2] and presents similarity degree of pictures. Reference [2] proposed PTree, a tree structure to cluster pictures. PTree's layers can be exchanged to increase the clustering efficiency. Since we focus on picture forwarding in DTN or MON, we used a generic fixed layer mapping in this paper; that is, each layer is one-to-one mapped onto the feature of a picture. As shown in Table 1, five layers from top to down are the task layer, the location layer, the date and time layer, the shooting angle layer, and the image layer, denoted by L^t, L^l, L^d, L^a, and L^i, respectively. Image layer is the bottom layer, where all images are stored.

Each layer of the PicTree is mapped onto a feature of the picture, so properties of tree nodes in different layers are denoted by different vectors. As shown in Table 1, most attributes of tree nodes correspond to attributes of both the task and the picture shown in Figure 2. Attributes (*ts*, *te*) in the layer L^d denote a time span. Particularly, each tree node has a common attribute, that is, the virtual link denoted by a Boolean variable *vLink*, to mark whether the tree node is actually live or not. *vLink* is an important attribute for PicTree fusion, and we use an example to explain the usage of it. A WN wn_1 uploads a PIC x to the data server and it then removes x. After that, wn_1 encounters wn_2 who also carries the PIC x; then the result is that wn_2 forwards the uploaded PIC x to wn_1, which is clearly a useless copy operation. In order to avoid such forwarding, our method introduces the parameter *vLink* to mark uploaded pictures. With the information from *vLink*, wn_1 can reject wn_2's sending-data request and also tells wn_2 that x is already available at the server.

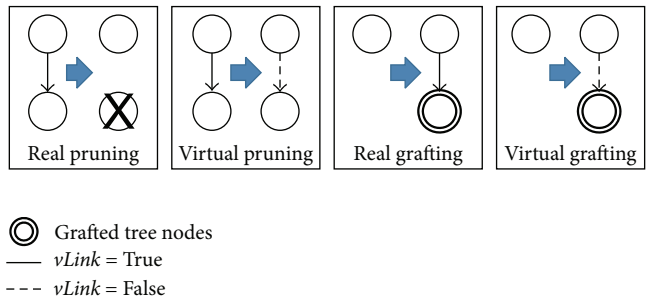


Figure 3: Four kinds of basic operations for PicTree fusion.

Traditional tree fusion utilizes two operations, pruning and grafting. In order to terminate useless epidemic, we proposed four operations for PicTree fusion shown in Figure 3, namely, *virtual* or *real grafting* and *virtual* or *real pruning*. The tree node whose *vLink* is False means that it is *virtual pruned*, and its offspring tree nodes can be removed, that is, *real pruned*. Consequently, if *vLink* of the tree node is False, then, the fact that this node is grafted is called *virtual grafting*, and if not, the grafting is called *real grafting*.

4. Methodology

We use the PicTree structure to store the collection log, including all the context information of carried pictures. Through merging two PicTrees of two WNs, useful pictures can be selected while useless ones can be deleted. Next, we will introduce PicTree fusion followed by PicTree fusion based picture forwarding.

4.1. PicTree Fusion

4.1.1. Tree Node Fusion. PicTree fusion is a procedure of merging one PicTree with another PicTree through merging or grafting tree nodes. The root node is meaningless and tree nodes mentioned in this paper do not include the root node.

In order to merge PicTrees, two methods are proposed as follows.

(1) A method to assess the similarity of two tree nodes: as shown in Table 2, $\langle tid, c\_loc, c\_tm, c\_agl\rangle$ is a TSKC instance, and *tid*, *loc*, *ts*, *te*, and *agl* are attributes of tree nodes. Two tree nodes N_1 and N_2 on the same layer are similar if they satisfy the condition in Table 2 and their parent tree nodes are also similar. Therefore, two pictures are similar only if they are in two similar leaf tree nodes.

(2) A method to value parameters of the merged tree node: if two tree nodes are merged, parameters of the new tree node will be valued with methods shown in Table 3.

Further, there are seven basic fusion rules for tree node fusion as follows:

(1) Any nonleaf tree nodes can be merged if they are similar.

Table 2: Conditions for accessing similarity of two tree nodes N_1 and N_2 on different layers.

Layer	Conditions for two tree nodes being similar
L^t	$N_1.tid = N_2.tid$
L^l	$\|N_1.loc, N_2.loc\| < c\_loc$
L^d	$\|N_1.ts - N_2.ts\| + \|N_1.te - N_2.te\| < c\_tm$
L^a	$\|N_1.agl - N_2.agl\| < c\_agl$

Table 3: Methods to combine tree nodes.

Parameter	Calculation Method
tid	$N_1.tid$ or $N_2.tid$
vLink	$N_1.vLink \wedge N_2.vLink$
loc	$(N_1.loc + N_2.loc)/2$
ts, *te*	$\text{Min}\{N_1.ts, N_2.ts\}$, $\text{Max}\{N_1.te, N_2.te\}$
agl	$(N_1.agl + N_2.agl)/2$

(2) If a picture is uploaded to the data center, its container leaf node's parent node will be *virtually pruned*.

(3) If two leaf nodes are similar, then the older one (i.e., the stored picture's timestamp is earlier) will be abandoned. So the leaf node has no siblings.

(4) Child nodes of a virtually pruned tree node is useless, and they will be really pruned, that is, deleted.

(5) The virtually pruned node (i.e., *vLink* = False) has the strongest epidemic capability.

(6) Any node can be grafted to a PicTree only if it is different from other nodes of this PicTree.

(7) Nodes of two branches are merged from top to bottom like a zip until nodes on a certain layer cannot be merged.

According to the conditions and valuation methods shown in Tables 2 and 3, we will use an example to show the PicTree fusion based on fusion rules mentioned above as well as to explain two operations: *pruning* and *grafting*. As shown in Figure 4, the WN wn_2 and the WN wn_3 combine their PicTrees with the PicTree T_1 of wn_1, respectively, where those same-named tree nodes are similar. There are two kinds of basic fusion modes operating in this figure: pruning and grafting.

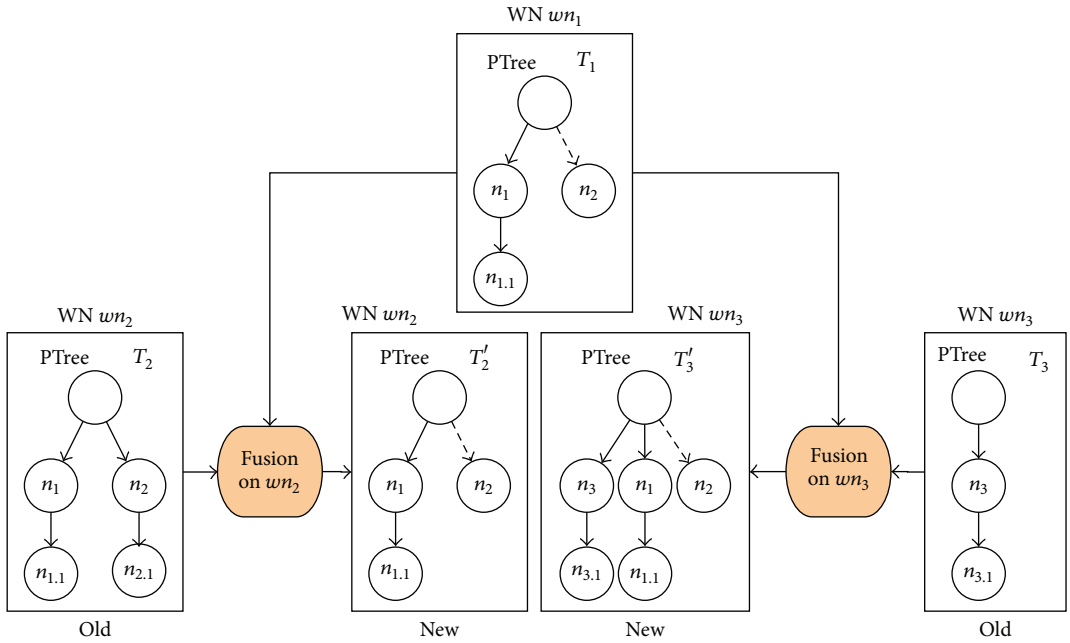


Figure 4: PicTree fusion of CooperSense. The dashed line means *vLink* = False. The left fusion is a pruning process while the right fusion is a grafting process.

(i) *Pruning*. The node n_2 on the PicTree T_2 of wn_2 (denoted by $T_2.n_2$) is infected to be virtually pruned because the node $T_1.n_2$ is virtually pruned and it has the highest epidemic capability. Then the leaf node $T_2.n_{2.1}$ will be really pruned because its parent node is virtually pruned. Pruning is used for useless epidemic termination.

(ii) *Grafting*. Nodes $T_1.n_1$, $T_1.n_2$ and their corresponding offspring nodes are grafted to the PicTree T_3 of the WN wn_3. Grafting is used to forward pictures.

Tree node fusion is a fundamental process in PicTree fusion. Next we will explain depth-first PicTree fusion algorithm.

4.1.2. Depth-First PicTree Fusion Algorithm. PicTree fusion uses the depth-first search. Given two PicTrees T_1 and T_2, the new PicTree T_1' is the result of merging T_1 and T_2. The main algorithm is shown in Algorithm 1. Functions used in this algorithm are *Combine*() and *CombineNode*(), which are shown in Algorithms 2 and 3, respectively. Function *similar*(,) in Algorithm 3 used to assess two nodes' similarity is calculated according to Table 2.

4.1.3. Fusion-Based PicTree Generation. PicTree fusion can be used not only for multi-WNs but also for single WN. Each WN has a PicTree, which is initially copied from the data center and further grows during the picture-taking and picture forwarding procedures.

Given a task (TSK) *tsk* and its task constraints (TSKC) *tc*, when a picture PIC p of this task is taken, a single-branch PicTree will be generated. Tree nodes of this new one-branch

Input: PicTree T_1 and PicTree T_2
Output: New PicTree T_1
$rn_1 = RootNodeOf(T_1)$;
$rn_2 = RootNodeOf(T_2)$;
$rn_1 = CombineNode(rn_1, rn_2)$;
Return rn_1.

Algorithm 1: Depth-first node search and PicTree fusion.

Input: Two tree nodes n_1, n_2
Output: New tree node n_1
$n_1.vLink = n_1.vLink \wedge n_2.vLink$;
Calculate other parameters of n_1 according to Table 3;
$n_1 = CombineNode\ (n_1, n_2)$;
Return n_1.

Algorithm 2: **Function** *Combine* (n_1, n_2).

PicTree are initiated according to Table 1. *vLink*s of all tree nodes are valued True, $L^d.ts = p.tm$, and $L^d.te = p.tm$. Other parameters of tree nodes are equal to attributes of *tsk*, *tc*, and p, such as $L^l.loc = p.loc$, $L^l.c\_loc = tc.c\_loc$. The PicTree will grow by merging the old PicTree and the new one-branch PicTree. If there are multitasks and more than two pictures, a complex multibranch PicTree will be generated through the PicTree fusion. As shown in Figure 5, the PicTree is generated with $\{p_1, p_2, p_3, p_4\}$. Because picture p_2 is similar to p_3, p_2 will be deleted and replaced by p_3.

Input: Two tree nodes n_1, n_2
Output: The new tree node n_1
  If n_1 is on the bottom layer L^i **Then**
    return n_1;
  End if
  $CN_1 = AllChildNodesOf\ (n_1)$;
  $CN_2 = AllChildNodesOf\ (n_2)$;
  For $\forall nn_2 \in CN_2$ **do**
    $haveCombined = False$;
    For $\forall nn_1 \in CN_1$ **do**
      If $similar(nn_1, nn_2) = True$ **Then**
        $nn_1 = Combine\ (nn_1, nn_2)$;
        $haveCombined = True$;
        Break;
      End if
    End for
    If $haveCombined = False$ **Then**
      nn_2 is grafted as a child node to the parent node of nn_1;
    End if
  End for
Return n_1.

Algorithm 3: **Function** *CombineNode* (n_1, n_2).

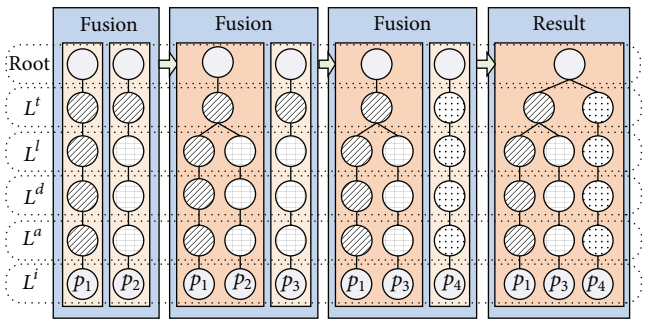


Figure 5: A PicTree is generated through three fusions from left to right with four pictures. Same colored nodes in the same layer are similar nodes.

4.2. Fusion-Based Picture Forwarding. In this section, how to utilize PicTree fusion to forward useful picture and terminate useless epidemic will be explained.

4.2.1. A PicTree Fusion Example. Both virtual/real grafting and virtual/real pruning are utilized for three purposes. (i) The picture file will be forwarded only if there is a real grafting and the picture is sent from the subbranch provider to the subbranch receiver. (ii) The accomplished task and copies of uploaded pictures will be eliminated through the virtual grafting and the virtual pruning. (iii) The storage space is saved through the real pruning.

Figure 4 basically explained the PicTree fusion, but how to utilize this fusion to forward pictures is still a question. Another PicTree fusion example was shown in Figure 6. wn_1 and wn_2 are two working WNs. Picture sets carried by them are $P^{wn_1} = \{p_1, p_2\}$ and $P^{wn_2} = \{p_3, p_4, p_5\}$. The WN wn_3 is a new WN; it downloads collection logs from the data center and enters into the observed area, so wn_3 took no pictures; that is, $P^{wn_3} = \{\}$. There are two forwarding procedures in Figure 6. (1) The WN wn_2 encounters wn_1. The PicTree fusion results are that wn_2 receives a picture p_2 from wn_1, and wn_1 receives two pictures p_4 and p_5 from wn_2. Pictures p_1 and p_3 are similar, so they are not forwarded. (2) The WN wn_2 encounters wn_3. Some branches of PicTree in wn_2 are virtually pruned and some are really pruned, which make wn_3 receive only one picture p_2.

4.2.2. Picture Forwarding. When two WNs encounter, they would like to receive pictures that they do not have. In order to select pictures, each WN costs a very small amount of traffic to receive the collection log from the other WN, and according to the collection log, the WN knows which picture needs be sent to and received from the other.

As shown in Figure 6, before these three WNs encounter, the effective pictures carried by wn_1, wn_2, and wn_3 are $\{p_1, p_2\}$, $\{p_3, p_4, p_5\}$, and $\{\}$, respectively. After wn_2 encounters wn_1 and wn_3, pictures carried by wn_1, wn_2, and wn_3 are $\{p_1, p_2, p_4, p_5\}$, $\{p_2, p_3, p_4, p_5\}$, and $\{p_2\}$, respectively. Pictures carried by each WN are different and should be uploaded.

4.2.3. Useless Epidemic Termination. The information of both accomplished and unaccomplished tasks is downloaded by WNs. In order to mark the accomplished task, a leaf tree node in the layer L^t corresponding to the task is created and its *vLink* is set to False. By the aid of the PicTree fusion, other WNs in the observed area will prune the branch mapped with this accomplished task. Furthermore, the WN will download the PicTree of uploaded pictures, whose *vLink*s of all tree nodes on the L^a layer are valued False. Here those virtually pruned nodes mean that the picture with the context

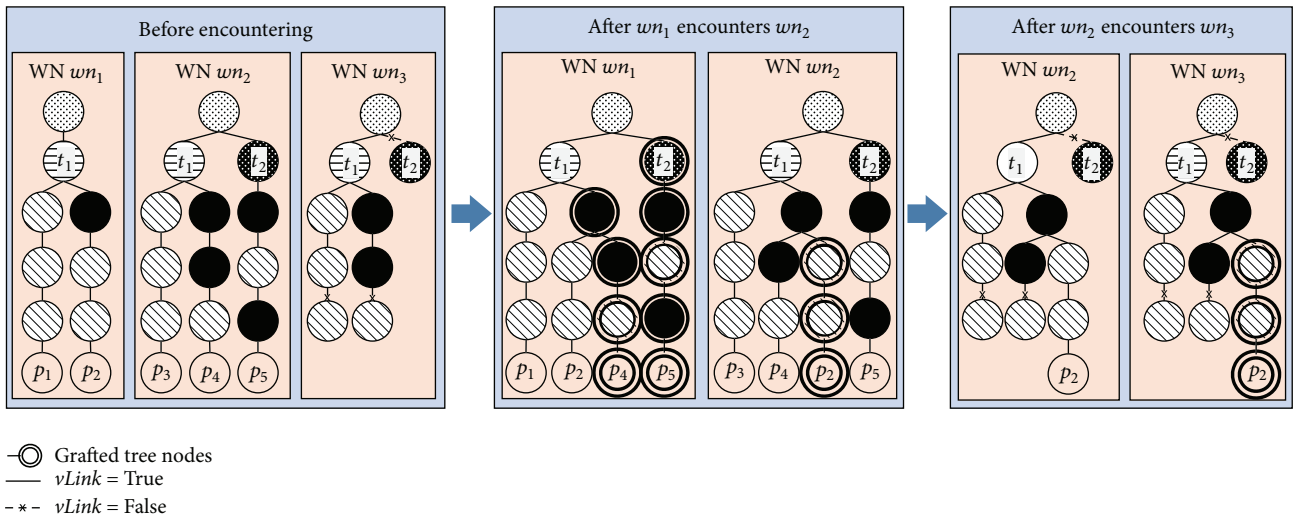


FIGURE 6: PicTree fusion of CooperSense. There are 3 WNs $\{wn_1, wn_2, wn_3\}$, 5 pictures $\{p_1, p_2, \ldots, p_5\}$, and 2 tasks $\{t_1, t_2\}$. wn_2 encounters wn_1 first and then encounters wn_3. Same colored nodes in the same layer are similar nodes.

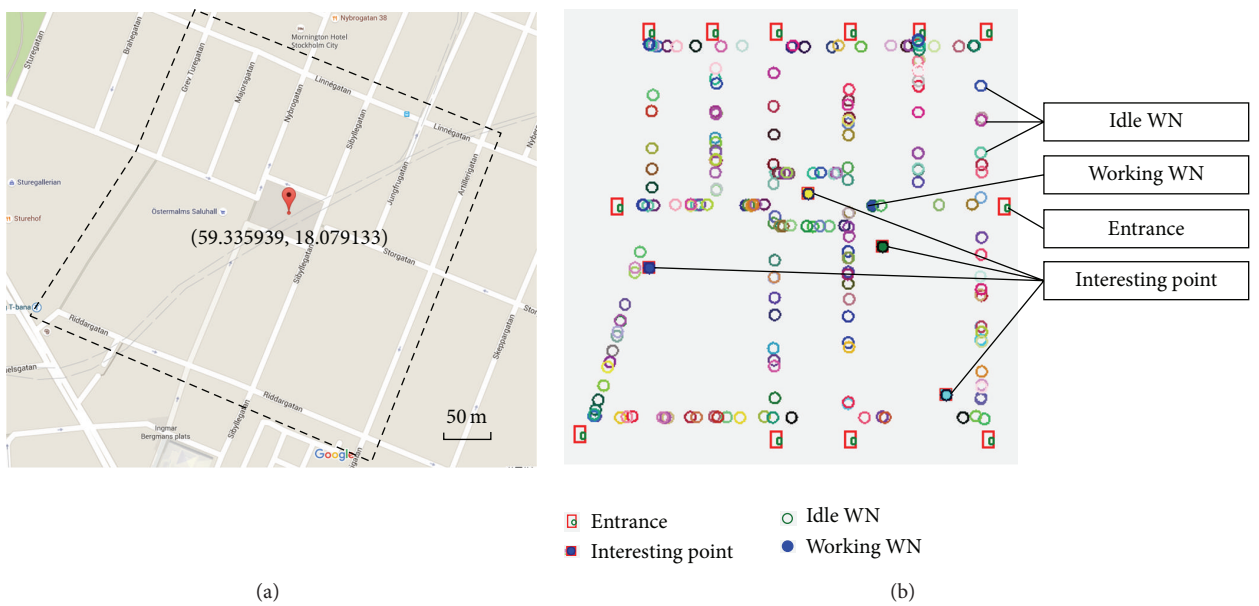


FIGURE 7: Simulation scenarios. (a) The electronic map of the observed area. It is Ostermalm area in downtown Stockholm. (b) Simulated moving WNs and interesting points. And medium-density WNs are simulated and shown in the map.

(denoted by tree nodes of this branch) has been uploaded to the data center already.

As shown in Figure 6, the task t_2 in the WN wn_3 is accomplished and the branch is virtually pruned, and after the WN wn_2 combines its PicTree with PicTree of wn_3, the task t_2 in the WN wn_2 is marked accomplished via a virtually pruned branch.

5. Performance Evaluation

In this section, we will evaluate the performance of CooperSense from different facets, including overhead, efficiency, traffic, and storage.

5.1. Simulation Setup. The Legion Studio [23] traces we use in this study are available at CRAWDAD [20], which are microsimulation of pedestrian mobility Contributed in the Web. They are simulated data set by the work in [13]. These traces are from a simulation of walkers in a part of downtown Stockholm in which several parameters are varied. Sparse and medium-density dataset are used here. The Ostermalm area consists of a grid of interconnected streets shown in Figure 7 and 4 MCP tasks are defined with 4 interesting points. Twelve entrances connect the observed area with the outside world. The active area of the outdoor scenario is 5,872 m^2.

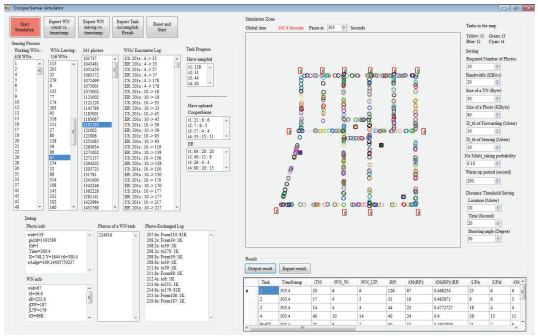


Figure 8: The interface of the simulator of CooperSense (CS: CooperSense; ER: Epidemic Routing). Medium-density WNs are used.

We started collecting data after a warm-up period because the system starts empty. After that we kept collecting observations until no WN forwarded pictures any more. The reason for that is a WN will not take and forward pictures when a WN learns that the task is accomplished from others through exchanging collection logs, and some WNs will still perform tasks when the data center has already got plenty of pictures for the task. The simulator of CooperSense is shown in Figure 8. In the simulation zone, the WN denoted by a color circle is moving according to the trajectory dataset of CRAWDAD [24].

In order to simulate the forwarding behavior of the WNs, we define four rules as follows:

(1) The data and task centers are connected with the outside world, so the data center can receive task requirements from the task center.

(2) If a WN is close to the interesting point (i.e., *whr* of a task) enough, then it will take pictures as a working WN. A WN might take multiple pictures from different positions for one task, and shooting angles will be computed based on this WN's trajectory with the range $[0, 2\pi)$, which is illustrated in Figure 7(b).

(3) When enough pictures have been uploaded to the data center, the corresponding task will be marked as accomplished, but the WN might still upload pictures.

(4) Once two WNs are close enough, only if the link duration time is larger than the transmission time, then data delivery can be finished.

5.2. Baseline and Metrics. ER (Epidemic Routing) [11] is a popular opportunistic forwarding approach, and we will compare CooperSense with it. In the simulation, each WN of ER has a set of pictures and a set of tasks. The flag of the accomplished task is set to False, all unexpired tasks are downloaded like CooperSense, and each WN forwards pictures and tasks' flags.

Different metrics are defined in this paper to evaluate the method, including traffic-saving, storage-saving, efficiency, effectiveness, and coverage. Detailed metrics are defined in the following sections along with the experimental results.

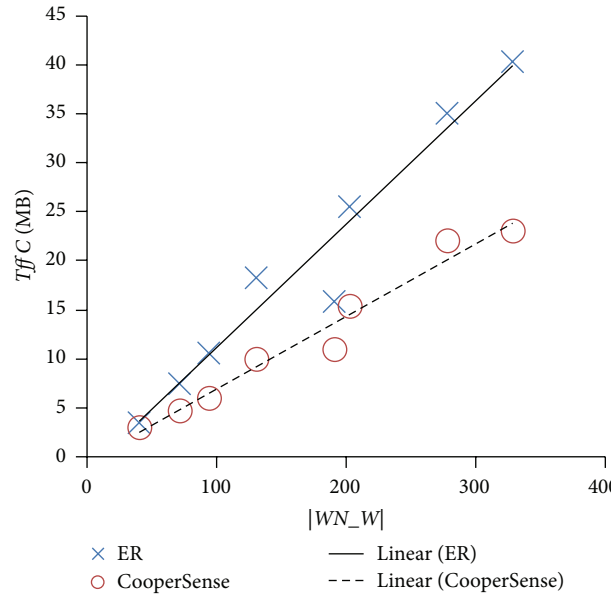


Figure 9: Experimental result of comparing the traffic-saving performance of CooperSense and ER. All tasks are accomplished.

5.3. Experimental Result

5.3.1. Energy-Saving Evaluation. Data transmission is the most energy-consuming process, so we use traffics of forwarding pictures (denoted by *TffC*) and ratio of extra uploaded pictures (denoted by *ExUpR*) to evaluate the energy-saving performance. *ExUpR* denotes the ratio of extra uploaded pictures calculated with (1), where *UPd* denotes the set of uploaded pictures, *UPd_op* denotes the optimal diversified subset [2] of *UPd* that contains valuable pictures, and *tsk* is a task TSK:

$$ExUpR = \frac{|UPd\_op| - tsk.ntr}{tsk.ntr}. \quad (1)$$

WN_W denotes the set of working WNs in the observed area. Because different WN enters and leaves uncertainly, *WN_W* at different time points are different. So $|WN\_W|$ in the experiment has an average value.

As shown in Figure 9, *TffC* of ER is much higher than that of CooperSense, especially when $|WN\_W|$ is high. This is due to the fact that more pictures will be forwarded when the density of *WN* increases and ER creates more useless forwarding.

ExUpR are utilized to evaluate the extra overhead. Epidemic termination is the way to decrease useless forwarding, which is evaluated by the metric *ExUpR*. As shown in Figure 10, most *ExUpR*s of both ER and CooperSense are close and less than 50%, but sometimes *ExUpR* of CooperSense is much lower than that of ER.

Through comparisons of *TffC* and *ExUpR* between ER and CooperSense, it is obvious that the latter can save more traffic overhead.

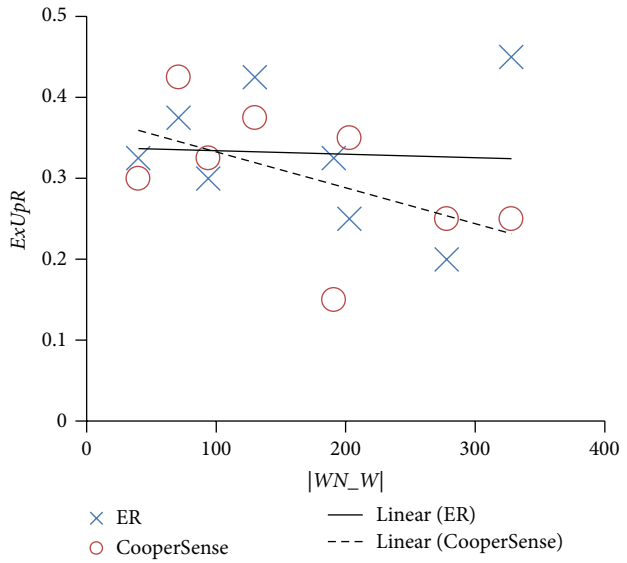


Figure 10: Experimental result of comparing the traffic-saving performance of CooperSense and ER. All tasks are accomplished.

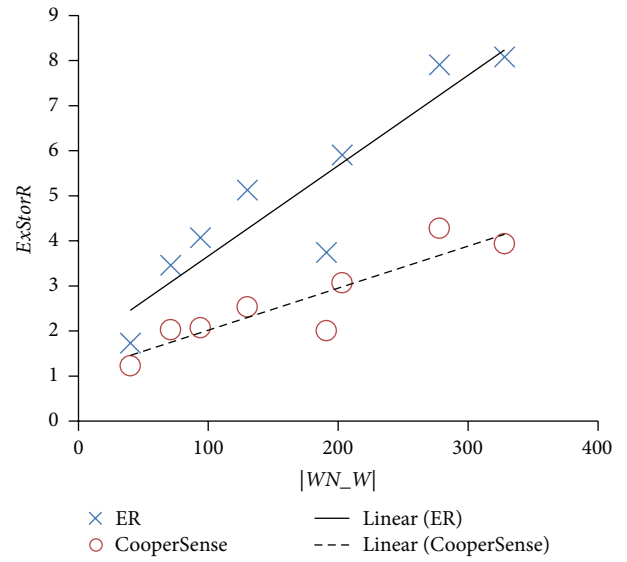


Figure 11: Experimental result of comparing the storage-saving performance of CooperSense and ER. All tasks are accomplished.

5.3.2. Storage-Saving Evaluation. *ExStorR* denotes extra storage overhead ratio calculated with (2), where *RP* refers to the set of raw pictures taken by cameras and *CP* denotes to the set of pictures (including copies) carried by all WNs:

$$ExStorR = \frac{|CP| - |RP|}{|RP|}. \quad (2)$$

As shown in Figure 11, *ExStorR* of ER is much higher than that of CooperSense. Therefore, CooperSense saves storage overhead more effectively. CooperSense uses PTree-fusion to stop forwarding redundant pictures, so the storage overhead and the traffic overhead are saved. Although ER can also use variables to mark already uploaded pictures and find duplicate pictures, for MCP, its traffic overhead is still higher than CooperSense's because WNs of ER must send complete PICs while WNs of CooperSense only need to send a certain part of PICs according to the PTree-fusion context.

5.3.3. Efficiency Evaluation. Delay of uploading pictures is used to evaluate the efficiency of opportunistic forwarding method. *AvgDelay* refers to the average delay of uploaded pictures calculated with (3), where *tArr* denotes the timestamp of the picture being uploaded to the data center. *TmC* denotes average time-consumption of a task:

$$AvgDelay = \frac{\sum_{p \in UPd} (p.tArr - p.tm)}{|UPd|}. \quad (3)$$

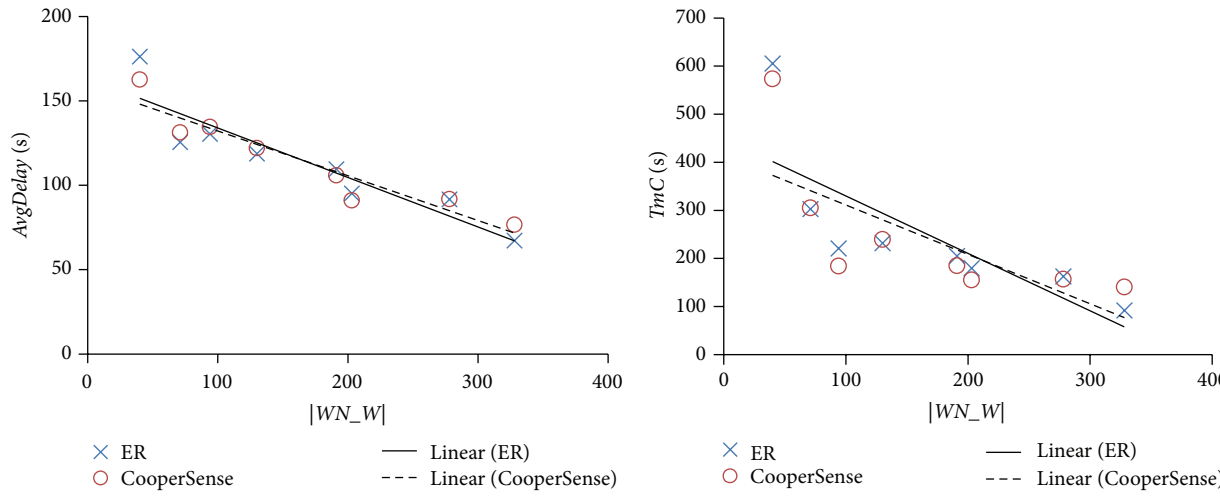
Efficiency evaluation is shown in Figure 12, where both *AvgDelay* and *TmC* of ER and CooperSense are close. Generally, ER is an efficient way to upload pictures, but the result shows that CooperSense provides a similar efficiency as ER can do. On the one hand, although WNs of CooperSense forward fewer pictures via the opportunistic network than those of ER do, CooperSense's efficiency for task performing is not influenced. On the other hand, both *AvgDelay* and *TmC* decline as $|WN\_W|$ increases, so the efficiency of opportunistic forwarding method cannot be improved by increasing the number of PIC copies but by involving more WNs. Hence the selective forwarding mode utilized by CooperSense consumes less resource on the promise of efficiently accomplishing tasks.

5.3.4. Effectiveness Evaluation. In order to evaluate the effectiveness of our method, we define metrics to indicate how much opportunity of uploading pictures our method can utilize. *OppUpR* denotes the ratio of opportunistically uploaded pictures to the data center calculated by (4), where the picture set *UPo* ($UPo \subseteq UPd$) consists of pictures that are not uploaded by their photo-takers:

$$OppUpR = \frac{|UPo|}{|UPd| - |UPo|}. \quad (4)$$

Effectiveness evaluation result is shown in Figure 13. *OppUpR* of them have no clear difference; therefore, CooperSense and ER have similar effectiveness of opportunistically uploading pictures. Additionally, but when $|WN\_W|$ increases, *OppUpR* of them increases because more people could opportunistically carry pictures.

5.3.5. Coverage Evaluation. Coverage evaluation is based on four metrics: $|UPa|$ is the number of those pictures that are carried out of the observed area, $|UPd|$ is the number of uploaded pictures, *DR* denotes the ratio of delivered pictures



(a) Average delay of uploading one picture

(b) Average time-consumption of a task

Figure 12: Experimental results of the efficiency of CooperSense and ER.

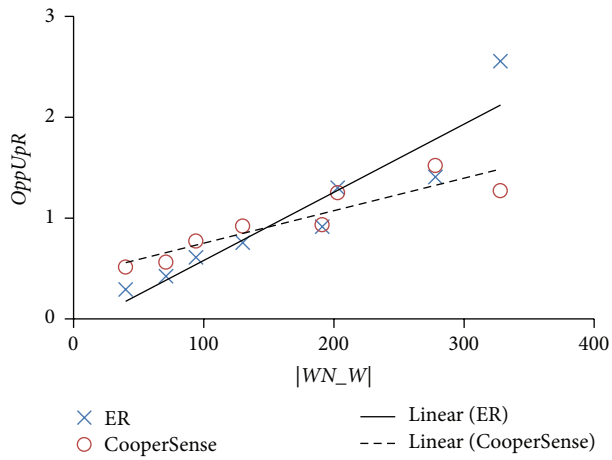


Figure 13: Comparison between ER and CooperSense on the ratio of opportunistically uploaded pictures.

calculated with (5), and *OpUpR* denotes the ratio of optimal uploaded pictures calculated by (6):

$$DR = \frac{|UPd|}{|RP|}, \tag{5}$$

$$OpUpR = \frac{|UPd\_op|}{|UPd|}. \tag{6}$$

The comparison results are shown in Figure 14. $|UPa|$ of ER are much larger than that of CooperSense and $|UPd|$ are similar, which proofs that CooperSense can save storage overhead comparing with ER. Additionally, since $|UPa|$ of CooperSense is less than that of ER but *OpUpR* of CooperSense is a little higher than that of ER, this means that *UPd* of CooperSense is more diverse than that of ER.

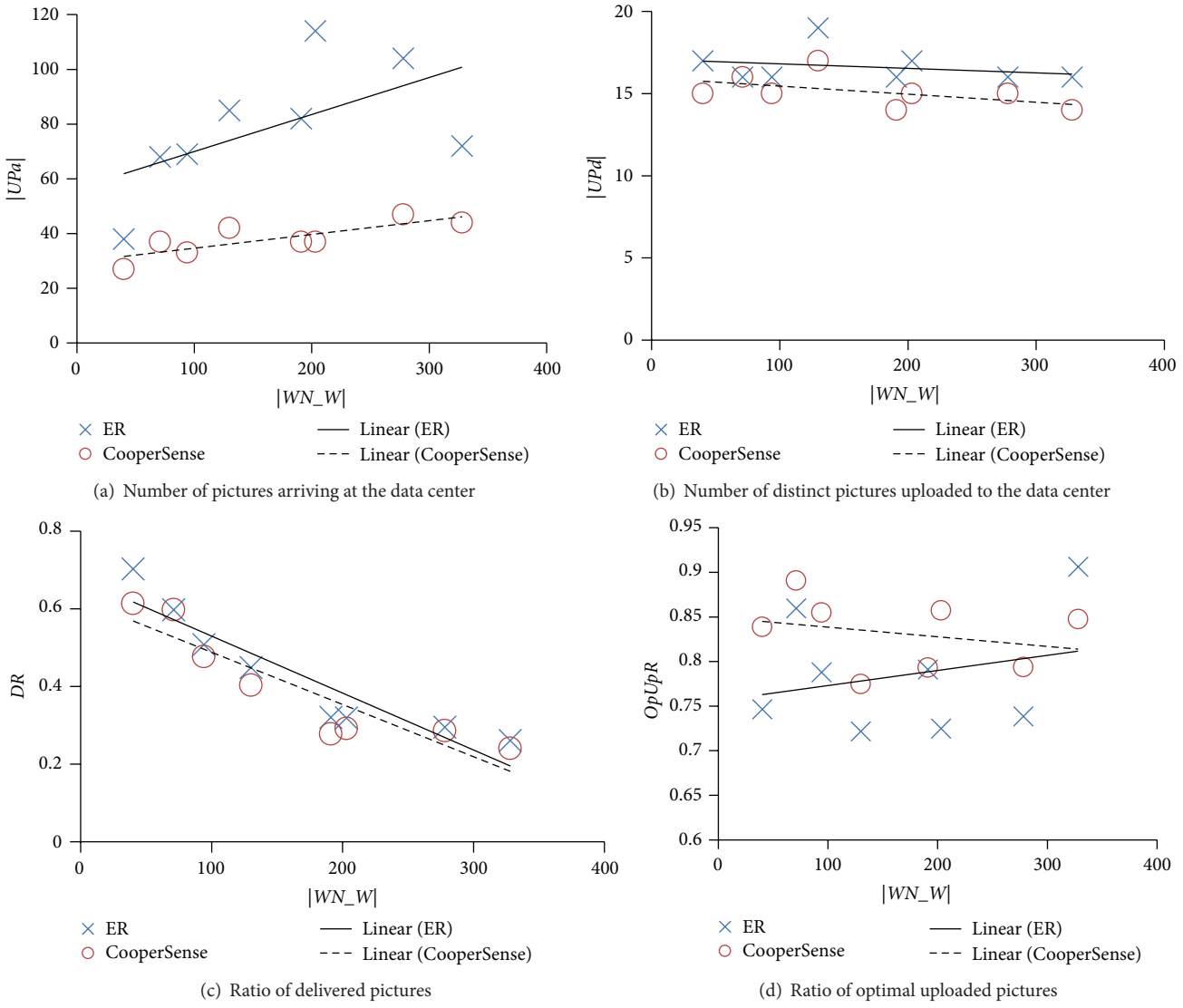
As experimental results show, CooperSense almost has the same picture collection capability of ER while it saves much traffic and storage overhead comparing to ER. The PTree-fusion method utilizes a tree structure to store picture collection logs and PTree-fusion is an efficient way to exchange different data among WNs for MCP in MON. Therefore, our method is flexible, which can selectively forward valuable pictures without complex computations.

5.4. Discussion. CooperSense forwards pictures through exchanging PicTree of encountered worker nodes, and there are still some issues that are not explained in this paper, some of which are listed below:

(1) The visual feature is not used during the PicTree fusion. In this paper, we do not include the visual feature of pictures to assess the similarity because the visual similarity measurement is highly related to the application. However, the image layer can be easily inserted when it is needed in real applications because of the flexibility of PicTree construction.

(2) Some works on opportunistic forwarding leverage the prediction of people's position to select the right people to carry and forward data, which is more adaptable for delay tolerant tasks. Since reducing delay is the goal of this scenario, we do not use this method in this paper, but it may be utilized in our future work.

(3) Although an emergency scenario is used to test our work, cooperative crowd sensing is also applicable in some crowed spots, such as in a thousand-attendee conference, a football match, or a singer show. Defining a task and its task constraint, using CooperSense, people can receive various pictures shared by others nearby who have better or different views without using the limited and narrow-bandwidth cellular communication.



(a) Number of pictures arriving at the data center

(b) Number of distinct pictures uploaded to the data center

(c) Ratio of delivered pictures

(d) Ratio of optimal uploaded pictures

Figure 14: Experimental result of comparing the performance of CooperSense and ER.

6. Conclusion and Future Work

Above all, CooperSense is a novel cooperative and selective data forwarding framework for MCP. In this paper, the Pic-Tree model and its fusion method were proposed for participant collaboration and selective data forwarding. Experiment results indicate that CooperSense has higher performance in terms of storage and network cost than ER. It is planned to be used in city-sense project for large-scale user study, and all the methods will be further improved accordingly.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was partially supported by the National Basic Research Program of China (no. 2015CB352400) and the National Natural Science Foundation of China (no. 61332005, 61373119, 61222209).

References

[1] B. Guo, Z. Wang, Z. Yu et al., "Mobile crowd sensing and computing: the review of an emerging human-powered sensing paradigm," *ACM Computing Surveys*, vol. 48, no. 1, article 7, 2015.

[2] H. Chen, B. Guo, S. Krishnaswamy, Z. Yu, and L. Chen, "Crowdpic: a multi-coverage picture collection framework for mobile crowd photographing," in *Proceedings of the UIC*, pp. 68–76, IEEE, 2015.

[3] S. Kim, C. Robson, T. Zimmerman, J. Pierce, and E. M. Haber, "Creek watch: pairing usefulness and usability for successful citizen science," in *Proceedings of the 29th Annual CHI Conference on Human Factors in Computing Systems (CHI '11)*, pp. 2125–2134, ACM, Vancouver, Canada, May 2011.

[4] B. Guo, H. Chen, Z. Yu, X. Xie, S. Huangfu, and D. Zhang, "FlierMeet: a mobile crowdsensing system for cross-space

public information reposting, tagging, and sharing," *IEEE Transactions on Mobile Computing*, vol. 14, no. 10, pp. 2020–2033, 2015.

[5] K. Tuite, N. Snavely, D.-Y. Hsiao, N. Tabing, and Z. Popović, "PhotoCity: training experts at large-scale image acquisition through a competitive game," in *Proceedings of the 29th Annual CHI Conference on Human Factors in Computing Systems (CHI '11)*, pp. 1383–1392, ACM, Vancouver, Canada, May 2011.

[6] Miit: Tianjin explosions have no serious impact on local communications network, http://en.people.cn/n/2015/0813/c98649-8935580.html.

[7] M. Y. S. Uddin, H. Wang, F. Saremi, G.-J. Qi, T. Abdelzaher, and T. Huang, "PhotoNet: a similarity-aware picture delivery service for situation awareness," in *Proceedings of the 32nd IEEE Real-Time Systems Symposium (RTSS '11)*, pp. 317–326, IEEE, Vienna, Austria, December 2011.

[8] A. Zaslavsky, P. P. Jayaraman, and S. Krishnaswamy, "ShareLikesCrowd: mobile analytics for participatory sensing and crowdsourcing applications," in *Proceedings of the IEEE 29th International Conference on Data Engineering Workshops (ICDEW '13)*, pp. 128–135, IEEE, Brisbane, Australia, April 2013.

[9] J. T. B. Fajardo, K. Yasumoto, and M. Ito, "Content-based data prioritization for fast disaster images collection in delay tolerant network," in *Proceedings of the 7th International Conference on Mobile Computing and Ubiquitous Networking (ICMU '14)*, pp. 147–152, Singapore, January 2014.

[10] M. Conti, S. Giordano, M. May, and A. Passarella, "From opportunistic networks to opportunistic computing," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 126–139, 2010.

[11] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Tech. Rep. CS-200006, Duke University, 2000.

[12] U. Weinsberg, Q. Li, N. Taft et al., "CARE: content aware redundancy elimination for challenged networks," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks (HotNets '12)*, pp. 127–132, ACM, Redmond, Wash, USA, October 2012.

[13] B. Guo, Z. Yu, D. Zhang, and X. Zhou, "Cross-community sensing and mining," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 144–152, 2014.

[14] H. Ma, D. Zhao, and P. Yuan, "Opportunities in mobile crowd sensing," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 29–35, 2014.

[15] Y. Jiang, X. Xu, P. Terlecky, T. Abdelzaher, A. Bar-Noy, and R. Govindan, "Mediascope: selective on-demand media retrieval from mobile devices," in *Proceedings of the 12th International Conference on Information Processing in Sensor Networks (IPSN '13)*, pp. 289–300, ACM, Philadelphia, Pa, USA, 2013.

[16] D. Zhao, H. Ma, S. Tang, and X.-Y. Li, "COUPON: a cooperative framework for building sensing maps in mobile opportunistic networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 2, pp. 392–402, 2015.

[17] Z. Yu, H. Xu, Z. Yang, and B. Guo, "Personalized travel package with multi-point-of-interest recommendation based on crowdsourced user footprints," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 151–158, 2016.

[18] M. Soltani, M. Hempel, and H. Sharif, "Utilization of convex optimization for data fusion-driven sensor management in WSNs," in *Proceedings of the International Wireless Communications and Mobile Computing Conference (IWCMC '15)*, pp. 1224–1229, IEEE, Dubrovnik, Croatia, August 2015.

[19] N. T. Baranasuriya, S. L. Gilbert, C. Newport, and J. Rao, "Aggregation in smartphone sensor networks," in *Proceedings of the 9th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '14)*, pp. 101–110, IEEE, Marina Del Rey, Calif, USA, May 2014.

[20] K.-Y. Chow, K.-S. Lui, and E. Y. Lam, "Efficient on-demand image transmission in visual sensor networks," *Eurasip Journal on Advances in Signal Processing*, vol. 2007, Article ID 95076, 1 page, 2007.

[21] F. Luqman, F.-T. Sun, H.-T. Cheng, S. Buthpitiya, and M. Griss, "Prioritizing data in emergency response based on context, message content and role," in *Proceedings of the 1st International Conference on Wireless Technologies for Humanitarian Relief (ACWR '11)*, pp. 63–69, ACM, December 2011.

[22] Y. Wang, W. Hu, Y. Wu, and G. Cao, "Smartphoto: a resource-aware crowdsourcing approach for image sensing with smartphones," in *Proceedings of the 15th ACM International Symposium on Mobile ad hoc Networking and Computing (MobiHoc '14)*, pp. 113–122, ACM, Philadelphia, Pa, USA, August 2014.

[23] Legion studio, http://www.legion.com/.

[24] S. T. Kouyoumdjieva, Ó. R. Helgason, and G. Karlsson, *CRAWDAD Data Set kth/Walkers (v.2014-05-05)*, 2014, http://crawdad.org/kth/walkers/.

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2016, Article ID 1693437, 7 pages
http://dx.doi.org/10.1155/2016/1693437

Research Article

Urban Impedance Computing Based on Check-In Records

Zhiyong Yu,[1] Yuzhong Chen,[1] Songpan Zheng,[1] Yao Shen,[2] Zhiwen Yu,[3] and Jordan Pascual[4]

[1]*College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China*
[2]*Space Syntax Lab, Bartlett School of Architecture, University College London, London NW1 2BX, UK*
[3]*College of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China*
[4]*Department of Computer Science, University of Oviedo, 33024 Gijón, Spain*

Correspondence should be addressed to Zhiyong Yu; yuzhiy@gmail.com

Received 25 December 2015; Accepted 27 March 2016

Academic Editor: Erik Mannens

Urban impedance is an important consideration in assessments of transportation and land-use systems. This work leverages check-in records obtained from mobile social networks to build a fine-grained but inexpensive urban impedance model. Check-in records and road networks are collected and used to calculate and adjust the various parameters of the model, including path length, number and angle of turns, number and direction of junctions, and population density. Check-in records can filter functional locations and supply the time factor, thereby providing excellent advantages over traditional models that do not employ this data type. The proposed model is more accurate than traditional impedance models, as verified by experiments using Sina Weibo data in Tianjin City.

1. Introduction

The rapid progress of urbanization has modernized many people's lives but also engendered several important issues, such as traffic congestion, high housing prices, and low neighborhood cohesion. Urban accessibility [1] is a fundamental measure of the benefits of urban life and answers how many destinations can be accessed within a given time. It is widely used to evaluate transportation and land-use systems. Increasing accessibility is essential for urban planning. Impedance, which is defined as the cost between the origin and destination locations, is an important indicator of urban accessibility.

Traditional impedance models require a large amount of supporting data, including the geographical distribution of public facilities, traffic volume of different time periods, and population in the range of study [2]. These data are difficult to collect from many cities of developing countries. As the use of mobile social networks has recently grown immensely, a large amount of user mobility data has been generated. Mobile social networks allow a user to "check in" at POIs (points of interest), which corresponds to an online record describing his/her current physical location, and share this record with his/her friends. This new data source can be leveraged to build a fine-grained but inexpensive urban impedance model. The present paper explores this idea and presents the following main contributions:

(1) Selection and preparation of appropriate data sources for the impedance model: we extract city blocks from road networks, collect POIs and check-ins from mobile social networks, and match them correspondingly.

(2) Proposal and implementation of the impedance model: we use check-in records to filter POIs and supply the time factor and then calculate distances, turns, junctions, and population parameters after path planning.

(3) Visualization of impedance and its parameters in a real city.

2. Related Work

In recent years, researchers have studied a number of urban accessibility models. Handy and Niemeier [3] proposed a utility model based on the discrete-choice model. By assessing a value for each candidate destination in the region, individuals will be more likely to visit destinations with larger utilities. Wachs and Kumagai [4] proposed a cumulative opportunity model that considers the number of destinations in a certain range; unfortunately, this model ignores differences in opportunity points. Hansen [5] proposed a gravity model that considers the distance between centers and opportunity points and could also be used to analyze the market potential [6]. Accessibility is heavily influenced by the selection of the impedance function. Common impedance functions include exponential functions, Gaussian functions, and negative power functions. The choice of impedance function mainly depends on the applications of the model and the characteristics of the data used. Since their functions and parameters are simplified, these models tend to ignore the complexity of city roads and, therefore, show limited accuracy.

In research on impedance functions, the United States BPR (Bureau of Public Roads) developed [7] a speed-flow model through regression analysis. To overcome the shortcomings of BPR model resulting in large deviations when the traffic volume reaches its peak, Spiess [8] and Wang et al. [9] improved the impedance function. According to queuing theory, Davidson [10] proposed a progressive impedance function that amends the parameters describing the relationship between travel time and distance. Wang et al. [11] improved the impedance model based on traffic flow by considering travel costs, time, hub locations, and other factors. Despite their benefits, however, as mentioned earlier, these traditional impedance models require a significant amount of supporting data.

Another research field related to urban accessibility is urban computing or smart city. Urban computing [12] aims to tackle urbanization issues through a process involving acquisition, integration, and analysis of data generated from various sources in urban spaces, such as sensors, devices, vehicles, buildings, and humans, to reflect traffic flows, human mobility, geographical information, and so forth. For example, Zheng et al. [13] described the underlying problems in Beijing's transportation network using the hypothesis that the connection between two regions cannot effectively support the traffic traveling between them, thereby resulting in a large volume, low speed, and high detour ratio. Fu et al. [14] predicted the rankings of residential real estate in a city at a future time according to their potential values inferred from a variety of data sources, such as human mobility data and urban geography, currently observed around the real estate. Chen et al. [15] leveraged a combination of location-based social networks and taxi GPS digital footprints to achieve personalized, interactive, and traffic-aware trip planning. Yu et al. [16] recommended personalized travel package with multiple points of interest based on crowd sourced user footprints. These works are mostly application-oriented and can be embedded into and improved by our model.

3. Urban Impedance Model

3.1. Preliminary

Definition 1 (block center). Road networks divide the city area into polygonal blocks [17, 18]. Let $V = \{v_1, v_2, \ldots, v_{|V|}\}$ indicate the centers of these blocks and let $|V|$ indicate the number of blocks available. People and facilities in a block usually share the same transport environment; thus, block centers may be considered representative locations when analyzing the impedance of a block (we can compute the impedance for all locations, but doing so is time-consuming and unnecessary). The position of the center of a block can be easily obtained given its vertices and edges.

Definition 2 (function density). This term considers how many functional locations (i.e., POIs) can be reached from an object location (e.g., the center of a block) given an accessibility radius. The positions of POIs are indicated by longitude and latitude. The function density is displayed in formula (1). Consider

$$\mathrm{Den}_i = \sum_{j=1}^{k} \partial_{ij}, \quad \partial_{ij} = \begin{cases} 1, & d\left(f_j, v_i\right) \le r \\ 0, & d\left(f_j, v_i\right) > r, \end{cases} \tag{1}$$

where $d(f_j, v_i)$ denotes the road network distance between a POI f_j and the center of a block v_i, r is the accessibility radius defined in terms of road network distance (because people travel along road networks and care about transport distances rather than absolute distances), and k is the total number of POIs.

POIs, such as shops, restaurants, and tourist attractions, have business hours, for example, opening from 8:00 a.m. to 8:00 p.m. Thus, a time factor must be considered when calculating their impedance. Out of their business hours, these POIs are regarded as absent. We set a time slot lasting for two hours and classify days into weekdays or weekends. As a result, a total of 24 time intervals should be considered. For each block center, the function density is adjusted according to formula (2). Consider

$$\begin{aligned} \mathrm{Den}_{i,t} &= \sum_{j=1}^{k} \partial_{ij} \beta_{jt}, \\ \partial_{ij} &= \begin{cases} 1, & d\left(f_j, v_i\right) \le r \\ 0, & d\left(f_j, v_i\right) > r \end{cases}, \quad \beta_{jt} = \begin{cases} 1, & t \in \mathrm{time}\left(f_j\right) \\ 0, & t \notin \mathrm{time}\left(f_j\right), \end{cases} \end{aligned} \tag{2}$$

where $\mathrm{time}(f_j)$ denotes the business hours of a POI f_j, t is a specific time slot, and the other variables are identical to those in formula (1). POIs with ≥10 check-ins are taken into account in this work.

Definition 3 (path). A path is a part of road network that starts from an origin location and ends at a destination location. A path is described by five elements $\langle O, D, \mathscr{T}, \mathscr{J}, \mathscr{L} \rangle$: origin ($O$), destination ($D$), turns ($\mathscr{T}$), junctions ($\mathscr{J}$), and line segments among them ($\mathscr{L}$). Consider $\mathscr{T} = \{\theta_1, \theta_2, \ldots, \theta_{|\mathscr{T}|}\}$,

where $|\mathcal{T}|$ is the number of turns and θ_i is the angle of each turn. Consider $\mathcal{J} = \{c_1, c_2, \ldots, c_{|\mathcal{J}|}\}$, where $|\mathcal{J}|$ is the number of junctions and c_i is the direction to take upon arriving at a junction, that is, turning right, turning left, or going straight. Consider $\mathcal{L} = \{d_1, d_2, \ldots, d_{|\mathcal{T}\cup\mathcal{J}|+1}\}$, where $|\mathcal{T} \cup \mathcal{J}|$ is the total number of turns and junctions and d_i is the length of each line segment.

We adopt the Baidu map traffic routing API [19] to obtain paths between block centers and POIs.

3.2. Modeling. The overall impedance situation of a city can be understood by the impedances of its block centers, which are denoted by $\{m_1, m_2, \ldots, m_{|V|}\}$. The larger the value of m_i, the larger the impedance of this block center.

Drawing lessons from existing models, we take both path parameters and population parameters into consideration. We also use check-in records to adjust these parameters. Check-in records can help our modeling procedure in two ways. First, check-in records can filter functional locations. Only locations that own many/frequent check-ins are meaningful for impedance computing. Second, check-in records can supply the time factor. A location presents different impedances at different times. Time not only influences the function density (as shown in formula (2)) but also impacts the real-time population (which will be described in Section 3.3.4).

Three parameters are considered for paths, that is, path length (L), number and angle of turns (T), and number and direction of junctions (J), and one parameter is considered for population, that is, the population density (P). The impedance of the block center v_i at the time interval t is

$$m_{i,t} = \alpha\overline{L_{i,t}} + \beta\overline{T_{i,t}} + \gamma\overline{J_{i,t}} + \delta\overline{P_{i,t}}, \quad (3)$$

where α, β, γ, and δ are the weights of different parameters.

3.3. Parameter Calculation

3.3.1. Path Length. Path length (i.e., road network distance), the most important component of the impedance model, refers to the sum of lengths of line segments in a path. The average of all path lengths from a block center to all accessible POIs is taken as an initial parameter of our model:

$$L_{i,j} = \sum_{k=1}^{|\mathcal{T}\cup\mathcal{J}|+1} d_k, \quad \text{where } d_k \in \mathcal{L} \text{ of path from } v_i \text{ to } f_j, \quad (4)$$

$$\overline{L_{i,t}} = \frac{\sum_{j=1}^{\mathrm{Den}_{i,t}} L_{i,j}}{\mathrm{Den}_{i,t}}.$$

We use the CartoDB map tool [20] for visualization. Figure 1 shows the average path lengths of Tianjin's block centers (at a particular time interval). Blocks with darker colors represent blocks with longer average path lengths. A detailed description of the data will be provided in Section 4.

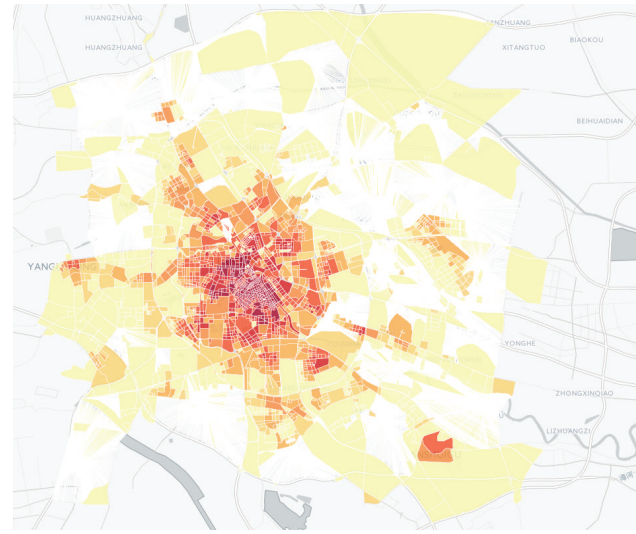


Figure 1: Average path lengths of Tianjin's block centers.

3.3.2. Number and Angle of Turns. The path from the block center to a POI may contain turns. Obviously, more and sharper turns will slow down traffic. Thus, the number and angle of turns in a path must be taken into account in the urban impedance model.

Baidu map traffic routing indicates a very large number of slight turns. Because modeling all of these turns is time-consuming and unnecessary, angles less than 10 degrees are neglected. Traditional impedance models tend to ignore the effect of the segment between two turns. In practice, however, if only a short distance must be traveled from one turn to another, the whole passing time will be greater. The average of path *turns* (short for the number and angle of turns, the same below) from a block center to all accessible POIs is taken as the second parameter of our model. Consider

$$T_{i,j} = \sum_{k=1}^{|\mathcal{T}|} \ln\left(\frac{\tan(\theta_k/4)}{d_k} + 1\right), \quad d_k \in \mathcal{L},\ \theta_k \in \mathcal{T} \text{ of path from } v_i \text{ to } f_j,\ \theta_k \geq 10^\circ, \quad (5)$$

$$\overline{T_{\iota,t}} = \frac{\sum_{j=1}^{\mathrm{Den}_{i,t}} T_{i,j}}{\mathrm{Den}_{i,t}},$$

where d_k is the length of the line segment between the k-1th turn and the kth turn. The function tan() can show angle differences properly, and the function ln() can restrict the value range. Figure 2 shows the average path turns of Tianjin's block centers.

3.3.3. Number and Direction of Junctions. Junctions on a path increase the passing time because people must wait for traffic lights to change and pedestrians from other directions to cross roads. The more the junctions on a path, the larger its impedance. Different directions at a junction present different waiting and passing times. Turning left, for example, requires more cost than going straight and then turning right.

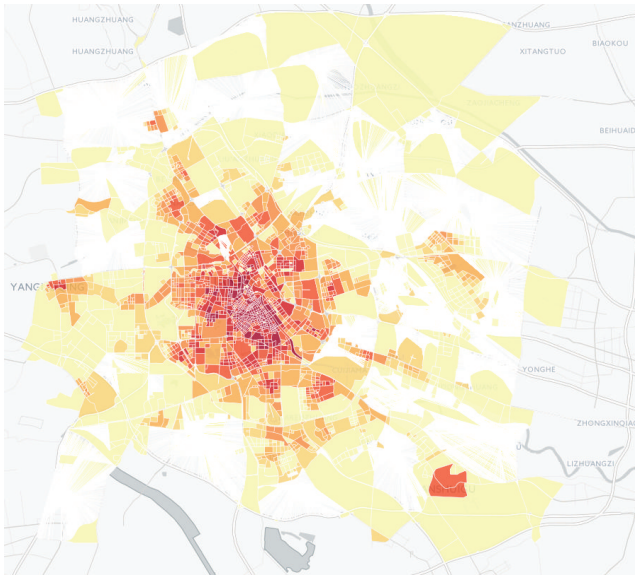


Figure 2: Average path turns of Tianjin's block centers.

No impedance function is widely used to consider junctions in existing works. In our model, we construct one using logical rules. The average of path *junctions* from block center to all accessible POIs is taken as the third parameter of our model. Consider

$$J_{i,j} = \sum_{k=1}^{|\mathcal{J}|} \frac{c_k}{d_k},$$

$$c_k = \begin{cases} 1, & \text{turn right} \\ 2, & \text{go staight}, \ c_k \in \mathcal{J} \text{ of path from } v_i \text{ to } f_j, \\ 3, & \text{turn left} \end{cases} \quad (6)$$

$$\overline{J_{i,t}} = \frac{\sum_{j=1}^{\text{Den}_{i,t}} J_{i,j}}{\text{Den}_{i,t}},$$

where d_k is the length of the line segment between the k-1th junction and the kth junction. Figure 3 shows the average of path junctions of Tianjin's block centers.

3.3.4. Population Density. The population density directly affects the traffic situation in a block; thus, its impact on impedance cannot be ignored. Different times of day show different population densities. Although some traditional impedance models also consider population, real-time population distribution data are difficult to obtain and their precisions are not able to meet the demands of accurate calculations. Users' check-in records can properly represent the population distribution and can be easily obtained from mobile social networks. As different area sizes present different abilities to relieve traffic congestion, the population density of an approximately circular area determined by

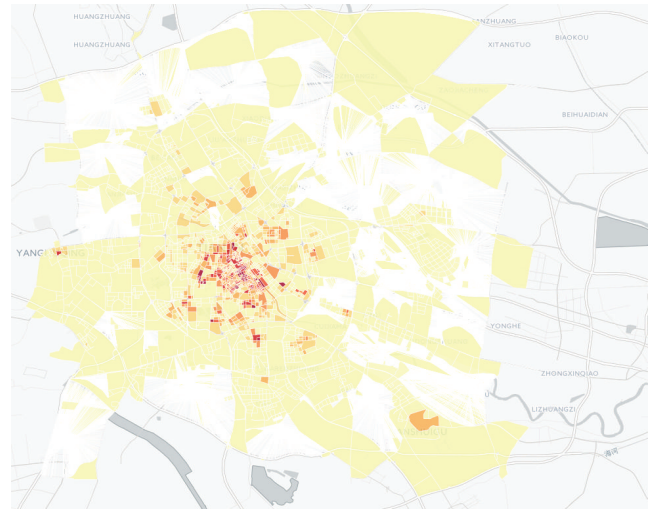


Figure 3: Average path junctions of Tianjin's block centers.

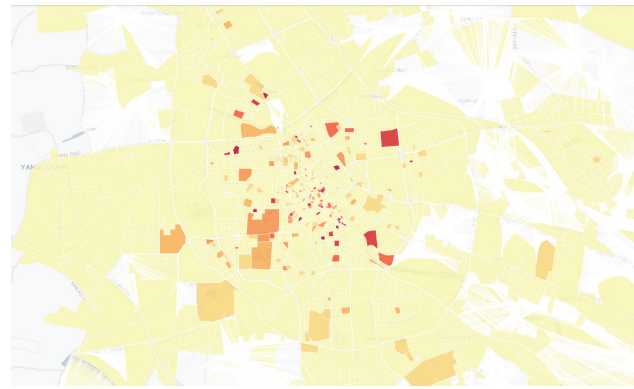


Figure 4: Population density of Tianjin's block centers (8:00 a.m.–10:00 a.m., weekend).

the block center and the accessibility radius is taken as the fourth parameter of our model:

$$\overline{P_{i,t}} = \frac{\sum_{j=1}^{\text{Den}_{i,t}} \text{check}_{j,t}}{s_{i,r}}, \quad (7)$$

where $\text{check}_{j,t}$ is the number of check-ins at POI f_j and $s_{i,r}$ is the size of an approximately circular area with center v_i and accessibility radius r (simply calculated as πr^2 in this work). Figure 4 shows the population density of Tianjin's block centers (from 8:00 a.m. to 10:00 a.m. on a weekend).

4. Experiments

The experimental data include road networks from the national data center and POIs and check-in records from Sina Weibo [21]. We use review records of POIs from Dianping [22] for cross-checking, as well as two traditional impedance models (the potential and utility models) for comparison. Real-time road conditions from the Baidu map are considered the ground truth.

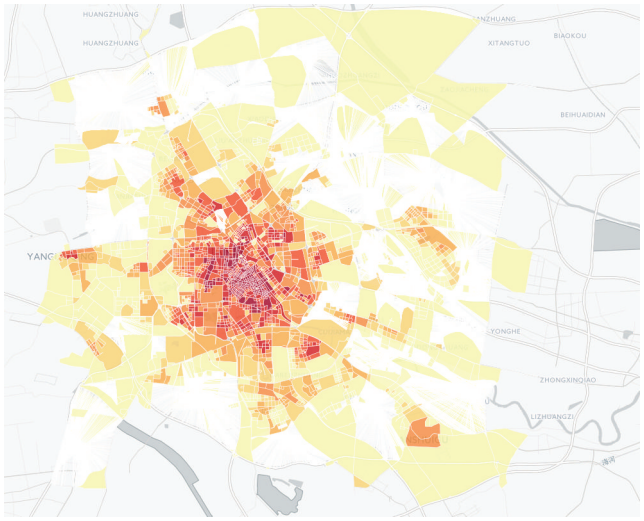


Figure 5: Impedance of Tianjin's block centers (8:00 a.m.–10:00 a.m., weekend based on Weibo).

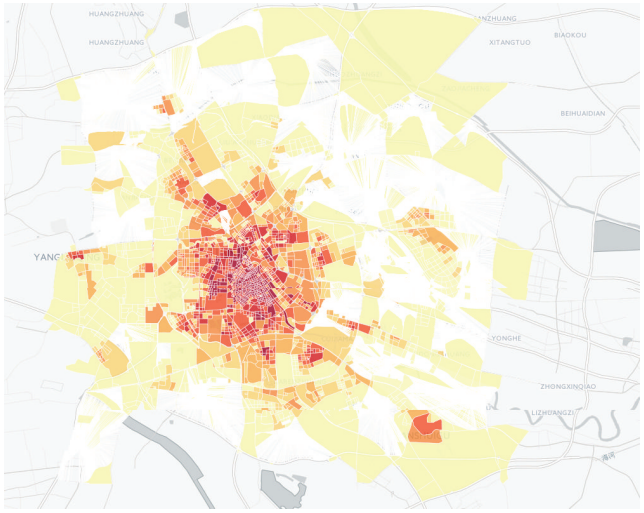


Figure 6: Impedance of Tianjin's block centers (8:00 a.m.–10:00 a.m., weekend based on Dianping).

This paper describes the case of Tianjin (one of four municipalities in China) as an example. Road networks divide the city area into 2,754 blocks. A total of 90,731 POIs and 533,006 check-in records were collected in 2014. The accessibility radius was set to 1,000 meters in terms of road network distance, and 357,681 paths should be calculated.

4.1. Cross-Checking with Different Data. First, we present the results according to Weibo data. Figure 5 shows the impedance of Tianjin's block centers (8:00 a.m.–10:00 a.m., weekend).

Using the same impedance model (see formula (3)), we compute Tianjin's impedance according to Dianping data and display the results in Figure 6. Dianping data include 45,767 POIs and 311,794 review records (regarded as check-in records). For the same time interval, the results of the Dianping data are basically identical to those of the Weibo data. This finding shows that our model works robustly and steadily with different data sources.

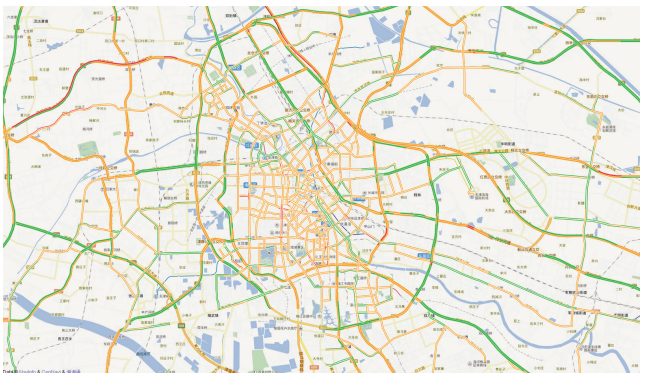


Figure 7: Real time road conditions in Tianjin at 9:00 a.m. on a Saturday.

4.2. Comparison with Other Models. The utility and potential models are two common impedance models.

The utility model is based on the discrete-choice model. The basic idea of this model is that different facilities present different utilities to people. For example, a supermarket is more likely to be visited than a car shop. For each destination within the accessibility radius, a utility value is assigned to the origin. The greater the utility is, the greater expectation this destination will be visited. The impedance function of this model is described as a logarithmic sum: $m_i = \ln^{-1} \sum_{j=1}^{\mathrm{Den}_i} \exp(u_{i,j})$, where $u_{i,j}$ is the utility value of v_i assigned by f_j. We use the number of ODs (a pair of check-in records left by the same user in a day) as the utility value.

The potential model is based on Newton's gravity model. Here, two factors are considered: the attractiveness of the destination, usually indicated by the population or facility density at the destination, and the distance decay. A typical expression of the potential model is $m_i = \sum_{j=1}^{\mathrm{Den}_i} (\exp(2L_{i,j})/\mathrm{check}_j)$, where f_j's check-in number check_j indicates the attractiveness of f_j.

Real-time road conditions are a good reflection of impedance. We take real-time road conditions from the Baidu map as a reference to evaluate all three impedance models. Figure 7 shows the real-time road conditions in Tianjin at 9:00 a.m. on a Saturday. Here, green lines indicate clear roads, yellow lines indicate slow roads, and red lines indicate roads with traffic congestion. To achieve a more intuitive contrast among the results of the impedance models, we project the road conditions to the corresponding blocks. Figure 8 shows the projected result of road conditions in Tianjin (average from 8:00 a.m. to 10:00 a.m. on a weekend). A comparison of Figures 5 and 8 indicates nearly coincident results.

We now present a quantitative comparison to demonstrate which model yields results most similar to those of the reference (i.e., projected results of real road conditions). The Minkowski distance, a proper similarity criterion, is written

Table 1: Similarity between each model and road condition (weekends).

Models	Minkowski distance		
	8:00–10:00	16:00–18:00	20:00–22:00
Proposed model	0.604	0.753	0.787
Utility model	5.607	6.387	6.516
Potential model	2.315	2.972	2.744

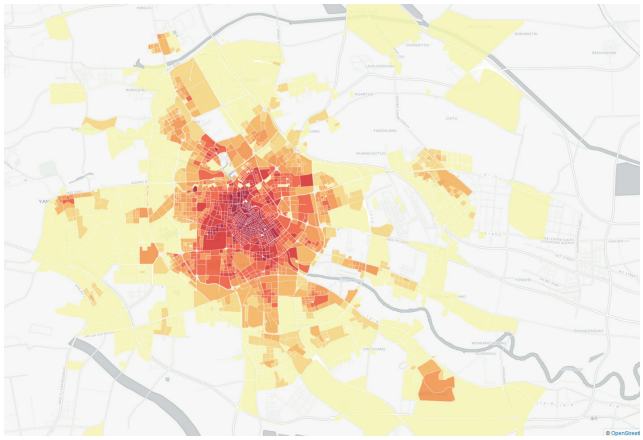


Figure 8: Projected result of road condition in Tianjin (8:00 a.m.–10:00 a.m., weekend).

as $(\sum_{i=1}^{|V|} |m_i - y_i|^p)^{1/p}$, where m_i is the impedance of the block center v_i computed by each model, y_i is the road condition of block v_i, and p is set to 2. Here, a smaller distance means more similarity. Notice that the time factor is considered for all models. Table 1 shows the similarity between each model considered in this work and the actual road conditions.

Table 1 reveals that our proposed model yields results with the most similarity to actual road conditions. The utility model performs the poorest among the models studied, and the potential model shows moderate accuracy. The poor performance of the traditional models may be explained as follows: The utility model only considers the attractiveness of the destination and ignores the path between origin and destination, while the potential model takes the distance from the origin to the destination into consideration but ignores other parameters, such as turns and junctions. Our proposed model combines the parameters path length, number and angle of turns, number and direction of junctions, and population density, thereby accurately depicting the city's impedance at different time periods.

5. Conclusions

Urban impedance is an important indicator to consider in assessments of transportation and land-use systems. However, traditional impedance models require extensive data collection, which is costly, but yield only coarse-grained results. The present work leverages check-in records obtained from mobile social networks to build a fine-grained but inexpensive urban impedance model. We use check-in records to adjust the path and population parameters of the model. Check-in records can filter functional locations and supply the time factor, which not only influences the function density but also impacts the real-time population of an area. Several experiments confirmed that our proposed model yields more accurate results than traditional impedance models.

In future research, we aim to take multiple-path factors into consideration and employ more types of data to obtain an improved impedance model.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This study is partially supported by the National Natural Science Foundation of China (no. 61300103).

References

[1] M. Batty, "Accessibility: in search of a unified theory," *Environment and Planning B: Planning and Design*, vol. 36, no. 2, pp. 191–194, 2009.

[2] N. He, N. Liu, and S. Zhao, "A study of road traffic impedance base on BPR function," *Journal of Nanjing Institude of Technology: Natural Science Edition*, vol. 11, no. 1, pp. 6–11, 2013.

[3] S. L. Handy and D. A. Niemeier, "Measuring accessibility: an exploration of issues and alternatives," *Environment and Planning A*, vol. 29, no. 7, pp. 1175–1194, 1997.

[4] M. Wachs and T. G. Kumagai, "Physical accessibility as a social indicator," *Socio-Economic Planning Sciences*, vol. 7, no. 5, pp. 437–456, 1973.

[5] W. G. Hansen, "How accessibility shapes land use," *Journal of the American Institute of Planners*, vol. 25, no. 2, pp. 73–76, 1959.

[6] D. Keeble, P. L. Owens, and C. Thompson, "Regional accessibility and economic potential in the European community," *Regional Studies*, vol. 16, no. 6, pp. 419–432, 1982.

[7] "Highway capacity manual," *Transportation Research Board*, vol. 12, no. 7, pp. 690–696, 1987.

[8] H. Spiess, "Conical volume-delay functions," *Transportation Science*, vol. 24, no. 2, pp. 153–158, 1990.

[9] W. Wang, M. Jing, and D. Liu, "Routing with capacity limitation and multi-paths based on LOGIT model and BPR impedance functions," *Journal of Highway and Transportation Research and Development*, vol. S1, pp. 81–85, 2012.

[10] K. B. Davidson, "The theoretical basis of a flow-travel time relationship for use in transportation planning," *Anstralian Road Research*, vol. 8, no. 1, pp. 32–35, 1978.

[11] Y. Wang, W. Zhou, and L. Lu, "Theory and application study of the road traffic impedance function," *Journal of Highway and Transportation Research and Development*, vol. 21, no. 9, pp. 82–85, 2004.

[12] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: concepts, methodologies, and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 3, article 38, 55 pages, 2014.

[13] Y. Zheng, Y. Liu, J. Yuan, and X. Xie, "Urban computing with taxicabs," in *Proceedings of the 13th International Conference on*

Ubiquitous Computing (UbiComp '11), pp. 89–98, ACM, Beijing, China, September 2011.

[14] Y. Fu, H. Xiong, Y. Ge, Z. Yao, Y. Zheng, and Z.-H. Zhou, "Exploiting geographic dependencies for real estate appraisal: a mutual perspective of ranking and clustering," in *Proceedings of the 20th SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*, pp. 1047–1056, ACM, 2014.

[15] C. Chen, D. Zhang, B. Guo, X. Ma, G. Pan, and Z. Wu, "TripPlanner: personalized trip planning leveraging heterogeneous crowdsourced digital footprints," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1259–1273, 2015.

[16] Z. Yu, H. Xu, Z. Yang, and B. Guo, "Personalized travel package with multi-point-of-interest recommendation based on crowdsourced user footprints," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 151–158, 2016.

[17] J. Yuan, Y. Zheng, and X. Xie, "Segmentation of urban areas using road networks," MSR-TR-2012-65, 2012.

[18] J. Yuan, Y. Zheng, and X. Xie, "Discovering regions of different functions in a city using human mobility and POIs," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*, pp. 186–194, ACM, Beijing, China, August 2012.

[19] Baidu Map API, http://lbsyun.baidu.com.

[20] CartoDB, http://www.cartodb.com.

[21] Sina Weibo, http://weibo.com.

[22] Dianping, http://www.dianping.com.

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2016, Article ID 6958710, 16 pages
http://dx.doi.org/10.1155/2016/6958710

Research Article

A Context-Driven Worker Selection Framework for Crowd-Sensing

Jiangtao Wang,[1,2] Yasha Wang,[1,3] Sumi Helal,[4] and Daqing Zhang[1,2]

[1]*Key Laboratory of High Confidence Software Technologies, Ministry of Education, Beijing 100871, China*
[2]*School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China*
[3]*National Engineering Research Center of Software Engineering, Peking University, Beijing 100871, China*
[4]*Computer and Information Science and Engineering Department, University of Florida, Gainesville, FL 116120, USA*

Correspondence should be addressed to Yasha Wang; wangys@sei.pku.edu.cn

Received 1 September 2015; Accepted 27 December 2015

Academic Editor: Diego López-de-Ipiña

Worker selection for many crowd-sensing tasks must consider various complex contexts to ensure high quality of data. Existing platforms and frameworks take only specific contexts into account to demonstrate motivating scenarios but do not provide general context models or frameworks in support of crowd-sensing at large. This paper proposes a novel worker selection framework, named WSelector, to more precisely select appropriate workers by taking various contexts into account. To achieve this goal, it first provides programming time support to help task creator define constraints. Then its runtime system adopts a two-phase process to select workers who are not only qualified but also more likely to undertake a crowd-sensing task. In the first phase, it selects workers who satisfy predefined constraints. In the second phase, by leveraging the worker's past participation history, it further selects those who are more likely to undertake a crowd-sensing task based on a case-based reasoning algorithm. We demonstrate the expressiveness of the framework by implementing multiple crowd-sensing tasks and evaluate the effectiveness of the case-based reasoning algorithm for willingness-based selection by using a questionnaire-generated dataset. Results show that our case-based reasoning algorithm outperforms the currently practiced baseline method.

1. Introduction

Recent years have seen rapid improvements in the capabilities of mobile phones, such as processing power, embedded sensors, storage capacities, and network data rates. These technology advances coupled with the sheer number of user-companioned mobile phones enable a new and fast-growing sensing paradigm, which is referred to as crowd-sensing. Crowd-sensing is a capability by which application developers can create tasks and recruit smartphone users to provide sensor data to be used towards a specific goal. In this paper, developers who create the crowd-sensing task are referred to as task creators, while mobile users who fulfill the task by contributing data are referred to as workers.

To support crowd-sensing at large, many mediation platforms and frameworks have been proposed recently to connect the task and worker, which can be divided into two modes. The first is pull mode. For many existing mediation platforms, it is the worker's responsibility to search for the task he/she wants to undertake. In these platforms, the task creators post tasks on the platform by describing what should be done, defining the layout of the user interface, and specifying the reward. Then workers actively login to the platform and search for tasks that they are qualified for and interested in by providing some keywords. Then a worker will get a list of tasks and decides which one to complete. Typical systems adopting the pull mode include Amazon Mechanical Turk [1], Medusa [2], CrowdDB [3], CrowdSearch [4], and TurKit [5]. The second is push mode, in which the mediation platform or framework must be able to select appropriate workers automatically based on certain constraints. Typical systems or frameworks using the push mode include [6], PRISM [7], and Anonysense [8]. With the popularity of crowd-sensing paradigm, the number of both tasks and workers has been increasing rapidly. As a result, both the pull mode and push mode connection

between task and worker are important, since they either help workers find their preferred tasks or deliver a certain task to the most appropriate workers. The pull/push modes are analogous to the search-based and recommendation-based approaches, respectively, utilized for dealing with the information overload problem on the Web.

There are several research works focusing on the worker selecting for crowd-sensing. However, they have the following limitations.

First, many platforms do not provide general support for managing various and complex contexts. This is a significant deficit given that one of the biggest challenges for worker selection is the ability to utilize complex and generalized contexts in many cases. In this paper, factors that need to be considered in identifying appropriate workers are supported by a variety of contexts to be utilized by the crowd-sensing worker selection process. Existing mediation platforms or frameworks also take contexts into account to some extent. However, they only consider specific or sample contexts to support their own motivating scenarios but do not provide general context models or frameworks in support of crowd-sensing at large. For example, [6] points out that people's availability in terms of space and time, transportation mode, and the coverage must meet certain requirements in the selection phase. PRISM [7] takes the device's sensing capabilities and worker's location as the contexts. Anonysense [8] takes time, worker's location, and privacy setup into account. The authors in [9] regard the location, battery level, and coverage as contexts. In [10, 11], the budget and coverage are the contexts for selection.

Second, the selection is not precise enough, because existing platforms exploit contexts only in terms of the requirements defined by the task creator without considering other worker-required contexts that could highly decide whether the worker would accept the task or not. The task creator is either a skilled software developer or people with very basic programming language skills. According to their programming capabilities, different levels of language support are provided to help them define the contexts for worker selection. For example, [2] provides XML-based language for nonprofessional programmers, while PRISM [7] provides higher level programming support for professional programmers. No matter what level of programming support is provided, existing systems select the workers based on the constraints predefined merely by the task creator. However, this is not precise enough if too many workers meet the constraints. With the increasing popularity of crowd-sensing, such imprecise task pushing may overwhelm the workers given the large number of recommended tasks. In fact, workers decide whether to accept and actually undertake a recommended task based on many other contexts, for example, whether the reward is attractive enough, whether the worker is interested in the task domain, and if accomplishing the task may not cause too much privacy violation. These contexts are not well exploited in existing systems.

To overcome abovementioned limitations, this paper proposes a novel worker selection framework, named WSelector, to select appropriate workers by taking various contexts into account. To achieve this goal, WSelector first provides programming time support, which is based on context modeling technique, to help task creators define all constraints for worker selection. Then a two-phase selection process is adopted at runtime to identify workers who not only are qualified but also would be more willing to undertake a crowd-sensing task. In the first phase, it selects workers satisfying predefined constraints. In the second phase, by leveraging the worker's past participation history, it uses a case-based algorithm to further select workers who are more likely to accept to undertake the task. WSelector is not a mediation platform and does not handle many of the complex issues found in many of the existing platforms. However, it is intended to be integrated into existing or future mediation platforms to better support their worker selection features.

The main contributions of this paper are as follows:

(1) We propose a core context model to semantically express the general context for worker selection in crowd-sensing systems and to provide a mechanism for task creators to utilize context modeling in their applications.

(2) We propose a two-phase worker selection framework to identify workers who not only are qualified but are more likely to be willing to undertake a crowd-sensing task by taking various worker-side contexts into account.

(3) We demonstrate the expressiveness of the framework based on various crowd-sensing tasks and evaluate the effectiveness of a case-based reasoning algorithm based on a dataset collected from an online questionnaire.

2. Related Work

In this section, we review the related literature from two perspectives. The first is the worker selection capabilities of related platforms/frameworks for crowd-sensing, which is the main problem we are addressing in this paper. The second is the ontology-based context modeling, which is related to a key technique we use as a solution to the problem in our framework.

2.1. Worker Selection Framework. Many crowd-sensing mediation platforms, including Amazon Mechanical Turk [1], Medusa [2], CrowdDB [3], CrowdSearch [4], TurKit [5], and mCrowd [12], adopt the pull mode to connect task and worker. In these platforms, workers search the task and the platform does not need to select workers. Compared with these frameworks, ours can identify appropriate workers based on various contexts information and push tasks to them.

There are many other frameworks that are based on the push mode. In this mode, the mediation platform must be able to identify appropriate workers automatically based on certain factors. References [11, 13] develop a selection framework to enable organizers to identify well-suited participants for data collections based on geographic and temporal availability, transportation mode, and the coverage. PRISM [7]

Table 1: Worker selection of typical mediation platforms or framework: a comparison.

Platforms/frameworks	Comparison aspect		
	Push/pull	Consideration for contexts	Whether to consider worker-side contexts
Amazon Mechanical Turk [1]	Pull	No context	No
Medusa [2]	Pull	No context	No
CrowdDB [3]	Pull	No context	No
CrowdSearch [4]	Pull	No context	No
TurKit [5]	Pull	No context	No
mCrowd [12]	Pull	No context	No
[6, 11]	Push	Geographic and temporal availability, transportation mode, and the coverage	No
PRISM [7]	Push	Device's sensing capabilities and worker's location	No
Anonysense [8]	Push	Time, worker's location, and privacy setup	No
[9]	Push	Device's location, battery level, and the overall spatial coverage	No
CrowdRecruiter [10]	Push	Incentive payments and coverage constraint	No
Crowdlab [14]	Push	Location and battery budget	Yes
[15]	Push	Location	No
[16]	Push	Energy consumption and data quality	No
[17]	Push	data quality requirements, location, and budget constraints	No
WSelector	Push	Providing general context model	Yes

takes the device's sensing capabilities and worker's location as the constraints to select appropriate workers. Anonysense [8] takes time, worker's location, and privacy setup into account when identifying workers. Reference [9] proposes an assignment policy to identify suitable workers based on their device's location, battery level, and the overall spatial coverage. CrowdRecruiter [10] aims at minimizing incentive payments by selecting a small number of participants while still satisfying probabilistic coverage constraint. Crowdlab [14] schedules the task based on location and battery resource budget. Reference [15] takes location as the only context for worker selection. A QoI-Aware energy-efficient worker selection framework has recently been proposed [16], which considers the QoI requirements, the energy consumption index, and the estimation of the collected amount of data. The literature [17] selects workers based on the data quality requirements, location, and budget constraints. Although these works also take many contexts into account for worker selection, they only consider demonstrative contexts to support their motivating scenarios but do not provide general context models in support of crowd-sensing at large. Our paper proposes a core context model for worker selection, in which the general concepts for crowd-sensing are defined and characterized. All the demonstrative contexts in above frameworks can be modeled by extending our core context model. Besides, worker selection in the most of existing frameworks is merely based on the constraint defined by the task creator, while WSelector takes the worker-side contexts into account.

A comparison of the platforms or frameworks in terms of worker selection is summarized in Table 1.

2.2. Ontology-Based Context Modeling. Ontology-based context modeling techniques are widely used in pervasive computing systems, especially for modeling heterogeneous contexts in smart pervasive spaces. One of the first approaches of modeling the context with ontologies has been proposed by [18], which analyzed psychological studies on the difference between recall and recognition of several issues in combination with contextual information. Another approach has been proposed as the Aspect-Scale-Context Information (ASC) model [19]. Ontologies provide a uniform way for specifying the model's core concepts as well as an arbitrary amount of subconcepts and facts, altogether enabling contextual knowledge sharing and reuse [20]. The model has been implemented by applying selected ontology languages. These implementations build up the core of a nonmonolithic Context Ontology Language (CoOL), which is supplemented by integration elements such as scheme extensions for Web Services and others [21, 22]. The CONON context modeling approach [23, 24] created an upper ontology which captures general features of basic contextual entities and a collection of domain-specific ontologies and their features in each subdomain. Above works inspire our work to some extent, especially the idea of building domain-specific ontology based on an upper-level ontology proposed by [23, 24]. However, to the best of our knowledge, our paper is a

first work using ontology-based context modeling in crowd-sensing.

There are also other literatures inspiring our work, in that they also use the crowd-sourced way for ontology-based knowledge modeling. Reference [25] presented a hybrid metamodel that combines features from key-value, markup, object oriented, and ontology-based context modeling approaches. The architecture is also introduced to allow the dynamic collaborative extension and crowd-sourced convergence of context models. In [26], the authors presented different concepts to create context models, including a collaborative one. In [27], the authors proposed consensus-building mechanisms for collaborative ontology building, which is based on offline discussions. Reference [28] presented an entirely online-based process for ontology building and convergence that is implemented on an extended Wiki.

3. Framework Design Overview

3.1. Challenges. The goal of our framework is to perform a precise worker selection for crowd-sensing. It should be able to identify workers who not only are qualified but also willing to undertake a specific task. In order to achieve the goals, at least the following challenges exist.

First, it is difficult to model various contexts for worker selection. A key challenge for worker selection is that in many cases complex and various contexts must be considered, and they are all factors that need to be considered for selecting appropriate workers. To the best of our knowledge, though context modeling is a widely used technique in mobile and pervasive systems, there are no existing context models for characterizing the concepts and their relationships in crowd-sensing applications.

Second, it is difficult for the task creator to define all contexts reasonably or properly. A naive idea for enabling the worker selection is to require the task creator to define all the contexts which can be realized by task specification languages such as two-level predicate in [7], AnonyTL [8], and Medscript [2]. Although these languages are useful in defining the contexts to some extent, there are limitations in many scenarios. It is not at all clear if certain combinations of contexts may be unnecessarily too restrictive, or to the contrary, adequate to achieve successful selection. Task creator may not be thoughtful enough to define all needed contexts that some workers are likely to be missed and the task is likely to be pushed to inappropriate workers. Sometimes they may define a context that is unnecessarily too restrictive, thus excluding many appropriate workers.

Third, selecting workers who are willing to undertake a task is not easy. On the one hand, different contexts could have different influence in deciding whether the worker is willing to undertake a task. It is very difficult for task creator to predict or learn these differences. On the other hand, even the same context may have different impact on different workers, which is subjective and varies from one worker to another. For example, privacy preserving is more important than earning money for some workers, but it may be the opposite case for others.

3.2. Insights. The design of our framework is based on the following insights.

3.2.1. Insight 1: Context Classification. Context is an abstract concept. According to the abovementioned definition, it can be any factor that needs to be considered in the crowd-sensing worker selection. However, these contexts are not semantically at the same level and can be divided into the following categories.

Constraints. Constraints are the minimum and basic requirements that can be set on the workers (e.g., spatial constraint, temporal constraint, and sensing capability constraint). They are the objective requirements from the task itself.

Worker-Side Factors. They are subjective factors considered when the worker decides whether to undertake a recommended task, for example, whether the reward (incentive) is attractive, whether the task falls into his interested domain, whether his privacy is well protected, and whether the workload is too heavy.

Task Properties. Task properties characterize basic information of a crowd-sensing task (e.g., title, description, reward, keywords, and assignment). They are commonly key-value pairs whose value can be assigned in the mediation platform by either the form-like user interface or the task specification language.

3.2.2. Insight 2: The Opportunity of Leveraging Worker's Participation History. When a crowd-sensing task is pushed to a worker, it may be undertaken or declined, which is referred to as the *outcome* in this paper. Over a period of time, the outcome, together with the corresponding worker-side factors and task properties, constitutes a worker's participation history.

Workers' participation history contains knowledge about how the worker-side factors and task properties affect the workers' decisions. Therefore, one basic idea in this paper is to leverage the participation history to improve the precision of future worker selection, making the task recommend to those who are more likely to accept it. Since WSelector must be integrated into an existing mediation platform (e.g., Amazon Mechanical Turk) in practical usage, we assume that our framework can access the participation history from the mediation platform.

3.3. Design Overview. With abovementioned challenges and insights, we design a novel worker selection framework, named WSelector. The basic idea for the design of WSelector is as follows: (1) since it is difficult for the task creator to define all contexts reasonably or properly, WSelector first provides a programming framework to help task creators define constraints for the worker selection; (2) WSelector adopts a two-phase process at runtime to select workers who are not only qualified but also more likely to undertake a crowd-sensing task. We demonstrate the system design in terms of programming time and runtime support as follows.

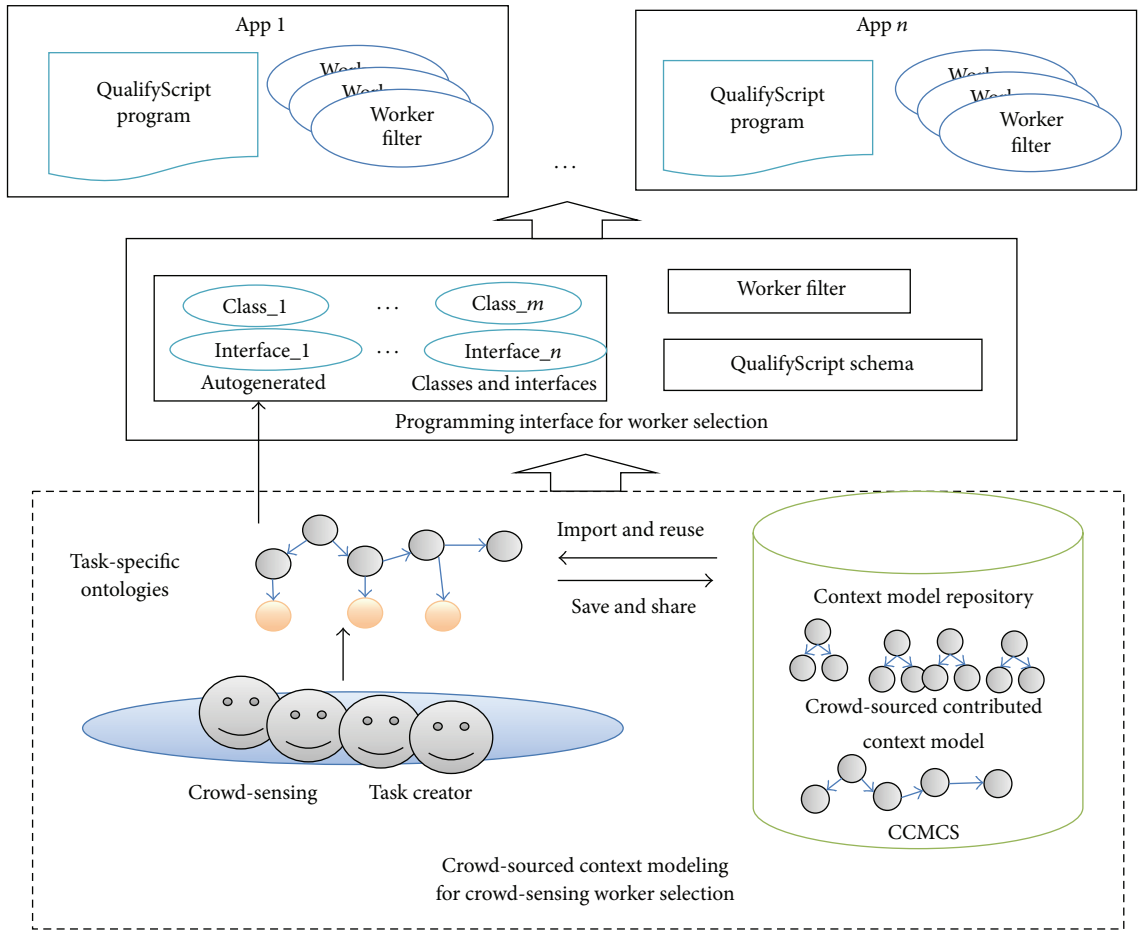


Figure 1: Programming time support.

3.3.1. Programming Time Support. A context-driven programming framework (see Figure 1) is used to assist the task creator define task properties and constraints.

First, this programming framework is based on a core context model and crowd-sourced context modeling mechanism. (1) We propose an ontology-based context model, named CCMCS (core context model for crowd-sensing), to define the most fundamental concepts for crowd-sensing. The objectives of the CCMCS include modeling a set of upper-level concepts and providing flexible extensibility to add specific concepts for different crowd-sensing tasks. In realistic crowd-sensing tasks, there are other task-dependent concepts needed to be characterized, which share common concepts that are modeled in CCMCS and differ significantly in detailed features. Our framework enables the task creators to build their task-specific ontologies by extending CCMCS. (2) To enable the reuse of context models, we also provide a crowd-sourced mechanism. When a task creator wants to create a context model, he can either extend the CCMCS or import existing models that are established and saved in the context model repository by other task creators. Since the creation and management of ontologies are a mature technology and there are many existing tools, our framework directly exploits the Protégé (http://protege.stanford.edu/), a free open-source Java tool, to support the creation and management of ontologies.

Second, the framework provides interfaces for task creators to define constraints for worker selection. (1) Our framework provides an abstract interface WorkerFilter(), and the task creator can create task-specific filters by extending this interface. To make the filter creation more efficient, the framework generates some JAVA components automatically, which can be directly imported to realizing task-specific filters. Task-specific ontology classes and their data properties defined by the task creator are automatically transformed into corresponding Java components including classes and interfaces. Our framework adopts the approach in [29] to implement the transformation from ontologies to Java components. (2) We propose QualifyScript, an XML-based language, to define the references of all the worker filters. The worker selection reference is defined between the <filter> and </filter> tags and refers to an executable filter implemented by the task creator. QualifyScript also provides other predefined tags, including title, description, reward, and assignment, to define task properties. It also allows task creator to define new tags.

Box 1 shows a QualifyScript program for the air quality report application.

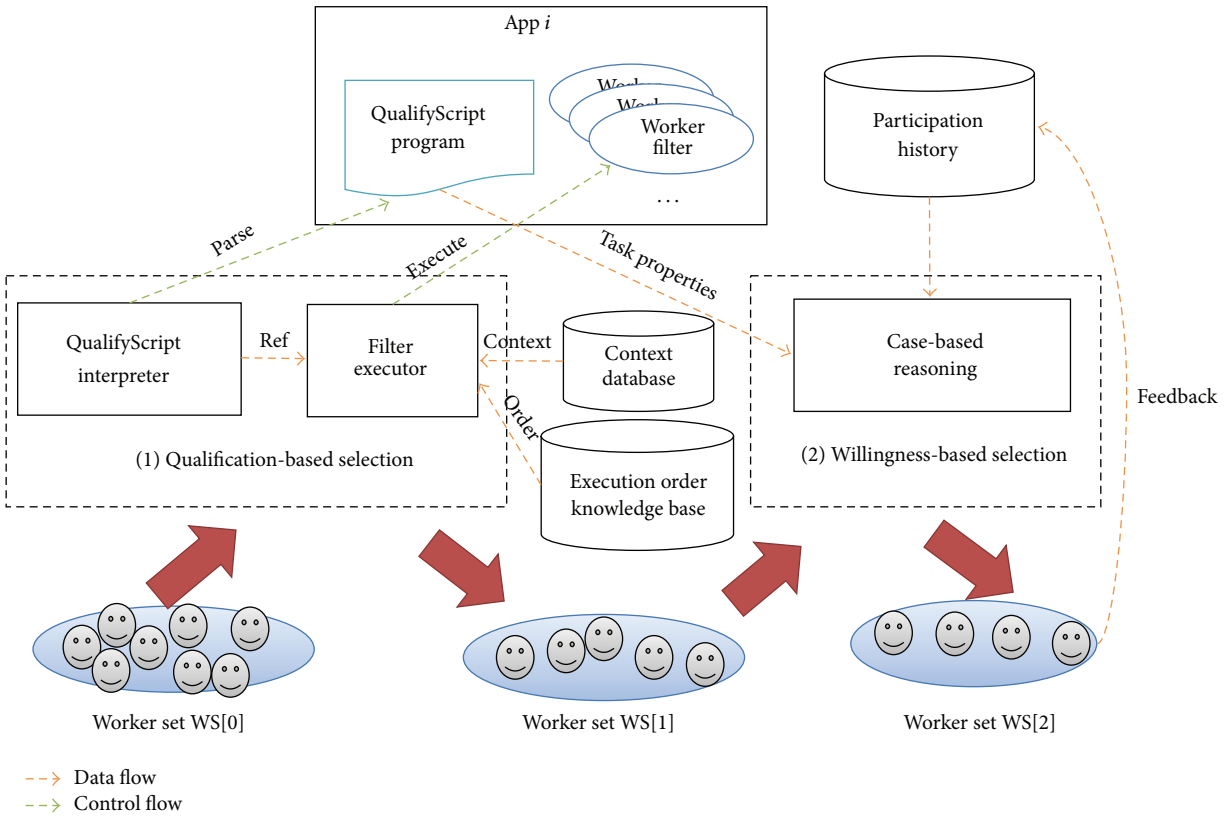


Figure 2: Runtime support.

```
<xml>
  <task_properties>
    <title> Air Quality Report </title>
    <descript>
        report PM 2.5 measurement in the scenic spot.
    </descript>
    <reward> 0.5 USD </reward>
    <assignments> 5 </assignments>
  </task_properties>
  <worker_filters>
    <filter> LocationFilter </filter>
    <filter> SenseCapFilter </filter>
  <worker_filters>
</xml>
```

Box 1: QualifyScript for air quality report task.

The advantage of our design is that it better supports software reuse and maintenance. First, existing filters created by others can be imported into a QualifyScript program very easily when defining a new task. Second, when a new constraint emerges for a predefined task, the task creator only has to implement a new filter and configure it in the QualifyScript without modifying and recompiling others.

3.3.2. Runtime Support. WSelector adopts a two-phase process at runtime for worker selection by taking both the predefined constraints and worker-side factors into account (see Figure 2).

Phase 1—Qualification-Based Selection. In this first phase, we propose a pipe filtering model to select workers who satisfy the predefined constraints. This is done through executing a flow of predefined filters one by one through a virtual pipeline at runtime.

The input for this stage is all registered users in the crowdsensing mediation platform, which is denoted as a worker set WS[0] in Figure 2, together with a predefined QualifyScript program and workers' filters of a certain task.

The qualification-based selection mainly consists of two steps: (1) QualifyScript interpreter parses the QualifyScript program into several references of worker filters and passes them to the filter executer. (2) The worker filters are then executed one by one by the executer (the filters are JARs (JAVA executable packet), while the executor is a component to invoke each JAR). The execution of the worker filters needs the current contexts of workers. For example, a worker filter f_i is to select workers who are currently in a certain place, and the runtime execution of f_i needs the location information of each candidate worker. The acquisition of contexts is the mediation platform's responsibility. We assume that real-time contexts are managed by the mediation platform in

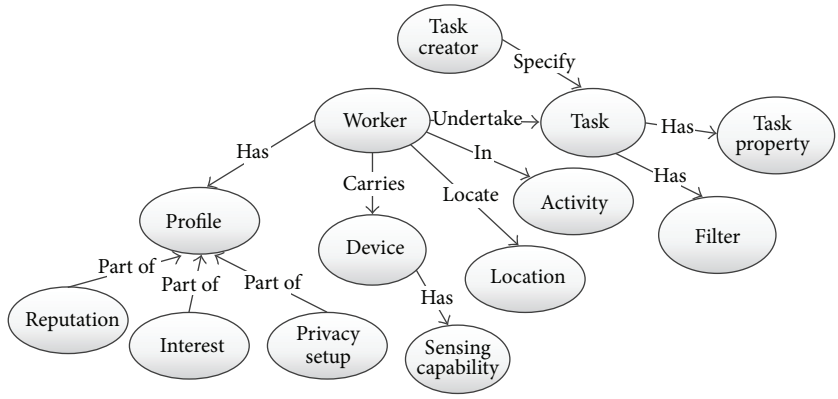


Figure 3: CCMCS: Core Context Model for Crowd-Sensing.

the context database, which our framework can access (see Figure 2).

Ultimately, at the end of the pipeline, we get the selected workers who are qualified for all predefined constraints, which is denoted as the worker set WS[1].

In the crowd-sensing paradigm, energy consumption and privacy violation are two key concerns for workers. It is more satisfactory if the mediation platform updates worker's dynamic or private-sensitive contexts in a lower frequency. For example, updating location information uses more smartphone battery and could lead to violation of privacy, thus workers may not want it to be collected too frequently. The optimized execution orders of filters can make the crowd-sensing mediation platform update the contexts in a more energy-saving and privacy-preserving way. We preestablish an execution order knowledge base (see Figure 2), in which the execution orders of some predefined contexts are specified. At runtime, worker filters will be executed one by one in the filter executer based on the predefined orders in the knowledge base. Filters relevant to contexts that are more static (e.g., those determined when the worker registered to the mediation platform or those which change very slowly) will be executed in front of those filters relevant to contexts that are more dynamic or privacy-sensitive. Take the air quality report task in Box 1 as an example. If the *LocationFilter* is executed at first, then the location information of all workers in WS[0] must be updated. However, if the sensing capability filter (*SenseCapFilter*) is executed before the *LocationFilter*, only workers whose devices are embedded with air quality sensors need to collect and upload their location information to the mediation platform. Therefore, in the execution order knowledge base, we specify that sensing capability filter is executed prior to location filter.

Phase 2—Willingness-Based Selection. After the qualification-based selection phase, if the number of candidates (i.e., |WS[1]|) is still larger than the number of assignments specified in the QualifyScript, the second phase (i.e., the willingness-based selection) will be started.

In this phase, WSelector further selects workers who are more likely to undertake the task. The input of this stage is the worker set WS[1], and the output is k workers in WS[1] who have the highest likelihood of actually undertaking a task if the system pushes it to them (denoted as WS[2]). The parameter k is the number of assignments defined in the QualifyScript program.

We achieve such a goal based on our second insight, which is leveraging the candidate worker's participation history over time to perform a more precise selection. To implement this idea, we propose a case-based reasoning approach to select those k workers, which will be introduced in great detail in Section 5. We assume that the mediation platform has been recording participation history and storing them in the case database, to which our framework can get access (see Figure 2).

4. Context Modeling for Crowd-Sensing

4.1. Core Context Model and Extensibility. We propose an ontology-based context model, named CCMCS (Core Context Model for Crowd-Sensing), to define basic concepts and their relationships in crowd-sensing worker selection (see Figure 3). Each entity in the context model is associated with its attributes (represented in owl:DatatypeProperty) and relations with other entities (represented in owl:ObjectProperty).

Due to space limitation, Figure 3 only shows owl:ObjectProperty. *Task*, *task creator*, and *worker* are the most basic concepts, to which other concepts are related. Generally speaking, each task has its *filters* defining constraints, *task property* including name, description, reward, and assignments. Each worker also has his/her *profile*, *location*, and carried *device*. In the CCMCS, we establish *reputation*, *interest,* and *privacy setup* as part of predefined profiles and *sensing capability* as predefined object property of carried device.

WSelector also provides extension ability for adding task-specific concepts based on the CCMCS. First, the built-in OWL property (such as owl:subClassOf and owl:partOf) allows for most of the extensions. Second, the task creator can

Table 2: Measurable worker-side factor.

Worker-side factor	How it is measured	Possible values	
		Values	Meaning
Incentive (Reward)	Measured by the reward specified in the QualifyScript program	**1**	0~10 US cents
		2	10~50 US cents
		3	50~100 US cents
		4	1~3 USDs
		5	more than 3 USDs
Interest	Measured by how many tasks a worker has undertaken in the same domain	**1**	0 tasks
		2	1~3 tasks
		3	3~5 tasks
		4	5~10 tasks
		5	More than 10 tasks
Data privacy sensitivity	Measured by the sensitivity of the data the worker must provide when to fulfill the task	**1**	Do not provide private data
		2	Only provide coarse-grained location data (e.g., @University of Florida)
		3	Provides very sensitive private data (e.g., sound data via microphone)

define its own OWL property (both the owl:ObjectProperty and owl:dataTypeproperty).

4.2. Crowd-Sourced Context Modeling. Our framework supports crowd-sourced context modeling. It allows a collaborative creation of context models and provides automated mechanisms for validation.

As Figure 1 shows, the context model repository is open to the public, into which everyone can contribute new context models. The repository automatically validates new models for using a unique name and in terms of the OWL (Web Ontology Language) grammar requirements. The automatic validity check ensures the integrity of all models that are published. After acceptance, all context models in the repository can directly be used by other task creators.

Even though they provide better support for context models reuse and for accelerating the task creation, crowd-sourced context modeling mechanisms come at a cost and overhead. Allowing everyone to submit models might lead to the fast growth in the size of repository, because each task creator creates the model that best fits his/her purpose. As an example it can be expected that there are many different models for "smartphone." When searching a specific model (e.g., smartphone), the task creator might be overwhelmed by getting numerous alternatives. Therefore, this paper proposes a mechanism to rank the context models based on their popularity. WSelector collects two types of different data as the metric to rate the popularity of context models. One is the number of downloads of a context model, and the other is the task creator's manual scoring for the model (from 0~10). When a task creator wants to search a model, the framework can sort the alternatives by either the number of downloads or the average scoring.

5. Willingness-Based Selection

In the willingness-based selection phase, we aim to select workers who are more likely to accept to undertake a certain task by leveraging worker's participation history over a period of time.

5.1. Measurable Worker-Side Factors. There are many worker-side factors that may affect a worker's decision to undertake or decline a certain task. However, the current implementation of WSelector takes only those measurable worker-side factors (in Table 2) into consideration. Here the term "measurable" means these factors can be measured by either the task property in the QualifyScript program or the worker behavior learned over time from the mediation platform.

5.2. Case-Based Reasoning Algorithm. We propose a case-based reasoning algorithm to fulfill the willingness-based selection. Although current implementation of the algorithm takes only those worker-side factors in Table 2 into account, the algorithm itself is not limited to these three factors and can be extended easily if additional measurable factors are used in the future.

5.2.1. Problem Formulation. The problem is formulated as follows.

Consider that $\mathrm{WS}_1(T_h) = \{\mathrm{wk}_1, \mathrm{wk}_2, \ldots, \mathrm{wk}_N\}$ is the output of the qualification-based selection phase and the input of willingness-based selection, where wk_i $(i = 1, 2, \ldots, N)$ is a worker satisfying the constraints of task T_h.

The goal is to find K $(K \leq N)$ workers from $\mathrm{WS}_1(T_h)$ who have the highest likelihood of accepting the task T_h. The output is $\mathrm{WS}_2(T_h) = \{\mathrm{wk}_{p_1}, \mathrm{wk}_{p_2}, \ldots, \mathrm{wk}_{p_k}\}$, where K is specified in the property "assignments" of the QualifyScript program by the task creators. For example, in the example of Box 1, $K = 5$.

5.2.2. The Algorithm. We first define the concept of historical case.

Consider that $\overrightarrow{\mathrm{WF}}_{i\theta} = (c_{1\theta}, c_{2\theta}, \ldots, c_{m\theta})$ is a *worker-side factor vector* of task T_θ on worker i. For example, in the current implementation of the framework, $m = 3$, and

$(1, 2, 3)$ means that a certain task provides the reward of 0~10 US cents; the worker i has undertaken 1~3 tasks in the same domain; and the task requires the worker to provide very sensitive private data. A *historical case* is defined as $(i, \theta, \overrightarrow{\mathrm{WF}}_{i\theta}, \lambda)$, where i and θ are the identity of a worker and a task, respectively, $\overrightarrow{\mathrm{WF}}_{i\theta}$ is a m-dimensional worker-side factor vector, and λ is the two-valued outcome (accept/decline).

Our case-based reasoning algorithm consists of the following two steps.

Step 1 (case selection). Since the influence of different worker-side factors varies from one worker to another, the algorithm first selects historical cases of worker i to compute the likelihood of a worker i to undertake the task T_h. Using selected cases, the following two reference matrices are established, which are the positive reference matrices $(\overrightarrow{\mathrm{RP}}_1, \overrightarrow{\mathrm{RP}}_2, \ldots, \overrightarrow{\mathrm{RP}}_{x(i)})^{\mathbf{T}}$ and the negative reference matrix $(\overrightarrow{\mathrm{RN}}_1, \overrightarrow{\mathrm{RN}}_2, \ldots, \overrightarrow{\mathrm{RN}}_{y(i)})^{\mathbf{T}}$, where $x(i)$ and $y(i)$ are the number of "accept" (positive) and "decline" (negative) cases, respectively. $\overrightarrow{\mathrm{RP}}_j = (a_{j1}, a_{j2}, \ldots, a_{jm})$ is a worker-side factor vector whose corresponding outcome λ is "accept." $\overrightarrow{\mathrm{RN}}_j = (b_{j1}, b_{j2}, \ldots, b_{jm})$ is a worker-side factor vector whose corresponding feedback λ is "decline":

$$\begin{aligned}
&\left(\overrightarrow{\mathrm{RP}}_1, \overrightarrow{\mathrm{RP}}_2, \ldots, \overrightarrow{\mathrm{RP}}_{x(i)}\right)^{\mathbf{T}} \\
&\quad = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ a_{x(i)1} & a_{x(i)2} & \cdots & a_{x(i)m} \end{pmatrix}, \\
&\left(\overrightarrow{\mathrm{RN}}_1, \overrightarrow{\mathrm{RN}}_2, \ldots, \overrightarrow{\mathrm{RN}}_{y(i)}\right)^{\mathbf{T}} \\
&\quad = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ b_{y(i)1} & b_{y(i)2} & \cdots & b_{y(i)m} \end{pmatrix}.
\end{aligned} \tag{1}$$

Step 2 (likelihood measurement and worker selection). We use $U(i, h)$ to measure the likelihood of worker i to accept task T_h (see (2)), where $\mathrm{Dist}(\overrightarrow{\mathrm{WF}}_{ih}, \overrightarrow{\mathrm{RN}}_j)$ is the Euclidean distance between $\overrightarrow{\mathrm{WF}}_{ih}$ and $\overrightarrow{\mathrm{RN}}_j$ and $\mathrm{Dist}(\overrightarrow{\mathrm{WF}}_{ih}, \overrightarrow{\mathrm{RP}}_j)$ is the Euclidean distance between $\overrightarrow{\mathrm{WF}}_{ih}$ and $\overrightarrow{\mathrm{RP}}_j$. K workers with maximum $U(i, h)$ are what we want to select:

$$\begin{aligned}
U(i, h) = &\sum_{j=1}^{y(i)} \mathrm{Dist}\left(\overrightarrow{\mathrm{WF}}_{ih}, \overrightarrow{\mathrm{RN}}_j\right) \\
&- \sum_{j=1}^{x(i)} \mathrm{Dist}\left(\overrightarrow{\mathrm{WF}}_{ih}, \overrightarrow{\mathrm{RP}}_j\right).
\end{aligned} \tag{2}$$

The intuition behind this function is that the more likely a worker i is to accept task T_h, the closer $\overrightarrow{\mathrm{WF}}_{ih}$ is to positive cases and the more distant it is from negative cases.

Note that different worker-side factors could have different impact on deciding the outcome; the weights of different factors have to be considered (see (3)) when calculating the distance, where $w_i \in (0, 1)$ is the weight of ith worker-side factor:

$$\begin{aligned}
\mathrm{Dist}\left(\overrightarrow{\mathrm{WF}}_{ih}, \overrightarrow{\mathrm{RN}}_j\right) &= \sqrt{\sum_{p=1}^{m} \left(c_{ph} - b_{jp}\right)^2 * w_p}, \\
\mathrm{Dist}\left(\overrightarrow{\mathrm{WF}}_{ih}, \overrightarrow{\mathrm{RP}}_j\right) &= \sqrt{\sum_{p=1}^{m} \left(c_{ph} - a_{jp}\right)^2 * w_p}.
\end{aligned} \tag{3}$$

Now the key problem is how to precompute the weight w_p $(p = 1, 2, \ldots, m)$. We propose a weight calculation algorithm based on the *sensitivity analysis principle* [37]. This principle is to identify the key variables for a certain target, by calculating the effect of variable's change on the target's change. This principle mainly consists of the following steps:

(a) Determine the variables and target to be analyzed according to the problem.

(b) Vary only one variable, observing and measuring the change of target.

(c) Repeat the analysis in (b) for each variable repetitively for calculating the corresponding effect on the target.

Based on the above sensitivity analysis principle, we propose a *personalized weight calculation* algorithm according to our problem. Some key points of this algorithm are explained as follows.

(1) The variables are worker-side factors, and the target is the outcome (accept/decline). Matrix $\mathbf{A} = (\vec{Q}_1, \vec{Q}_2, \ldots, \vec{Q}_{[x(i)+y(i)]})^{\mathbf{T}}$ is all cases of worker i, which is the mergences of positive reference matrix $(\overrightarrow{\mathrm{RP}}_1, \overrightarrow{\mathrm{RP}}_2, \ldots, \overrightarrow{\mathrm{RP}}_{x(i)})^{\mathbf{T}}$ and the negative reference matrix $(\overrightarrow{\mathrm{RN}}_1, \overrightarrow{\mathrm{RN}}_2, \ldots, \overrightarrow{\mathrm{RN}}_{y(i)})^{\mathbf{T}}$.

(2) Lines 3~19 calculate the weight of a certain worker-side factor x_k by changing x_k and remaining others unchanged.

(3) In lines 11~12, $L = \mathrm{Max}\{q_{1k}, q_{2k}, \ldots, q_{(x(i)+y(i))k}\} - \mathrm{Min}\{q_{1k}, q_{2k}, \ldots, q_{(x(i)+y(i))k}\}$ is the maximum difference among the possible values of the kth variable, and $L/|q_{jk} - q_{lk}|$ measures the effect of variable x_k's change on the target's change. It is reasonable, because when $|q_{jk} - q_{lk}|$ is smaller and the target has changed, it means that even the slightest change of x_k is enough to leading to the change of target (indicating that this variable's weight is heavier).

(4) We adjust the weight to make sure that their sum is equal to 1 (in line 20).

Input: Matrix **A** is the merging of positive and negative reference matrix of the worker i, and vector **B** is the corresponding feedbacks ($\vec{Q}_j$ corresponds to λ_j).

$$\mathbf{A} = (\vec{Q}_1, \vec{Q}_2, \ldots, \vec{Q}_{x(i)+y(i)})^{\mathbf{T}} = \begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1m} \\ q_{21} & q_{22} & \cdots & q_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ q_{(x(i)+y(i))1} & q_{(x(i)+y(i))2} & \cdots & q_{x((i)+y(i))m} \end{pmatrix}$$

$$\mathbf{B} = (\lambda_1, \lambda_2, \ldots, \lambda_{(x((i)+y(i))})^{\mathbf{T}} \ \lambda_i \in \{\text{accept, decline}\}.$$

Output: w_i $(i = 1, 2, \ldots, m)$

(1) **ALGORITHM BEGIN**
(2) float total = 0, effect = 0;
(3) **FOR** ($k = 1$ to $k = m$){
(4) /* Calculate the weight w_k. */
(5) **FOR** (from $j = 1$ to $j = x(i) + y(i)$)
(6) **FOR** (from $l = j + 1$ to $l = x(i) + y(i)$)
(7) **IF** (& $q_{jk} \neq q_{lk}$ &
(8) other components of $\vec{Q}_j$ and $\vec{Q}_l$ are identical)
(9) total ++;
(10) **IF** ($\lambda_j \neq \lambda_l$)
(11) $L = \text{Max}\{q_{1k}, q_{2k}, \ldots, q_{(x(i)+y(i))k}\} -$
(12) $\text{Min}\{q_{1k}, q_{2k}, \ldots, q_{(x(i)+y(i))k}\}$.
(13) effect = effect + $L/|q_{jk} - q_{lk}|$;
(14) **END FOR**
(15) **END FOR**
(16) w_i = effect/total.
(17) /* clear the parameters for calculating next weight. */
(18) total = 0, effect = 0.
(19) **END FOR**
(20) **FOR** ($j = 1$ to $j = m$) $w_j = w_j / \sum_1^m w_i$
(21) //Adjust the weight to make their sum to be 1
(22) **ALGORITHM END**

Algorithm 1: Personalized weight calculation algorithm.

5.2.3. Strategy for Cold-Start Problem. We may encounter the cold-start problem when a worker just registered in a crowd-sensing mediation platform. In this situation, the number of one's historical cases is zero and our framework knows nothing about the new worker.

We deal with the cold-start problem by leveraging the cases of other workers, because users tend to make similar decisions under similar contexts. The approach is almost similar to Algorithm 1, but is different in the following two aspects.

(1) Case Selection. In the case selection step, we select historical cases of other workers whose worker-side factor vector is identical to the current context vector $\overrightarrow{\text{WF}}_{ih}$ of worker i and task h.

(2) General Weight Calculation. The weight calculation algorithm is almost the same as the personalized weight calculation algorithm, but the input is changed to the entire datasets (all cases from all workers). The calculated weights are referred to as the *general weight*, which will be used in the likelihood measurement and worker selection.

5.3. Iterative Selection Strategy

5.3.1. Problem Statement. After the execution of willingness-based worker selection algorithm, K workers will be selected and the task is pushed to them. However, in real-world circumstances, the selected workers may not be able to accomplish the task due to some unexpected reasons.

For example, a candidate worker, named Tom, has a very high probability of accepting a certain crowd-sensing task according to his historical participation records and the attributes of the task. Thus, our willingness-based worker selection algorithm selects him, and the platform pushes the task to him. Nevertheless, Tom has something urgent to do all of a sudden, so he declines the pushed task.

Therefore, sometimes the number of workers who actually accept and accomplish the task may be less than K, which does not satisfy the requirement of the task creator.

5.3.2. Strategy Description. To deal with above-stated problem, this paper proposes an iterative worker selection strategy. The main idea of the strategy is to divide the willingness-based selection phase into the iterative execution of several

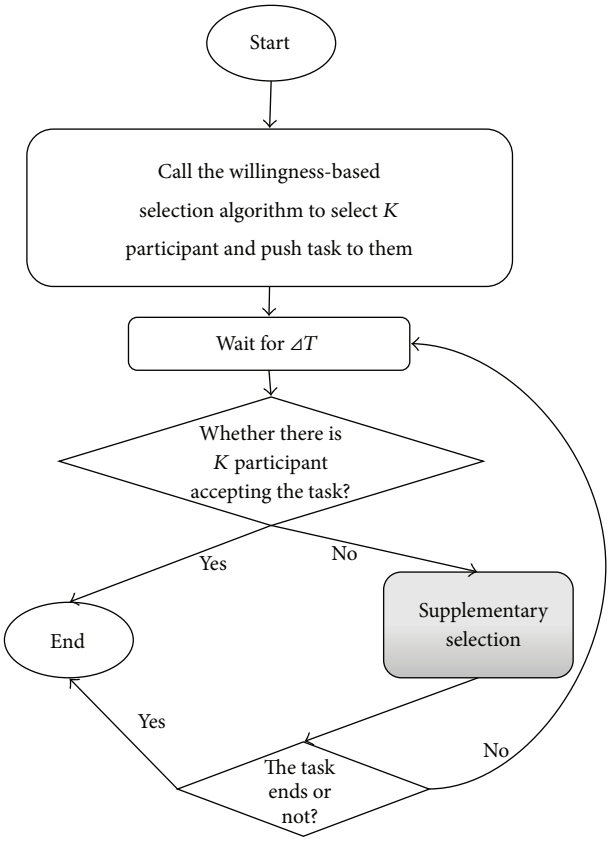


Figure 4: Iterative selection process.

substeps, by which it can achieve a high probability that there are K workers actually accepting the task.

Figure 4 shows the workflow of the strategy. Before the execution of workflow, willingness-based worker selection algorithm will calculate the all qualified candidate workers' probability of accepting a task named tsk. Then the workflow is executed as follows:

(1) Select K workers who are more likely to accept the task.

(2) Observe the result after a short period of time (denoted as ΔT). If the total number of workers who accept the task reaches K or the task is out of time, then the entire process comes to an end. For a certain task, we denote its entire duration as $[\theta_{\text{start}}, \theta_{\text{end}}]$, where θ_{start} and θ_{end} are the start and end time point. We assume that $\theta_{\text{start}} - \theta_{\text{end}} > 2\Delta T$, ensuring that our iterative process has at least one chance to reselect workers.

(3) Reselect certain number of workers (supplementary selection) and then go to (2). Here, the reselected workers are those who have not been selected in previous iterations and with highest likelihood of accepting the task.

Now the key problem is how many workers should be selected in each of the iterations.

We denote the number of selected workers in each of the iterations as x_i ($i = 1, 2, \ldots,$ max_round), where max_round is the maximum number of iterations decided by $(\theta_{\text{start}} - \theta_{\text{end}})/\Delta T$.

For the ith iteration, the number of reselected workers λ_i should satisfy the following formula:

$$\frac{\sum_{j=1}^{j=i-1} x_j}{\alpha * \sum_{j=1}^{j=i-1} \lambda_j} = \frac{E(x_i)}{\lambda_i}, \tag{4}$$

where $\sum_{j=1}^{j=i-1} \lambda_j$ is the total number of selected workers in the previous $i-1$ iterations, $\sum_{j=1}^{j=i-1} x_j$ is the total number of workers who have accepted the task, $E(x_i)$ is the expectation of the number of workers who will accept the task in the ith iteration. Here we use the concept of expectation, because at this time we do not know how many reselected workers in the ith iteration will eventually accept the task. In each of the iterations, we use the greedy strategy hoping that after the iteration the total number of workers accepting the task reaches K. Thus, $E(x_i) = K - \sum_{j=1}^{j=i-1} x_j$. The parameter α is used to characterize the probability of accepting the task by analyzing the historical participation record, and it is calculated as follows:

$$\alpha = \frac{\sum_{j=1}^{j=\text{Sum}} U(w_j, \text{tsk})}{\sum_{j=\text{Sum}+1}^{j=\text{Sum}+E(x_i)+1} U(w_j, \text{tsk})}, \quad \text{where Sum} = \sum_{j=1}^{j=i-1} x_j. \tag{5}$$

Therefore, the number of workers that should be selected in each of the iterations is calculated as

$$\lambda_i = \frac{\alpha * \left(K - \sum_{j=1}^{j=i-1} x_j\right) * \sum_{j=1}^{j=i-1} \lambda_j}{\sum_{j=1}^{j=i-1} x_j}. \tag{6}$$

The probability that the total number of workers accepting the task reaches K after the ith iteration is denoted as p_i ($i = 1, 2, \ldots,$ max_round). Then after max_round iterations, the probability that total number of workers is still less than K is Prob (failure) $= (1 - p_1) * (1 - p_2) * \cdots * (1 - p_{\text{max\_round}})$.

Thus after max_round iterations, the probability that total number of workers reaches K is Prob (success) $= 1 - (1 - p_1) * (1 - p_2) * \cdots * (1 - p_{\text{max\_round}})$.

As $p_i \in (0, 1)$, $1 - p_i \in (0, 1)$, Prob (success) will get close to 1 with the increase of the number of iterations. On the other hand, max_round is decided by $(\theta_{\text{start}} - \theta_{\text{end}})/\Delta T$. Therefore, the iterative selection strategy described above can make Prob (success) get close to 1 by reducing ΔT.

6. Evaluation

In this section, we evaluate the qualification-based and willingness-based selection of WSelector. The goals of our evaluation are as follows. (1) For the qualification-based selection, we first demonstrate the extensibility of the core

Table 3: Constraint for worker selection.

Crowd-sensing task	Constraint for worker selection
Petrol Watch [30]	Location, sensing capability (camera)
Haze Watch [31]	Location, sensing capability (air quality sensor)
Citizen Journalist [32]	Location, sensing capability (camera)
EarPhone [33]	Location, sensing capability (microphone), privacy setup (microphone can be accessed)
Nericell [34]	Location, activity (driving), sensing capability (accelerometer), privacy setup (accelerometer can be accessed)
Bus Waiting Time Prediction [35]	Activity (riding bus), sensing capability (accelerometer)
QTime [36]	Location, activity (queuing), sensing capability (accelerometer)

context model and the expressiveness of the programming interfaces. Then we evaluate the correctness of the runtime selection. (2) For the willingness-based selection, we evaluate the validity of our case-based reasoning algorithm.

Our experiments are conducted on a prototype of WSelector. The core context model is created using the Protégé [38], a free and open-source ontology editor. The management of the crowd-sourced context modeling process is implemented with the help of OWL API [39]. The programming interfaces and the runtime environments (including the QualifyScript interpreter, filter executor, and case-based reasoning module) are implemented using Java. The context database and case database are established and managed by the MySQL system, and Java SDK is imported for accessing the MySQL database.

6.1. Qualification-Based Selection. We have implemented worker selection module of 10 crowd-sensing tasks, among which 7 tasks are from literature review and other three are borrowed from the examples we mentioned in the introduction section. The constraints for worker selection of these tasks are listed in Table 3.

In Table 3, we assume that all mobile devices are all embedded with modules for localization, so that we do not include it in the sensing capability. Since this paper does not focus on the context collection from mobile nodes and assumes this to be the crowd-sensing mediation platform's duty, the client side of these crowd-sensing tasks ranges from mobile phone to wearable devices.

By implementing the worker selection of all these tasks, we can report the results as follows.

(1) All these constraints can be modeled by our framework successfully, by either directly importing the classes and properties in the core context model or extending by the user-defined ones.

(2) Our programming interfaces are expressive enough for supporting the task creator to define relevant worker filters. The ontology classes and properties in the task-specific ontology model are transformed into Java classes and interfaces, which are used to develop relevant worker filters. Finally, we define the reference of worker filters in the QualifyScript program. Then we randomly generated 100 testing cases for each task to evaluate the result of qualification-based selection at runtime. Each testing case corresponds to one candidate worker and contains the values for different constraints. We manually label each worker as "selected" or "not selected" based on the constraints in Table 3, which are taken as the ground truth. By running the implemented worker selection module in the runtime environments of WSelector, we demonstrate that all the $10 * 100 = 1000$ selection results obey the predefined constraints.

6.2. Willingness-Based Selection

6.2.1. Data Collection. We post a questionnaire on an online platform to generate the dataset [40]. First, we assume that there is a crowd-sensing task with basic description. Then we design 15 questions, each of which ask for the outcome under different combination of worker-side factors. Second, we invite 26 volunteers (including the undergraduate, graduate students, and faculties in our institute) to respond to this questionnaire. Third, we collect all these questionnaires and generated the dataset. The generated dataset consists of $75 * 26 = 1950$ historical cases (each question and its answer can generate 5 cases of a worker, so that each worker has $15 * 5 = 75$ cases and the overall dataset consists of $75 * 26$ cases).

6.2.2. Experimental Methodology. After the dataset has been generated, we design a 6-round experiment to evaluate the case-based reasoning algorithm for willingness-based worker selection. The generated datasets are used to establish both the case database, which is the input of our algorithm, and testing cases containing the ground truth. Our 6-round experiment is summarized in Table 4.

(1) The First Round Is for the Baseline Method. It is assumed that none of the cases have been preacquired, so that we can only randomly select k workers from 26 candidates. This round of experiment consists of the following steps: (a) randomly assign value to the components of worker-side factor vector $\overrightarrow{\mathrm{WF}}_{ih}$. (b) Perform the baseline method, that is, guessing k workers randomly. (c) Check how many of those guessed workers will undertake the task based on the ground truth in the datasets. (d) Perform (a)~(c) for 10 times and calculate the average number of right-selected workers.

(2) The 2nd~6th Round of the Experiments Is to Test Our Case-Based Reasoning Algorithm When the Number of Cases in the Case Database Changes. (1) The 2nd~4th round of

Table 4: Summary of 6-round experiment.

Round	The number of cases as historical data	Selection approach
1st round	Zero	Baseline method (random selection)
2nd round	Each worker has 25 cases	Approach in Section 5.2.2
3rd round	Each worker has 50 cases	Approach in Section 5.2.2
4th round	Each worker has 75 cases	Approach in Section 5.2.2
5th round	Two workers have no historical cases Each of the other workers has 50 cases	Approach in Section 5.2.2 + Approach in Section 5.2.3
6th round	Five workers have no historical cases Each of the other workers has 50 cases	Approach in Section 5.2.2 + Approach in Section 5.2.3

experiments assumes that each of 26 candidate workers has preacquired cases, and the difference among these three rounds is the number of cases. In these three rounds, we calculate the personalized weight based on the approach in Section 5.2.2. (2) The 5th~6th round assumes that some workers do not have preacquired cases (3 and 5 workers, resp.). In these two rounds, we will use our mechanism in Section 5.2.3 for workers without historical cases. In both 2nd~6th rounds, we first randomly generated 100 worker-side factor vectors as testing cases. Then for each vector, we changed the second component (i.e., interest) randomly since each candidate worker's interest for the same task may be different and kept the other two components (i.e., reward and data privacy sensitivity) unchanged since they are the same among workers for a specific task.

6.2.3. Experimental Results. The willingness-based selection accuracy of each round is measured by Accuracy(i) $= T(i)/k$ ($i = 1, 2, \ldots, 6$), where $T(i)$ is the number of workers in the ith round who will undertake the task based on the ground truth and k is the number of selected workers.

Since the number of selected workers (i.e., k) has impact on the selection accuracy, we conduct our 6-round experiments for 3 times by changing k from 10 and 15 to 20, respectively. Hence we have completed $3 * 6 = 18$ rounds of experiments in total, whose selection accuracy is demonstrated in Figure 5.

According to the observation of the results, we can draw the following conclusions:

(1) Our case-based reasoning algorithm outperforms the baseline method no matter $k = 10$, 15, or 20 and no matter the number of cases as historical data is 25, 50, or 75. Therefore, it shows the effectiveness of our algorithm in different situations.

(2) The smaller k is, the bigger the increase in selection accuracy of our algorithm is compared to the baseline method.

(3) Compared to the 2nd, 3rd, and 4th rounds, we can see that the selection accuracy of our algorithm is improved with the increasing of the number of historical cases.

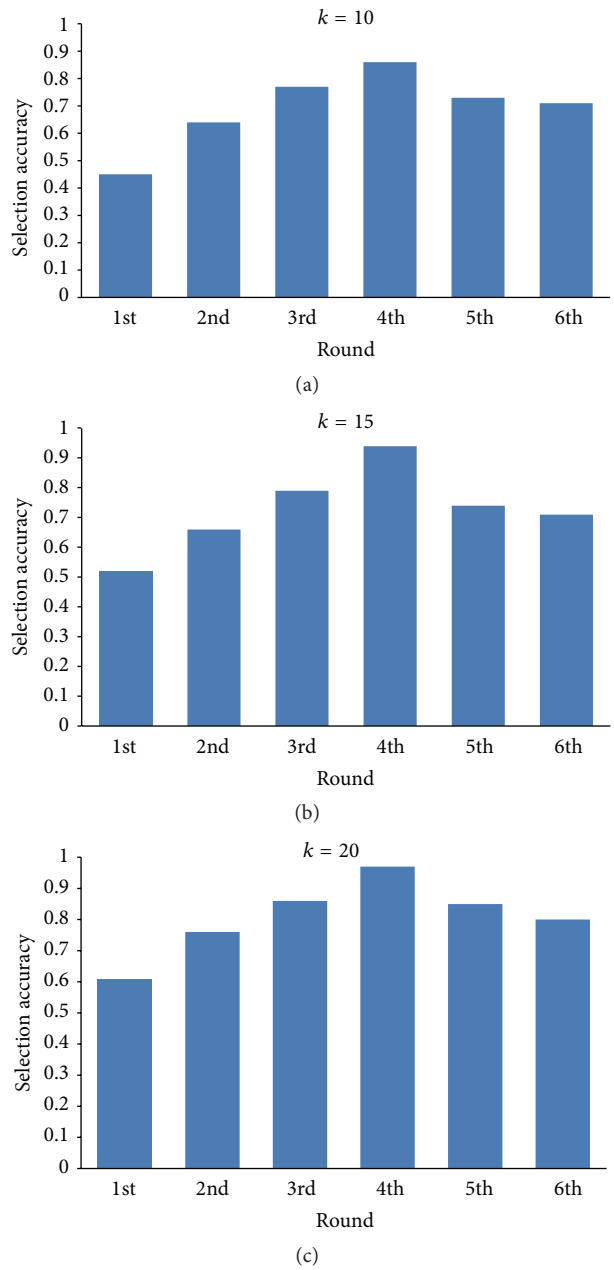


Figure 5: Selection accuracy of different rounds.

(4) The experimental result of 5th and 6th round shows that our mechanism in Section 5.2.3 is effective to solve the cold-start problem to some extent. Although the accuracy of 5th and 6th round is lower than the 3rd round, it still significantly outperforms the baseline method.

7. Discussion

In this section, we discuss some limitations of WSelector and additional important issues, some of which will be the subject of our future work.

7.1. Access to Mediation Platforms. A fundamental assumption in this paper is that WSelector can get access to the participation history in the mediation platform. In practical usage, WSelector is to be integrated into an existing crowd-sensing mediation platform. If our framework is effective enough in selecting appropriate workers, it is reasonable to assume that mediation platforms will provide participation history APIs, to enable a loose and inexpensive integration. At the same time, in our future work, we plan to propose a simple API of the same, which could guide and encourage existing mediation platforms to implement it. Besides, device's functionalities/sensing capabilities are determined when the worker registers to a certain crowd-sensing platform. When integrated to the platform, our framework can get the device's functionalities of each worker.

7.2. The Number of Worker-Side Factors. Although current implementation and evaluation of the case-based reasoning algorithm only take three worker-side factors into account, the algorithm itself is not limited to these factors only. It can be extended easily if we find new measurable ones in future work. For example, if a mediation platform requires task creator to assign value for a new task property (e.g., the approximate workload) when specifying the task, we can easily add it as the fourth factor for willingness-based selection. In our future work, we will analyze and propose a broader set of worker-side factors. We will also include additional factors that we may learn about from the literature. However, when more factors can be added, there is a new challenge, that is, how to determine which subset of factors is more significant in selecting workers? We plan to exploit feature selection mechanisms in the future work.

7.3. Dynamic Adjustment of the Number of Selected Workers. This paper assumes that all workers are selected before they start to undertake the task. However, not all crowd-sensing tasks obey this assumption. For example, a crowd-sensing task may require that selected workers must meet certain geographical coverage rather than reaching certain number in total. In this case, it is a better strategy to dynamically adjust the number of selected workers online according to the current distribution of workers who have already fulfilled the task. Therefore, we plan to propose a dynamic worker selection mechanism in the future work to make WSelector able to handle more crowd-sensing tasks.

7.4. Fine-Grained Privacy Measurement. In the current implementation of WSelector, we consider privacy as one of three contexts in the willingness-based selection. However, since our main contribution mainly lies in the case-based reasoning algorithm, we only measure privacy factor in a very simple way by giving them three possible values. However, the measurement of the privacy factor is much more complex in reality. We plan to leverage the model proposed in [41] to measure the privacy factor in a more fine-grained manner.

8. Conclusion

In this paper, we proposed a novel worker selection framework, named WSelector, to select appropriate workers for crowd-sensing more precisely by taking various contexts into account. It first provides programming time support to help task creator define all constraints. Then a two-phase process was adopted at runtime to select workers who not only are qualified but also have higher possibility to undertake a crowd-sensing task. Evaluations with 11 crowd-sensing tasks indicate the expressiveness of our core ontology model and programming interface. Besides, evaluations with a questionnaire-generated dataset show that our case-based reasoning algorithm outperforms the baseline method.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is funded by the National High Technology Research and Development Program of China (863) under Grant no. 2013AA01A605. Besides, this research is supported by NSFC Grant (no. 61572048) and Microsoft Collaboration Research Grant.

References

[1] Amazon mechanical turk, https://www.mturk.com/.

[2] M.-R. Ra, B. Liu, T. F. La Porta, and R. Govindan, "Medusa: a programming framework for crowd-sensing applications," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys '12)*, pp. 337–350, ACM, Lake District, UK, June 2012.

[3] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin, "CrowdDB: answering queries with crowdsourcing," in *Proceedings of the ACM SIGMOD International Conference on Management of data (SIGMOD '11)*, pp. 61–72, Athens, Greece, June 2011.

[4] T. Yan, V. Kumar, and D. Ganesan, "CrowdSearch: exploiting crowds for accurate real-time image search on mobile phones," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys '10)*, pp. 77–90, ACM, San Francisco, Calif, USA, June 2010.

[5] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller, "TurKit: human computation algorithms on mechanical turk," in *Proceedings of the Proceedings of the 23nd Annual ACM Symposium*

on User Interface Software and Technology, pp. 57–66, ACM, October 2010.

[6] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment framework for participatory sensing data collections," in *Pervasive Computing*, vol. 6030 of *Lecture Notes in Computer Science*, pp. 138–155, Springer, Berlin, Germany, 2010.

[7] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma, "PRISM: platform for remote sensing using smartphones," in *Proceedings of the 8th Annual International Conference on Mobile Systems, Applications and Services (MobiSys '10)*, pp. 63–76, ACM, June 2010.

[8] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos, "Anonysense: privacy-aware people-centric sensing," in *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, pp. 211–224, ACM, usa, June 2008.

[9] G. Cardone, L. Foschini, P. Bellavista et al., "Fostering participaction in smart cities: a geo-social crowdsensing platform," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 112–119, 2013.

[10] D. Zhang, H. Xiong, L. Wang, and G. Chen, "CrowdRecruiter: Selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 703–714, ACM, September 2014.

[11] S. Reddy, K. Shilton, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using context annotated mobility profiles to recruit data collectors in participatory sensing," in *Location and Context Awareness*, pp. 52–69, Springer, Berlin, Germany, 2009.

[12] T. Yan, M. Marzilli, R. Holmes, D. Ganesan, and M. Corner, "Demo abstract: mCrowd—a platform for mobile crowdsourcing," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*, pp. 347–348, ACM, Berkeley, Calif, USA, November 2009.

[13] A. Joki, J. Burke, and D. Estrin, "Campaignr: a framework for participatory data collection on mobile phones," Tech. Rep., UCLA Center for Embedded Network Sensing, 2007.

[14] E. Cuervo, P. Gilbert, B. Wu, and L. P. Cox, "Crowdlab: an architecture for volunteer mobile testbeds," in *Proceedings of the 3rd International Conference on Communication Systems and Networks (COMSNETS '11)*, Bangalore, India, January 2011.

[15] Y. Xiao, P. Simoens, P. Pillai, K. Ha, and M. Satyanarayanan, "Lowering the barriers to large-scale mobile crowdsensing," in *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications (HotMobile '13)*, ACM, Jekyll Island, Georgia, February 2013.

[16] Z. Song, B. Zhang, C. H. Liu, A. V. Vasilakos, J. Ma, and W. Wang, "QoI-aware energy-efficient participant selection," in *Proceedings of the 11th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON '14)*, pp. 248–256, IEEE, Singapore, July 2014.

[17] S. He, D.-H. Shin, J. Zhang, and J. Chen, "Toward optimal allocation of location dependent tasks in crowdsensing," in *Proceedings of the 33rd IEEE Conference on Computer Communications (INFOCOM '14)*, pp. 745–753, IEEE, Toronto, Canada, May 2014.

[18] P. Öztürk and A. Aamodt, "Towards a model of context for case-based diagnostic problem solving," in *Proceedings of the 1st International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT '97)*, pp. 198–208, Rio de Janeiro, Brazil, February 1997.

[19] T. Strang, *Service interoperability in ubiquitous computing environments [Ph.D. thesis]*, Ludwig-Maximilians-University Munich, Munich, Germany, 2003.

[20] J. De Bruijn, "Using ontologies—enabling knowledge sharing and reuse on the semantic web," Tech. Rep. DERI-2003-10-29, Digital Enterprise Research Institute (DERI), Innsbruck, Austria, 2003.

[21] T. Strang, C. Linnhoff-Popien, and K. Frank, "CoOL: a context ontology language to enable contextual interoperability," in *Distributed Applications and Interoperable Systems*, vol. 2893 of *Lecture Notes in Computer Science*, pp. 236–247, Springer, Berlin, Germany, 2003.

[22] T. Strang, C. Linnhoff-Popien, and K. Frank, "Applications of a context ontology language," in *Proceedings of the International Conference on Software, Telecommunications and Computer Networks (SoftCom '03)*, pp. 14–18, Split, Croatia, October 2003.

[23] T. Gu, X. H. Wang, H. K. Peng, and D. Q. Zhang, "Ontology based context modeling and reasoning using OWL," in *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS '04)*, San Diego, Calif, USA, January 2004.

[24] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung, "Ontology based context modeling and reasoning using OWL," in *Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops (PerCom '04)*, pp. 18–22, IEEE, Orlando, Fla, USA, March 2004.

[25] M.-O. Pahl and G. Carle, "Crowdsourced context-modeling as key to future smart spaces," in *Proceedings of the Network Operations and Management Symposium (NOMS' 14)*, pp. 1–8, IEEE, Krakow, Poland, May 2014.

[26] C. W. Holsapple and K. D. Joshi, "A collaborative approach to ontology design," *Communications of the ACM*, vol. 45, no. 2, pp. 42–47, 2002.

[27] S. Karapiperis and D. Apostolou, "Consensus building in collaborative ontology engineering processes," *Journal of Universal Knowledge Management*, vol. 1, no. 3, pp. 199–216, 2006.

[28] V. Ludovici, F. Smith, and F. Taglino, "Collaborative ontology building in virtual innovation factories," in *Proceedings of the International Conference on Collaboration Technologies and Systems (CTS '13)*, pp. 443–450, San Diego, Calif, USA, May 2013.

[29] A. Kalyanpur, D. J. Pastor, S. Battle, and J. A. Padget, "Automatic mapping of OWL ontologies into java," in *Proceedings of the 16th International Conference on Software Engineering & Knowledge Engineering (SEKE '04)*, vol. 4, pp. 98–103, Alberta, Canada, June 2004.

[30] Y. F. Dong, S. S. Kanhere, C. T. Chou, and N. Bulusu, "Automatic collection of fuel prices from a network of mobile cameras," in *Distributed Computing in Sensor Systems: 4th IEEE International Conference, DCOSS 2008 Santorini Island, Greece, June 11–14, 2008 Proceedings*, vol. 5067 of *Lecture Notes in Computer Science*, pp. 140–156, Springer, Berlin, Germany, 2008.

[31] V. Sivaraman, J. Carrapetta, K. Hu, and B. G. Luxan, "HazeWatch: a participatory sensor system for monitoring air pollution in Sydney," in *Proceedings of the IEEE 38th Conference on Local Computer Networks Workshops (LCN '13)*, pp. 56–64, IEEE Computer Society, Sydney, Australia, October 2013.

[32] R. Sasank, K. Shilton, G. Denisov, C. Cenizal, D. Estrin, and M. Srivastava, "Biketastic: sensing and mapping for better biking," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*, 1820, 1817 pages, Atlanta, Ga, USA, April 2010.

[33] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu, "Ear-phone: an end-to-end participatory urban noise mapping system," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '10)*, pp. 105–116, ACM, Stockholm, Sweden, April 2010.

[34] P. Mohan, V. Padmanabhan, and R. Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys '08)*, pp. 323–336, Raleigh, NC, USA, November 2008.

[35] P. Zhou, Y. Zheng, and M. Li, "How long to wait: predicting bus arrival time with mobile phone based participatory sensing," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys '12)*, pp. 379–392, ACM, June 2012.

[36] Y. Wang, J. Wang, and X. Zhang, "QTime: a queuing-time notification system based on participatory sensing data," in *Proceedings of the IEEE 37th Annual Computer Software and Applications Conference (COMPSAC '13)*, pp. 770–777, IEEE Computer Society, 2013.

[37] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto, *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models*, John Wiley & Sons, 2004.

[38] http://protege.stanford.edu/.

[39] S. Bechhofer, R. Volz, and P. Lord, "Cooking the semantic web with the OWL API," in *The Semantic Web—ISWC 2003*, vol. 2870 of *Lecture Notes in Computer Science*, pp. 659–675, Springer, Berlin, Germany, 2003.

[40] (Questionnaire) http://www.sojump.com/jq/3999847.aspx.

[41] J. Lin, S. Amini, J. I. Hong, N. Sadeh, J. Lindqvist, and J. Zhang, "Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing," in *Proceedings of the14th International Conference on Ubiquitous Computing (UbiComp '12)*, pp. 501–510, ACM, Pittsburgh, Pa, USA, September 2012.

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2016, Article ID 9274715, 10 pages
http://dx.doi.org/10.1155/2016/9274715

Research Article

Twitter Can Predict Your Next Place of Visit

Arun Chauhan,[1] Ravi Tejwani,[2] and Durga Toshniwal[1]

[1]*Computer Science, Indian Institute of Technology, Roorkee 247667, India*
[2]*IBM Watson, Cambridge, MA 02142, USA*

Correspondence should be addressed to Arun Chauhan; aruntakhur@gmail.com

Received 25 September 2015; Revised 19 January 2016; Accepted 26 January 2016

Academic Editor: Antonio J. Jara

The present work focuses on predicting users' next place of visit using their past tweets. We hypothesize that tweets of the person have predictive power on his location and therefore can be used to predict his next place of visit. This problem is important for location based advertising and recommender based services. To predict the next place of visit, we calculate the probabilities of visiting different types of places using bank of binary classifiers and Markov models. More specifically, we train bank of binary classifiers on past tweets and calculated the probabilities of visiting next places. Since bank of binary classifiers is based on a bag-of-words model, to account for time of last visited place and place itself, we built Markov models for different time duration to calculate probabilities of visiting next place. Empirical evaluation shows that by combining the probabilities obtained from bank of binary classifiers and Markov models the accuracy of predicting next place increased from 65% to 80%.

1. Introduction

How the life would be if we know the location of desired and intended place around us anywhere in the world. Though it seems to be superficial, the development in wireless and location acquisition technology helps us to build systems that solve this problem to some extent. Because of location acquisition technology, nowadays, it is not difficult to find out the places in given proximity. The bigger challenge is in predicting places for user according to his need and intent in given proximity. The existence of microblogging services like Twitter and Facebook provides a means to know the person's values and needs in better way.

People share their thoughts and happening on Twitter along with mundane information. In recent few years, the popularity of Twitter is growing exponentially. As of now, at the time of writing this paper the number of users on Twitter are 300 million producing 500 million (https://about.twitter.com/company) tweets per day. With mundane information, people also share their activities like what they are planning, visited places, experiences of visited places, with whom they are going, how they are feeling, and so forth. Availability of such valuable information motivates us to build a system that can predict the future places of visits for the user according to their intent and behavior.

In this paper, we hypothesize that future places of visit can be predicted considering two important factors that are (i) previous visited place and (ii) recent tweets. What people write on their timeline reflects their intent and need, which have a significant role in deciding the next activity to be performed. Similarly, previous visited place also has a major role in deciding the next place to be visited. For example, after having lunch people have coffee. Based on given hypothesis we propose a novel approach for predicting next places of visit. We use bank of binary classifiers (BBC) for predicting the probability of visiting next place using recent tweets only. And to account for time of last visited place and place itself, we build Markov models (MMs) for predicting probabilities. Both BBC and MMs compute the probability of visiting next place independently with high accuracy. We show that by combining these two probabilities we can further improve the prediction accuracy. In attaining the goal of this work, we also proposed two algorithms for tagging the tweets with location and for finding out the optimum number of past tweets used for prediction. Our approach consists of the following steps: (a) assigning location to tweets if relevant information is

presents; (b) extracting features from past tweets that are used to train the models; (c) building models using bank of binary classifiers; (d) using contextual information to enhance the accuracy of predicting the next place. Here contextual information is time of previous visited place and place itself.

We crawl more than 4600 Twitter timelines for exhaustive experimentation. Ground truths are extracted from these timelines using Google Places API and by analyzing tweets for mentions of visited places. We extract features from users timelines by using our proposed algorithm for finding out the optimum length of past tweets for prediction. We build models and do performance analysis. Our best model yields 80% of accuracy for top 5 predictions. We evaluate our approach on new users also and show that performances of models are similar to the seen users to model. The major contribution of this works is as follows:

(1) Building generic models for predicting future places of visit using recent tweets only.

(2) Building Markov models for predicting next place to be visited given previous place visited and time since it is visited.

(3) Ensemble of both models to enhance the model accuracy in predicting next place.

The remainder of the paper is organized as follows. Section 2 presents the proposed approach in detail for predicting next place of visit. Next, Section 3 discusses the experiments and analysis of results. In Section 4 we review relevant literature work. Finally, Section 5 concludes the paper.

2. Proposed Approach

There are two points we want to mention before explaining the 4 steps of proposed approach. Firstly, we build the generic prediction model that captures the relationship between vocabulary used in past tweets and visited places, without considering user's demographics. The main advantage of this approach is that we do not need individual specific training data for predicting their next location.

Secondly, our focus would be the category of the establishment like restaurant, supermarket, pub, and gym, rather than the specific establishment. By doing this, both establishment owners and users can get mutual benefits. For example, if proposed system is predicting restaurant in given spatial proximity, then all the owners of restaurants in the proximity of user's location can approach user with their available offers for promotions. Along with this, the user can also have the option to choose place according to his own suitable interest.

Four steps of our proposed approach are as follows.

2.1. Assigning Location to Tweets. From here onwards we are using location and place of visit interchangeably. Assigning location to tweet is two-step process. First, filter all those tweets having geocoordinates and location information. For location information, we use a regular expression ("I'm at" or "@"). Then by using Google Places API [1] (GPA), get all the places around geocoordinates of the given tweet. If place name present in a tweet is present among places returned by GPA, then that tweet is labeled by the name and categories of location. We want to mention that GPA also returns categories of places which may be more than one. For matching the place name, we extract three words after regular expression from a tweet and find out the ratio r, with total words in each place name returned by GPA. If $r > 0.75$, we label the tweet with the first match among GPA places.

Table 1: Toy Dataset having five instances and four different categories.

Past Tweets	Categories visited
P1	C_2, C_3
P2	C_1, C_3, C_4
P3	C_4
P4	C_2, C_3
P5	C_1, C_4

2.2. Extracted Features. For predicting next place of visit, we use past tweets to infer the intent and interest of the user. We analyze from the ground truths recovered from timelines that people post activity related text before performing that activity. For example, user tweeted, "*We are planning to go for some fun*" before visiting the central park in New York. The time interval of activity related post may vary from weeks to hours based on the activity to be performed. For example, user interested in cricket match going to happen next month has already started tweeting about the event whereas a user interested in going to a restaurant in the evening tweets just a few hours before visiting restaurant. Therefore, we built independent binary classifiers for each category using appropriate size of the window of past tweets. Here window stands for the time window of past tweets that is to be taken to form feature vectors. The window size is found empirically for each classifier/category independently and explained in next section.

To form feature vectors, first we label past tweets (concatenation of past tweets in given window) with the categories of location, visited by the user, just after posting these past tweets. Note that the location which user has visited can have more than one category, and, therefore, the labels of past tweets may be more than one. It is worth mentioning here that while concatenating past tweets, we are not considering location tweet (tweet that has location information). To build a binary classifier for category C_i corresponding data set D_i is formed. To form D_i, those past tweets which are labeled by C_i are treated as positive samples and remaining as negative samples. Hence for training 100 (total number of categories) binary classifiers, we have 100 corresponding data sets. For example, toy data set in Table 1 is having four different categories. Table 2 shows four binary data sets (D_i) for four categories, respectively, as explained above.

2.3. Bank of Binary Classifiers (BBC) for Predicting Future Location. For each category, we build independent binary classifier and the size of the window of past tweets for each

Table 2: Four binary datasets constructed for each category respectively using dataset given in Table 1. Here – denotes the negative samples.

(a) Data Set D_1 for category C_1

Past Tweets	Categories Visited
P1	$-C_1$
P2	C_1
P3	$-C_1$
P4	$-C_1$
P5	C_1

(b) Data Set D_2 for category C_2

Past Tweets	Categories Visited
P1	C_2
P2	$-C_2$
P3	$-C_2$
P4	C_2
P5	$-C_2$

(c) Data Set D_3 for category C_3

Past Tweets	Categories Visited
P1	C_3
P2	C_3
P3	$-C_3$
P4	C_3
P5	$-C_3$

(d) Data Set D_4 for category C_4

Past Tweets	Categories Visited
P1	$-C_4$
P2	C_4
P3	C_4
P4	$-C_4$
P5	C_4

classifier is determined empirically. The steps for determining the window size of past tweets for category C_i are given in Algorithm 1.

In Algorithm 1, for each category, we vary window size w (in hours) from 6 hours to 600 hours to construct feature vectors. Then, for each window size w, we form feature vectors that are used to train the binary classifier. Finally, the window size of a particular category is set, based on the best performance of classifier among all window sizes. For constructing binary data set of each category, function $getBinaryDataSet(D, C_i, w)$ is called with parameters that are timelines of all users (D), name of category (C_i), and size of window of past tweets in hours (w). For all location tweets available on all timelines in D, we form feature vectors by concatenating past tweets in w hours before location tweet (excluding location tweet) and labeled these feature vectors with categories of the location visited by the user in location tweet. Those feature vectors having label C_i are denoted by the "positive" value of the class attribute in the binary data set and remaining feature vectors by "negative" value of the class attribute, as shown in Table 4. Hence, this function returns the binary data set D_i for category C_i. Then binary model H_i is built using returned data set D_i for category C_i and accuracy of models are calculated on validation data set to perceive the performance of model on given window size (w). Similarly, model performance is evaluated on different window sizes and based on the best performance of model, respective window size of past tweets for that category is set. This optimum window size of category C_i is denoted by W_i. Once the window size of each category is determined, we use the respective window size in constructing feature vectors for that category. These feature vectors are used in training and prediction.

Figure 1 shows the block diagram of prediction of places of visit using past tweets. From past tweets we form feature vector FV(C_i) using window size (W_i) for ith category determined in Algorithm 1. Then FV(C_i) is used to infer the probability of visiting the category C_i. Please note that while predicting the probability of visiting category C_i, we use past tweets depending on W_i for H_i classifier at prediction time, as it is hypothesized that history of activity related tweets depends on activity to be performed. Therefore, this probability is denoted by $P(C_i \mid W_i)$. For our purpose, we use Naive Bayes as binary classifier H_i in training and testing.

2.4. Considering Previous Visited Place and Time. We consider two other important factors in predicting next place of visit that are previously visited place and amount of time before it is visited (i.e., time duration) along with the recent tweets. For example, predicting next place of visit as restaurant, after user has visited restaurant only, within an hour, is not considered as a good prediction. For our purpose, to track the duration of time between visits of next and previous place, we discretize the time into one-hour slots. We capture the human behavior of visit in a very simplified manner by building Markov chain for different time duration. Therefore, for each time duration, we form Markov chain and transition matrix. In transition matrix t_h, entry $t_h(i, j)$ represents the probability of visiting place j after place i between time interval $[h, h + 1)$ hours. Let $n_h(i, j)$ be the number of times users' visited place j after place i between time interval $[h, h + 1)$ hours; then

$$t_h(i, j) = P_h\left(C_j \mid C_i\right) = \frac{n_h(i, j)}{\sum_k n_h(i, k)}. \tag{1}$$

We form transition matrix for each time duration (hours) in set $\{1, 3, 6, 12, 24\}$. The probability of visiting next category C_i after h hours given previous visited location having categories $C_aC_b \cdots C_x (\equiv C_{ab\cdots x})$ (C_1 is name of category whereas C_a is variable that can be any category. E.g., C_aC_b can be $C_{22}C_3$.) is calculated as follows. Recall that visited location may have more than one category:

$$P_h\left(C_i \mid C_{ab\cdots x}\right) = \frac{P_h\left(C_{ab\cdots x} \mid C_i\right) P\left(C_i\right)}{P\left(C_{ab\cdots x}\right)}. \tag{2}$$

Input: D: Timelines of all users after labeling, as described in Section 2.1.
Output: Window size of each ith category that is W_i.
(1) **foreach** *category* C_i *in* D **do**
(2) $W_i = 6$ hours;
(3) $\text{Acc}_i = 0$;
(4) $w = 0$;
(5) **repeat**
(6) $w = w + 6$;
(7) $D_i = \texttt{getBinaryDataSet}(D, C_i, w)$;
(8) train binary classifier H_i using data set D_i;
(9) calculate accuracy A_i, of classifier H_i on validation data set;
(10) **if** $A_i > \text{Acc}_i$ **then**
(11) $\text{Acc}_i = A_i$;
(12) $W_i = w$;
(13) **end**
(14) **until** $w \leq 600$;
(15) **end**

(1) $\texttt{getBinaryDataSet}(D, C_i, w)$;
(2) For each location tweet in D, concatenate past tweets in w hours, excluding location tweet;
(3) Concatenated tweets are feature vectors labeled by the categories of location;
(4) Feature vectors labeled with C_i are consider as positive samples and the rest as negative sample;
(5) **return** D_i
(6) **end**

Algorithm 1: For determining the window size of past tweets use to construct feature vectors of each category.

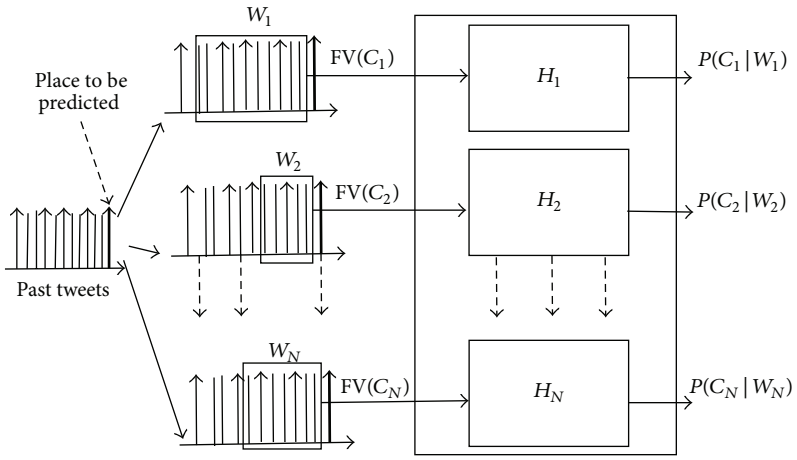


Figure 1: Predicting probability of visiting category C_i given past tweets in window size W_i.

Considering independence assumption between previous visited categories we can write (2) as

$$P_h(C_i \mid C_{ab\cdots x}) = \frac{P_h(C_a \mid C_i) P_h(C_b \mid C_i) \cdots P_h(C_x \mid C_i) P(C_i)}{P(C_a) P(C_b) \cdots P(C_x)} \tag{3}$$

$$= \frac{(P_h(C_i \mid C_a) P(C_a)/P(C_i))(P_h(C_i \mid C_b) P(C_b)/P(C_i)) \cdots (P_h(C_i \mid C_x) P(C_x)/P(C_i)) P(C_i)}{P(C_a) P(C_b) \cdots P(C_x)}. \tag{4}$$

After simplifying, we can write (4) as

$$P_h(C_i \mid C_{ab\cdots x}) = \frac{P_h(C_i \mid C_a)\, P_h(C_i \mid C_b) \cdots P_h(C_i \mid C_x)}{P(C_i)^{|\{ab\cdots x\}|-1}}, \quad (5)$$

where $P_h(C_j \mid C_i)$ are conditional probabilities estimated in (1) and $P(C_i)$ is estimated as follows:

$$P(C_i) = \frac{n(C_i)}{\sum_k n(C_k)}. \quad (6)$$

$n(C_i)$ represents the number of times user visited the category C_i in the training set. Equation (5) uses information of both factors that are previously visited place and how much time before it is visited by picking the appropriate transition matrix. We refer to this Markov modeling which considers time information also as MMs for the sake of brevity.

In order to predict the next place of visit using past tweets and amount of time before which categories are visited by user, we combine these two probabilities that are $P(C_i \mid C_{ab\cdots x})$ (estimated in (5)) and $P(C_i \mid W_i)$ (refer to Figure 1) under the assumption of independence. We refer to the combined classifier $P(C_i \mid W_i, C_{ab\cdots x})$ as CC for the sake of brevity. Consider

$$P(C_i \mid W_i, C_{ab\cdots x}) = \frac{P(W_i, C_{ab\cdots x} \mid C_i)\, P(C_i)}{P(W_i, C_{ab\cdots x})}. \quad (7)$$

As W_i and $C_{ab\cdots x}$ are assumed to be independent, therefore we can write (7) as follows:

$$P(C_i \mid W_i, C_{ab\cdots x}) = \frac{P(W_i \mid C_i)\, P(C_{ab\cdots x} \mid C_i)\, P(C_i)}{P(W_i)\, P(C_{ab\cdots x})}, \quad (8)$$

$$P(C_i \mid W_i, C_{ab\cdots x}) = \frac{\left(P(C_i \mid W_i)\, P(W_i)/P(C_i)\right)\left(P(C_i \mid C_{ab\cdots x})\, P(C_{ab\cdots x})/P(C_i)\right) P(C_i)}{P(W_i)\, P(C_{ab\cdots x})}; \quad (9)$$

after simplifying (9)

$$P(C_i \mid W_i, C_{ab\cdots x}) = \frac{P(C_i \mid W_i)\, P(C_i \mid C_{ab\cdots x})}{P(C_i)}. \quad (10)$$

3. Experiments

We have done exhaustive experiments on the data set crawled from Twitter using publicly available Twitter API. We have collected 4606 users' timelines that have tweeted at least once from New York between April 24, 2014, and April 29, 2014. Each timeline contains approximately 3200 recent tweets. Also, the tweet rate of these users is at least 20 tweets per day. Among 4606 users the number of location tweets on timelines is highly variable. Though we have started from New York the locations visited by users in our data set are around the world.

3.1. Data Sets. For evaluation, we divide our data set into two subsets according to the number of locations on user's timeline. Therefore, among 4606 users, those who have more than 60 location tweets (tweet that has location information) on their Twitter timeline are considered in the first subset named Data Set 1 and the remaining users in the second subset named Data Set 2. Table 3 has shown both data sets mentioned.

Training Data Set (TDS). From Data Set 1, we use oldest $n - 50$ location tweets of each user for training the binary models, where n is total number of location tweets available on their timeline. Tweets on timelines are ordered in reverse chronological order.

Testing Data Set. For evaluating the performance of models on both seen and unseen users, we use two different sets for testing, named as Test Set 1 and Test Set 2.

(a) Test Set 1 (TS1): we use the remaining latest 50 location tweets of each user from Data Set 1 to form feature vectors.

(b) Test Set 2 (TS2): we use all location tweets from Data Set 2 to form feature vectors.

3.2. Evaluation. First we present methods to compute the accuracies of proposed models one by one in the following subsections and then propose few baselines for comparing our proposed models.

3.2.1. Using Markov Models (MMs) Only . We form transition matrix t_h from training data set, where $h \in \{1, 3, 6, 12, 24\}$ in hours, as explained in (1). By using the ground truth of test sets, we compute the accuracy of model as follows:

(1) For every test instance, first we find out both time duration and categories the user has visited just before prediction. Here test instance is the location of a tweet which we want to predict.

(2) Duration is then discretized to integer value h, such that if duration lies within time interval $[h, h+1)$ then transition matrix t_h is used for computation in step (3).

(3) If previously visited categories are $C_{ab\cdots x}$, then by using t_h in (5) we estimate the probability of visiting each category present in training data set.

Table 3: Division of users according to number of location tweets on their timelines.

Data Set	Number of Users	Number of Location
Data Set 1	1706	>60
Data Set 2	2900	⩽60

Table 4: Description of training and testing data sets.

Data Set	# Samples	Detail
Training Data Set (TDS)	260,423	Oldest $n-50$ location of each user from Data Set 1
Testing Data Set 1 (TS1)	84,470	Latest 50 location of each user from Data Set 1
Testing Data Set 2 (TS2)	56,774	All location of each user from Data Set 2

(4) Considering top N categories in nonascending order in the previous step (3), if we recover the ground truth, then we take this test instance as correctly classified. Hence, the accuracy of the model can be defined as follows:

$$\text{Acc@}N = \frac{M}{K}, \tag{11}$$

where M represent the number of test instances accurately classified and K represents total number of test instances.

3.2.2. Using Bank of Binary Classifiers (BBC). By using the ground truth of test sets, we compute the accuracy of model as follows:

(1) For a given test instance (location tweet) we formed feature vector $\text{FV}(C_i)$ for each category C_i as described in Section 2.3. We compute the probability $P(C_i \mid W_i)$ of visiting the category C_i by using the binary classifier H_i modeled in the training phase.

(2) By considering top N categories according to probabilities predicted by binary classifiers, if we recover the ground truth, then we take this test instance as correctly classified. The accuracy of this proposed classifier is calculated in the same way as mentioned in (11).

3.2.3. Using $(P(C \mid W, C_{ab\cdots x}))$ (CC). The evaluation method is similar to evaluation method illustrated for BBC. For every test instance, we compute the probability of visiting each category by using (10). Considering top N categories in nonascending order, if we recover the ground truth, then we classify this test instance as correctly classified. Accuracies of this model are also calculated in similar fashion as mentioned earlier in (11).

3.2.4. Baselines. Our proposed approach is generic and can be applied to both seen and unseen (new) users. As per our knowledge, there is no such literature available that predicts the future place of visit based on only recently used words and latest visited location without using user's demographics. Hence, for evaluating the performance of models, we proposed the following baselines.

Baseline Model 1 (BM1). In this baseline, we have considered the most frequent check-in category in the training data. Let $n(C_i)$ be the number of times users visited the category C_i in the training set. Then,

$$P(C_i) = \frac{n(C_i)}{\sum_{\forall j} n(C_j)}, \tag{12}$$

where $n(C_i)$ represent the number of times user has visited the category C_i.

Baseline Model 2 (BM2). Markov models are built for predicting the next place of visit based on the latest visited location, say, C_L, by the user. The probability of visiting C_i is estimated by computing a fraction of number of user's visits to C_i after C_L in total number of visits to any category after C_L. For example, the probability of visiting category C_i is estimated as follows:

$$P(C_i) = \frac{n(C_L C_i)}{\sum_j n(C_L C_j)}, \tag{13}$$

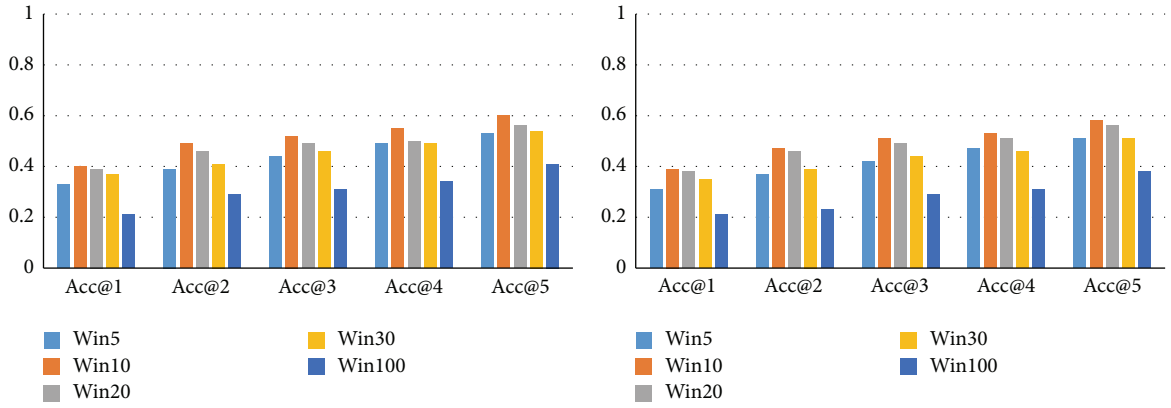
where $n(C_L C_i)$ represents the count of visiting categories C_i after visiting C_L in order.

Here, we want to mention that, in Section 2.4, we form Markov models (MMs) considering amount of time between consecutive visits. Therefore, if we use this baseline model to predict future place, we have only one transition matrix, but in MMs, we have different transition matrix depending upon the granularity of time.

Baseline Model 3 (BM3). As described in Section 2.3, we discuss the need of different window size for each category while predicting the future place of visit. This baseline is defined to show the importance of our approach for deriving window size for each category. For this, we use fixed window size for each category and show the performance against our approach used in Algorithm 1. Similar to BBC in Section 2.3, in this baseline, Naive Bayes is used as binary classifier H_i (see Figure 1), for training and testing. We use five different window sizes that are 5, 10, 20, 30, and 100 hours.

3.3. Results. We conduct two sets of experiments. In the first set, we compare the performance of model BBC proposed in Section 2.3 with Baseline Model 3 (BM3). In the second set of experiments, we compare the performances of proposed models with Baseline Models 1 and 2.

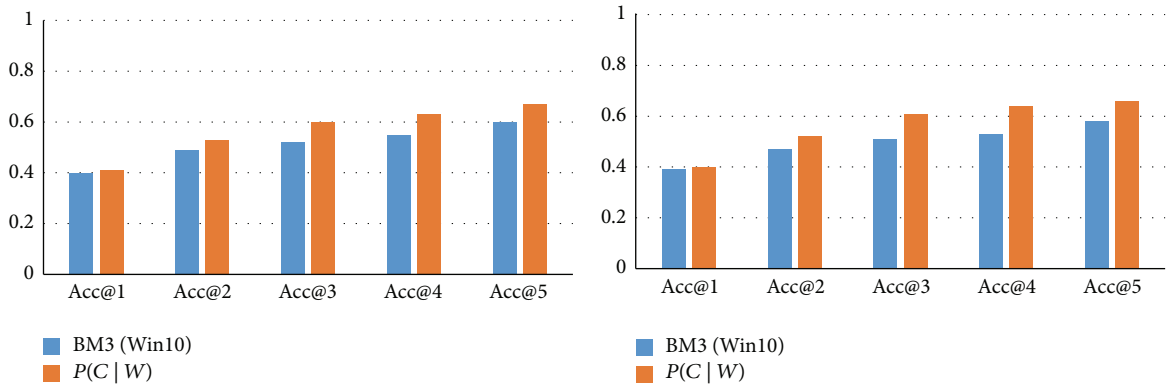
3.3.1. $P(C \mid W)$ versus BM3. The objective of this set of experiments is to show that BBC performs better when feature vectors are constructed by using window sizes derived by Algorithm 1 for training and testing. We want to recall that window size derived for category C_i may not be equal to category C_j, where i and j are some arbitrary category, as window size depends upon the performance of the binary classifier.



(a) Performance of BM3 on Test Set 1 (TS1)

(b) Performance of BM3 on Test Set 2 (TS2)

FIGURE 2: Performance of BM3 on different window sizes for top 5 accuracies. Win$N$ represent the window size of N hours, where $N \in \{5\ 10\ 20\ 30\ 100\}$.



(a) Performance of BM3 and $P(C \mid W)$ on TS1

(b) Performance of BM3 and $P(C \mid W)$ on TS2

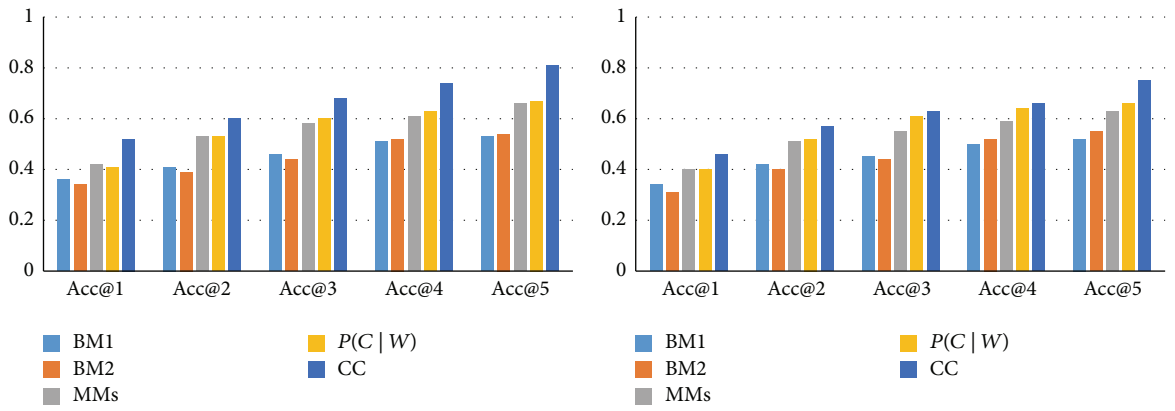
FIGURE 3: Performance of BM3 and $P(C \mid W)$ on top 5 accuracies.

For this, first we show the performance of the classifier on same window size for each category (BM3). One by one, we use five different window sizes that are 5, 10, 20, 30, and 100 hours. Results are shown in Figure 2. Win$N$ represents the performance of BM3 when window size of past tweets is set to N hours. From these results, we observe that BM3 performs better when the size of a window is 10 hours for each category. Also, BM3 performs similarly on both test sets, that is, seen (TS1) and unseen (TS2), which validates that this classifier can be applied to wide variety of users. These results also support our hypothesis that words in tweets posted by users are highly correlated with the future activity or the location of that activity done by that user. By learning the relation of words and locations, BM3 infers the future location using words with high accuracy.

Results shown in Figure 2 validate the existence of the relationship between words and future location. By observing the text in a window, we found out that using same window (BM3) for all categories does not model the problem appropriately. If the window size is 5 hours then the words are very few which results in erroneous predictions. But if we increase the window size, then some words may come which have no role in predicting the current location. For example, tweet like "*now looking for some fun*" tweeted 30 hours before has no significance in deciding the current activity. From the above results, we found optimum window size is 10 hours for BM3.

For appropriate modeling, as discussed in Section 2.2, we derive window size for each category by using Algorithm 1 for training and testing. We compare these two approaches that are BM3 (10 hours) and $P(C \mid W)$. Results in Figure 3 show that we can enhance the accuracy of prediction by considering appropriate duration of tweets' window.

3.3.2. Comparison of Proposed Models with BM1 and BM2. In this section, we compare the proposed models with BM1 and BM2. The performance of our proposed models outperforms both baselines as shown in Figure 4. We can see that MMs perform much better than BM2. The reason of this improvement is that when we use BM2 only, we have only one transition matrix with no time information in it. Thus, the only input to BM2 is previous visited category. But when we use MMs for prediction, we have two input parameters that are previous visited category and the time duration when it is



(a) Results on Test Set 1

(b) Results on Test Set 2

Figure 4: Comparison of proposed models with baselines.

visited. Based on the time duration we pick the corresponding transition matrix for computing predictions. The information of visiting preference according to time is lost in BM2, hence resulting in erroneous predictions. $P(C \mid W)$ is explained in Section 3.3.1 in detail. From accuracies given by MMs and $P(C \mid W)$, we can infer that both recent tweets and previous visited category with duration have almost equal role in predicting next place of visit. But when both these model accuracies are combined under the independent assumption, the prediction rate improves significantly as seen in Figure 4. From this improved accuracy we can infer that the accuracy of proposed model can be enhanced by having more contextual information of the user which helps the model in understanding the user in better way.

4. Related Work

In last decade, researchers have used Twitter data for predicting various things in future. Bollen et al. [2] used tweets for predicting stock market index with high accuracy. Similarly Asur and Huberman [3] predict box-office revenues of movies in advance using related tweets.

Tweets are also explored heavily for inferring users interests for commercial purposes. References [4–6] have used different techniques over tweets for inferring user interest and show that tweets contain a lot of valuable information related to user interest.

For recommendation also people used Twitter data. For example, Sadilek et al. [7] used tweets to recommend those restaurants that user should not go. References [8–10] proposed a recommender system for news recommendations by modeling the user profile and exploiting the tweet-news relationship.

With the exponential growth in usage of smart phones, now users publish millions of tweets frequently from anywhere and at any time. Because of this, one other field emerged that studies the mobility prediction of users; that is, where the user will be? or where he was when given tweets were published? References [11–13] have shown that by using Twitter data we can predict the location of user with high accuracy. But the granularity level of predicting the user location using tweets is at either the country level or regional level. For example, Han et al. [14] predict user location, that is, country or region by identifying location indicative words (i.e., frequent word used at the specific location), in contrast to our approach where we predict locations such as shop, church, and restaurant, where granularity level is very deep.

Some of the efforts have been made for predicting user location such as restaurant and shops, but the data used there are generated by Location Based Social Networks (LBSN). Using LBSN for prediction is very different from using Twitter, as Twitter data is very unstructured and challenging in comparison to structured data of LBSN. References [15–19] have explored LBSN (Foursquare) for recommending or predicting next location to the user.

Some people used text published by users for deriving personality traits and based on common traits recommendations have been made. For deriving personality traits Linguist Inquiry and Word Count (LIWC) [20] had been used frequently. References [20–29] have shown that lexicons used by people can be used for understanding their personal values and how to use these traits for a recommendation. Though all these approaches have been used extensively in analyzing personality traits, these also have shortcomings of predefined word category correlation. Alternatively, Schwartz et al. [30] used rather different approach for using vocabulary known as open-vocabulary technique (using all set of words available on social media) in comparison to closed-vocabulary technique, Linguistic Inquiry and Word Count (LIWC) [20], where some predefined sets of words are used for deriving the personality traits. In their study, they have shown that by using the open-vocabulary approach they got higher state-of-the-art accuracy in predicting gender in comparison to LIWC by exploring latent factors that are not captured by closed-vocabulary approach. Motivated by this approach, we have used all words available on timelines for modeling the users' behavior.

5. Conclusion

In the present work, we study the problem of predicting next place of visit using tweets. We proposed a methodology for

predicting next place of visit for user. For experiments, we crawled more than 4600 users' timelines from Twitter. For modeling and generating ground truths, we labeled those tweets with visited locations where relevant information is present. For labeling tweets, we propose simple pattern matching technique labeling tweets with high accuracy. We have also proposed an algorithm for deriving optimal length of window for each category for deriving feature vectors which is used in training and testing of BBC (bank of binary classifiers). From this trained model, BBC (Naive Bayes), we compute the probabilities of visiting next place. To account for the time of last places visited, we trained Markov models that also compute probabilities of visiting next places. Assuming the independence of probabilities produced by bank of binary classifiers and Markov models, we combined these two probabilities. From experiments, we found out that probabilities of visiting next place increased up to 80% from 65%. This shows that our model can be potentially used in location based advertising, intelligent resource allocation, and so forth. Also, similar performance of proposed model on both seen and unseen data sets make it applicable to wide variety of users.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] "Google Places API," https://developers.google.com/places.

[2] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of Computational Science*, vol. 2, no. 1, pp. 1–8, 2011.

[3] S. Asur and B. A. Huberman, "Predicting the future with social media," in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI '10)*, pp. 492–499, Toronto, Canada, September 2010.

[4] P. Bhattacharya, M. B. Zafar, N. Ganguly, S. Ghosh, and K. P. Gummadi, "Inferring user interests in the twitter social network," in *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*, pp. 357–360, Foster City, Calif, USA, October 2014.

[5] C. Budak, A. Kannan, R. Agrawal, and J. Pedersen, "Inferring user interests from microblogs," Tech. Rep., 2014.

[6] D. Ramasamy, S. Venkateswaran, and U. Madhow, "Inferring user interests from tweet times," in *Proceedings of the 1st ACM Conference on Online Social Networks (COSN '13)*, pp. 235–240, ACM, Boston, Mass, USA, October 2013.

[7] A. Sadilek, S. P. Brennan, H. A. Kautz, and V. Silenzio, "nEmesis: which restaurants should you avoid today?" in *Proceedings of the 1st AAAI Conference on Human Computation and Crowdsourcing (HCOMP '13)*, AAAI, 2013.

[8] N. Jonnalagedda and S. Gauch, "Personalized news recommendation using twitter," in *Proceedings of the IEEE International Conference on Web Intelligence and Intelligent Agent Technology (WIC/ACM '13)*, pp. 21–25, Atlanta, Ga, USA, November 2013.

[9] G. D. F. Morales, A. Gionis, and C. Lucchese, "From chatter to headlines: harnessing the real-time web for personalized news recommendation," in *Proceedings of the 5th ACM International Conference on Web Search and Data Mining (WSDM '12)*, Seattle, Wash, USA, February 2012.

[10] F. Abel, Q. Gao, G.-J. Houben, and K. Tao, "Twitter-based user modeling for news recommendations," in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI '13)*, pp. 2962–2966, AAAI Press, Beijing, China, August 2013.

[11] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. Magnenat-Thalmann, "Who, where, when and what: discover spatio-temporal topics for twitter users," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '13)*, pp. 605–613, Chicago, Ill, USA, August 2013.

[12] M. Lichman and P. Smyth, "Modeling human location data with mixtures of kernel densities," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*, pp. 35–44, ACM, New York, NY, USA, August 2014.

[13] K. Lee, R. K. Ganti, M. Srivatsa, and L. Liu, "When twitter meets foursquare: tweet location prediction using foursquare," in *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MOBIQUITOUS '14)*, pp. 198–207, ICST, London, UK, December 2014.

[14] B. Han, P. Cook, and T. Baldwin, "Textbased twitter user geolocation prediction," *Journal of Artificial Intelligence Research*, vol. 49, pp. 451–500, 2014.

[15] W. Mathew, R. Raposo, and B. Martins, "Predicting future locations with hidden Markov models," in *Proceedings of the ACM Conference on Ubiquitous Computing (UbiComp '12)*, Pittsburgh, Pa, USA, September 2012.

[16] J. Bao, Y. Zheng, and M. F. Mokbel, "Location-based and preference-aware recommendation using sparse geo-social networking data," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12)*, pp. 199–208, ACM, Redondo Beach, Calif, USA, November 2012.

[17] Q. Yuan, G. Cong, and A. Sun, "Graph-based point-of-interest recommendation with geographical and temporal influences," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM '14)*, pp. 659–668, ACM, Shanghai, China, November 2014.

[18] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. Magnenat-Thalmann, "Time-aware point-of-interest recommendation," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '13)*, pp. 363–372, 2013.

[19] H. Gao, J. Tang, X. Hu, and H. Liu, "Exploring temporal effects for location recommendation on location-based social networks," in *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13)*, pp. 93–100, Hong Kong, October 2013.

[20] J. W. Pennebaker, C. K. Chung, M. Ireland, A. Gonzales, and R. J. Booth, *The Development and Psychometric Properties of LIWC2007*, LIWC. Net, Austin, Tex, USA, 2007.

[21] Y. R. Tausczik and J. W. Pennebaker, "The psychological meaning of words: LIWC and computerized text analysis methods," *Journal of Language and Social Psychology*, vol. 29, no. 1, pp. 24–54, 2010.

[22] A. D. I. Kramer and C. K. Chung, "Dimensions of self-expression in facebook status updates," in *Proceedings of the 5th*

International AAAI Conference on Weblogs and Social Media (ICWSM '11), pp. 169–176, Barcelona, Spain, July 2011.

[23] J. Chen, G. Hsieh, J. Mahmud, and J. Nichols, "Understanding individuals' personal values from social media word use," in *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '14)*, pp. 405–414, ACM, San Francisco, Calif, USA, February 2014.

[24] S. A. Golder and M. W. Macy, "Diurnal and seasonal mood vary with work, sleep, and daylength across diverse cultures," *Science*, vol. 333, no. 6051, pp. 1878–1881, 2011.

[25] S. Argamon, M. Konnel, J. W. Pennebaker, and J. Schier, "Mining the blogosphere: age, gender and the varieties of self-expression," *First Monday*, vol. 12, no. 9, 2007.

[26] E. Gilbert, "Phrases that signal workplace hierarchy," in *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW '12)*, pp. 1037–1046, Seattle, Wash, USA, February 2012.

[27] J. Mahmud, M. X. Zhou, N. Megiddo, J. Nichols, and C. Drews, "Recommending targeted strangers from whom to solicit information on social media," in *Proceedings of the International Conference on Intelligent User Interfaces (IUI '13)*, pp. 37–47, ACM, March 2013.

[28] K. Lee, J. Mahmud, J. Chen, M. X. Zhou, and J. Nichols, "Who will retweet this?: automatically identifying and engaging strangers on twitter to spread information," in *Proceedings of the 19th International Conference on Intelligent User Interfaces (IUI '14)*, pp. 247–256, ACM, Haifa, Israel, February 2014.

[29] H. Badenes, M. N. Bengualid, J. Chen et al., "System U: automatically deriving personality traits from social media for people recommendation," in *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*, pp. 373–374, Foster City, Calif, USA, October 2014.

[30] H. A. Schwartz, J. C. Eichstaedt, M. L. Kern et al., "Personality, gender, and age in the language of social media: the open-vocabulary approach," *PLoS ONE*, vol. 8, no. 9, Article ID e73791, 2013.

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2016, Article ID 4127358, 11 pages
http://dx.doi.org/10.1155/2016/4127358

Research Article

Prefetching Scheme for Massive Spatiotemporal Data in a Smart City

Lian Xiong,[1,2] Zhengquan Xu,[1,2] Hao Wang,[1,2] Shan Jia,[1,2] and Li Zhu[3]

[1]*State Key Laboratory of Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430079, China*
[2]*Collaborative Innovation Center for Geospatial Technology, Wuhan 430079, China*
[3]*Hubei Collaborative Innovation Center for High-Efficiency Utilization of Solar Energy, Hubei University of Technology, Wuhan 430068, China*

Correspondence should be addressed to Zhengquan Xu; xuzq@whu.edu.cn

Received 14 September 2015; Accepted 31 December 2015

Academic Editor: Liming Chen

Employing user access patterns to develop a prefetching scheme can effectively improve system I/O performance and reduce user access latency. For massive spatiotemporal data, traditional pattern mining methods fail to directly reflect the spatiotemporal correlation and transition rules of user access, resulting in poor prefetching performance. This paper proposed a prefetching scheme based on spatial-temporal attribute prediction, named STAP. It maps the history of user access requests to the spatiotemporal attribute domain by analyzing the characteristics of spatiotemporal data in a smart city. According to the spatial locality and time stationarity of user access, correlation analysis is performed and variation rules are identified for the history of user access requests. Further, the STAP scheme mines the user access patterns and constructs a predictive function to predict the user's next access request. Experimental results show that the prefetching scheme is simple yet effective; it achieves a prediction accuracy of 84.3% for access requests and reduces the average data access response time by 44.71% compared with the nonprefetching scheme.

1. Introduction

The development of smart cities based on cloud computing and the Internet of Things has generated massive spatiotemporal data, including meteorological data, hydrological data, natural disaster data, and remote-sensing images, with three basic attributes, namely, location, time, and type. Such data are characterized by wide variety, large quantity, high redundancy, and dynamic growth over time. A smart city can quickly and conveniently provide users with rich predefined applications through a network platform based on the users' demands for spatiotemporal data services such as data visualization, spatiotemporal correlation analysis, temporal emergency aid, and massive information retrieval.

Low latency, high concurrency, and high aggregate bandwidth are the three important criteria for measuring the quality of spatiotemporal data services in a smart city. Under the same bandwidth and computing power, the key factor affecting the quality of a spatiotemporal data service is the system delay in the network environment. Prefetching schemes have been widely used because they can effectively improve the data transfer rate and reduce the user access latency [1]. Therefore, it is important to develop an efficient prefetching scheme for improving the quality of spatiotemporal data services in a smart city.

Compared with nonspatiotemporal data, spatiotemporal data not only have three basic spatiotemporal attributes but also have obvious spatiotemporal correlation of user access; moreover, the corresponding prefetching schemes are different. Based on the different types of data, data prefetching schemes can be divided into two categories in the network environment.

(1) Nonspatiotemporal Data Prefetching. Nonspatiotemporal data prefetching mainly concerns web prefetching and personalized recommendations. In general, user access information is obtained by clustering or correlation analysis of webpages or users in order to mine user access patterns and

develop a prefetching scheme. Pallis et al. [2] employed the association rule to filter webpages visited by users in order to perform webpage clustering; then, they used the clustering result sets to develop a prefetching scheme for overcoming the problem of web access latency. Further, Wan et al. [3] used clustering to develop a method based on random indexing with various weight functions in order to track user access and cluster users with similar activity patterns. Khosravi and Tarokh [4] adopted a naive Bayesian approach for dynamic mining of user access patterns in order to predict the pages accessed by users. Bamshad et al. [5] developed a personalized webpage recommendation system by using the a priori algorithm to identify pagesets frequently accessed by users; then, they matched the users' currently accessed pages with the frequently accessed pagesets. Matthews et al. [6] proposed a genetic algorithm based on association rules and discovered extra rules that are complementary to existing algorithms, which can facilitate the development of more effective prefetching schemes. Similar studies have been described in the literature [7–13].

(2) Spatiotemporal Prefetching. Spatiotemporal data prefetching mainly concerns WebGIS. The corresponding prefetching schemes use not only the characteristics of the data but also the spatiotemporal correlation of user access. Typically, the characteristics of the data are used to mine user access patterns and develop a prefetching scheme by spatiotemporal correlation analysis, transition probability calculation, access frequency ranking, and other methods related to user access requests or data. In order to overcome the problem of a long delay when users browse large objects in WebGIS, Park and Kim [14] used the spatial clustering characteristic of the Hilbert curve. They divided the entire geographic space using Hilbert curves and gave them appropriate values; then, they proposed a prefetching method based on the spatial locality of user access. Dong et al. [15] exploited the spatial locality of user access and proposed two prefetching methods. The first method computes transition probabilities between tiles and prefetches the most probable tile; the second method uses a "Neighbor Selection Markov Chain" to compute the objects to be prefetched based on the data of the k tiles previously requested. Considering the previous action of a given user, Yeşilmurat and Işler [16] proposed a heuristic prefetching algorithm that analyzes and ranks the previous moves of a user to predict the user's next move; then, it identifies the locations of candidate tiles to be prefetched. Considering both long-term and short-term popularity features for tile access in a geographic space, Li et al. [17] presented a Markov prefetching model in a cluster-based caching system based on the Zipf distribution and verified that the method has a high prefetch hit rate and a short average response time.

From existing studies, it can be seen that, in the network environment, a typical data prefetching scheme is based on current/historical user access information. It analyzes and processes the information at the level of access requests or data by using user access continuity, spatial locality, popularity, association rules between objects, and other methods in order to mine the user access patterns. Then, it predicts user access requests according to these patterns in order to achieve data prefetching.

However, we note that user access to spatiotemporal data usually has obvious spatiotemporal features in a smart city. The general approach mines user access patterns at the level of access requests or data, the results can only reflect this feature indirectly, and they are not useful for developing a high-efficiency prefetching scheme for massive spatiotemporal data. But if we analyze and process user access information at the level of spatiotemporal attributes, then the hidden spatiotemporal correlation and transition rules can be found, and we can develop a more targeted prefetching scheme. Therefore, how to effectively mine the spatiotemporal features and patterns from the user access information is the focus of this paper.

In this paper, we propose a prefetching scheme, STAP, for massive spatiotemporal data in a smart city. The proposed method analyzes the characteristics of spatiotemporal data and the spatiotemporal correlation of user access, parameterizes the history of user access requests, and extracts the spatiotemporal attributes. Then, it uses regional meshing, association rules, and the autoregressive integrated moving average (ARIMA) model in the spatiotemporal attribute domain to perform correlation analysis and identify transition rules, mines user access patterns, and constructs a predictive function to predict the user's next access request, in order to achieve spatiotemporal data prefetching.

The remainder of this paper is organized as follows. Section 2 introduces the motivation and principle of our prefetching scheme. Section 3 describes the implementation of our prefetching scheme, which mainly involves two steps. The first step shows how to (i) mine the user access patterns from the history of user access requests and (ii) construct the predictive function for predicting requests. The second part explains how to (i) predict the user's next access request according to the current one by using the abovementioned predictive function and (ii) prefetch the corresponding data. Section 4 presents and discusses the performance evaluation results of our prefetching scheme. Finally, Section 5 briefly summarizes our findings and concludes the paper.

2. Principle of Prefetching Scheme

2.1. Motivation. Typical prefetching schemes involve two steps. The first step is to mine the user access patterns and construct the predictive function; the second step is to predict access requests and prefetch the data. The first step is usually based on historical user access information as well as the characteristics of the data. It uses clustering, association rules, Markov models, and other methods to mine associate items accessed by users. Then, it merges them to form associate itemsets or uses mathematical functions to describe the correlation between the associate items. Thus, the corresponding request predictive function is constructed. The second step is based on the current user access request, the predictive function is used to predict the next access request of the user, and then the corresponding data is loaded into the cache.

For instance, suppose that h_i, $1 \le i \le n$, is a user access request; then, the history of user access requests can be expressed as the sequence $H = \langle h_1, h_2, h_3, \ldots, h_n \rangle$. The first step is to mine the access pattern, that is, to mine associate items $h_i \rightarrow h_j$ from H and construct the request predictive function according to the associate itemsets. The second step involves access request prediction and data prefetching. Take the current user access request (h_i, h_j) as the input for the prediction function. Then, scan the associate itemsets to find the matching associate items $(h_i, h_j \rightarrow h_k)$; the output of the function h_k is the predicted access request. Finally, prefetch the corresponding data of request h_k to the cache.

It can be seen that the key aspect of the prefetching scheme is to mine user access patterns and construct the request predictive function. However, unlike ordinary user access features in the network environment, the users obtain spatiotemporal data services based on predefined applications in a smart city, which have obvious spatiotemporal correlation. For example, if a user checks the weather conditions by predefined applications, the access data are current time and location related meteorological data; when searching for the nearby living facilities, the access data, such as restaurants and parking lots, are closely related to the user's current location. Therefore, if we treat the access request as a whole for direct mining as in the case of traditional mining patterns, the spatiotemporal correlation of user access will not be reflected directly, and the corresponding prediction function will not predict user access requests accurately.

In order to overcome the inherent drawback of mining user access patterns directly at the level of access requests, we start with the characteristics of spatiotemporal data and the spatiotemporal correlation of user access. Then, we mine the user access patterns at the level of spatiotemporal attributes and construct the access request predictive function.

2.2. Principle. Suppose that the history of user access requests in a smart city can be expressed as the sequence $A = \langle a_1, a_2, a_3, \ldots, a_n \rangle$, where each a_i, $1 \le i \le n$, contains the following information: location attribute p, type attribute s, time attribute t, user IP, and session time. In order to analyze and process the access sequences at the level of the spatiotemporal domain, we parameterize the information and extract the spatiotemporal attributes to form spatiotemporal attribute sequences:

$$\begin{aligned} A &= \langle (p_1, s_1, t_1), (p_2, s_2, t_2), \ldots, (p_n, s_n, t_n) \rangle \\ &= \{P_n, S_n, T_n\}, \end{aligned} \tag{1}$$

where $a_i = (p_i, s_i, t_i)$ represents a parameterized request with the extraction results of the spatiotemporal attributes. Specifically, $P_n = \langle p_1, p_2, p_3, \ldots, p_n \rangle$ represents the sequence of location attributes, $S_n = \langle s_1, s_2, s_3, \ldots, s_n \rangle$ represents the sequence of type attributes, and $T_n = \langle t_1, t_2, t_3, \ldots, t_n \rangle$ represents the sequence of time attributes.

Because the spatiotemporal attribute sequences $\{P_n, S_n, T_n\}$ contain three types of spatiotemporal attributes, it is extremely difficult to find the hidden spatiotemporal correlations and variation rules. However, we observe that, in a smart city, when most users request access to spatiotemporal data, the spatiotemporal attributes of the request have strong self-correlation but weak cross-correlation. That is to say, any two consecutive access requests, a_i, a_{i+1}, have weak correlation between the location attribute p_i and the type attribute s_{i+1} but very strong correlation between p_i and p_{i+1}. For example, when a user checks the current temperature of regional A, there is a huge possibility that he will further query the wind speed, PM2.5 of region A, rather than the water quality of other regions.

Therefore, to simplify access pattern mining, we process spatiotemporal attribute sequences $\{P_n, S_n, T_n\}$ based on the self-correlation and cross-correlation of the spatiotemporal attributes of the access requests, in order to construct the access request predictive function. The specific steps are as follows:

(1) For access requests with self-correlation of spatiotemporal attributes, we analyze the self-correlation of the spatiotemporal attribute sequences to mine associate items $p_i \rightarrow p_j$, $s_i \rightarrow s_j$, and $t_i \rightarrow t_j$. Then, we construct the independent attribute prediction function $\mathrm{Pre}'(p, s, t) = \{\mathrm{Pre}(p), \mathrm{Pre}(s), \mathrm{Pre}(t)\}$, where $\mathrm{Pre}(p)$ represents the location attribute predictive function, $\mathrm{Pre}(s)$ represents the type attribute predictive function, and $\mathrm{Pre}(t)$ represents the time attribute predictive function.

(2) For access requests with cross-correlation of spatiotemporal attributes, we carry out cross-correlation analysis of the spatiotemporal attribute sequences, and we mine associate items $(p_i, s_i, t_i) \rightarrow (p_j, s_j, t_j)$. Then, we construct the conjoint attribute prediction function $\mathrm{Pre}''(p, s, t)$.

3. Implementation

The method is implemented in two steps. The first step is the offline mining of user access patterns to construct the predictive function, and the second step is the online access request prediction and data prefetching.

3.1. Construction of Predictive Function. The predictive function consists of the independent attribute prediction function $\mathrm{Pre}'(p, s, t) = \{\mathrm{Pre}(p), \mathrm{Pre}(s), \mathrm{Pre}(t)\}$ and the conjoint attribute prediction function $\mathrm{Pre}''(p, s, t)$.

3.1.1. Construction of Independent Attribute Predictive Function

(1) Construction of Location Attribute Predictive Function. The key aspect of the location attribute predictive function $\mathrm{Pre}(p)$ is the correlation of access requests in the spatial domain. Therefore, we can use the association rule algorithm [18] to mine associate items $p_i \rightarrow p_j$ from the sequence of location attributes $P_n = \langle p_1, p_2, p_3, \ldots, p_n \rangle$ and construct

the location attribute predictive function according to the associate rulesets.

(a) Regional Meshing. The location attribute of spatiotemporal data represents the geographical location of a data source in a smart city, usually expressed by latitude and longitude coordinates $p = (x, y)$. However, solving the association rules of the location attribute coordinate points directly requires numerous calculations. Moreover, updating, modification, addition, or deletion operations on location attributes will require recalculation for the entire area. Therefore, we exploit regional meshing for the entire area, which allows for both the early solution of rules and late update of associate rules in the cell area, thereby providing partial and incremental solution of association rules and reducing the computation considerably.

Suppose that the geographic area is a two-dimensional Euclidean rectangular space $[0, X][0, Y]$ in a smart city. We divide it into row × col rectangular cells with coding, where the code of the area covered by the ith row jth column is $g_{ij} = j + \text{col} \cdot (i-1)$. Then, for any location attribute coordinate point $p_k = (x_k, y_k)$ in the geographic area, we assume that it belongs to the cell g_{ij} if it satisfies the following equation:

$$\begin{aligned} (i-1)\frac{X}{\text{row}} \le x_k \le i\frac{X}{\text{row}}, \quad 1 \le i \le \text{row} \\ (j-1)\frac{Y}{\text{col}} \le y_k \le j\frac{Y}{\text{col}}, \quad 1 \le j \le \text{col}. \end{aligned} \tag{2}$$

Figure 1(a) shows the geographic rectangular area divided into 4 × 5 cells and the meshing cell coding. Figure 1(b) shows all the neighbor cells of the cell g_{ij}.

(b) Construction of Predictive Function. Through regional meshing, we can use an association rule algorithm to mine the associate items of each cell from the sequence of location attributes $P_n = \langle p_1, p_2, p_3, \ldots, p_n \rangle$ and construct the location attribute predictive function $\text{Pre}(p)$ according to the associate rulesets. The specific steps are as follows:

(1) Calculate the location coordinate sets $p_{g_{ij}} = \{p_{g_1}, p_{g_2}, \ldots, p_{g_m}, \ldots, p_{g_n}\}$ contained in the cell g_{ij} and its neighbor cells, as shown in Figure 1(b).

(2) Count the number of times every coordinate point $p_{g_i}, p_{g_i} \in p_{g_{ij}}$, appears in the sequence of location attributes, that is, *support*, and compare it with the predefined *support threshold* δ_p to find frequent 1-itemsets. By looping through the location attribute sequence via the connection and cut between the frequent itemsets, we can find frequent 2-itemsets, frequent 3-itemsets, and so on until frequent m-, $2 \le m \le n$, itemsets.

(3) Calculate the *confidence* of each frequent m-itemsets and its subset frequent m-1-itemsets. Generate association rules $(p_{g_i}, p_{g_{i+1}}, \ldots, p_{g_{i+m-1}}) \rightarrow \langle p_{g_{i+m}}, \phi_{g_i g_{i+m}} \rangle$ on those associate itemsets whose *confidence* values are greater than the *confidence threshold* ϕ_p. Then, form the association rulesets $R(g_{ij}, \phi_{ij}) = \bigcup_m ((p_{g_i}, p_{g_{i+1}}, \ldots, p_{g_{i+m-1}}) \rightarrow \langle p_{g_{i+m}}, \phi_{g_i g_{i+m}} \rangle)$ of the cell g_{ij}.

(4) Loop through each cell in the geographical area to calculate the location attribute association rulesets and merge them to form the associate rulesets $R(p_{ij}, \phi_{ij}) = \bigcup_{i,j} R(g_{ij}, \phi_{ij})$ of the entire geographic area. Then, construct the location attribute predictive function,

$$\text{Pre}(p) = \text{Match}\left(p, R\left(p_{ij}, \phi_{ij}\right)\right), \tag{3}$$

where $\text{Match}(\cdot)$ is a rule matching function, whose output is an associate rule matching successfully with location attribute p; the specific methods are as in Section 3.2.1.

(2) Construction of Type Attribute Predictive Function. The key aspect of the type attribute predictive function $\text{Pre}(s)$ is the correlation of access requests in the type domain. Therefore, we can use the association rule algorithm to mine associate items $s_i \rightarrow s_j$ from the sequence of type attributes $S_n = \langle s_1, s_2, s_3, \ldots, s_n \rangle$ and construct the type attribute predictive function according to the associate rulesets. The specific steps are as follows:

(1) Count the number of times every $s_i, s_i \in S$, appears in the sequence of type attributes, that is, *support*, and compare it with the predefined *support threshold* δ_s to find frequent 1-itemsets. By looping through the type attribute sequences via the connection and cut between the frequent itemsets, we can find frequent 2-itemsets, frequent 3-itemsets, and so on until frequent m-, $2 \le m \le n$, itemsets.

(2) Calculate the *confidence* of each frequent m-itemsets and its subset frequent m-1-itemsets. Generate association rules $(s_i, s_{i+1}, \ldots, s_{i+m-1}) \rightarrow (s_{i+m}, \phi_{i,i+m})$ on those associate itemsets whose *confidence* values are greater than the *confidence threshold* ϕ_s, and form the association rulesets $R(s_{ij}, \phi_{ij}) = \bigcup_m ((s_i, s_{i+1}, \ldots, s_{i+m-1}) \rightarrow (s_{i+m}, \phi_{i,i+m}))$. Then, construct the type attribute predictive function

$$\text{Pre}(s) = \text{Match}\left(s, R\left(s_{ij}, \phi_{ij}\right)\right). \tag{4}$$

(3) Construction of Time Attribute Predictive Function. The key aspect of the time attribute predictive function $\text{Pre}(t)$ is the correlation of access requests in the time domain. Therefore, we can analyze the sequence of time attributes $T_n = \langle t_1, t_2, t_3, \ldots, t_n \rangle$, to develop a model to describe this underlying correlation.

The time attribute sequence of user access requests is a typical nonstationary sequence influenced by a predefined application, which has obvious trends in a local range. ARIMA is an important and widely used short-term time series prediction model. It can predict future values according to the current and historical values of the sequence, but it requires the sequence to be stationary [19–23]. To this end, we can piecewise represent the time attribute sequence and

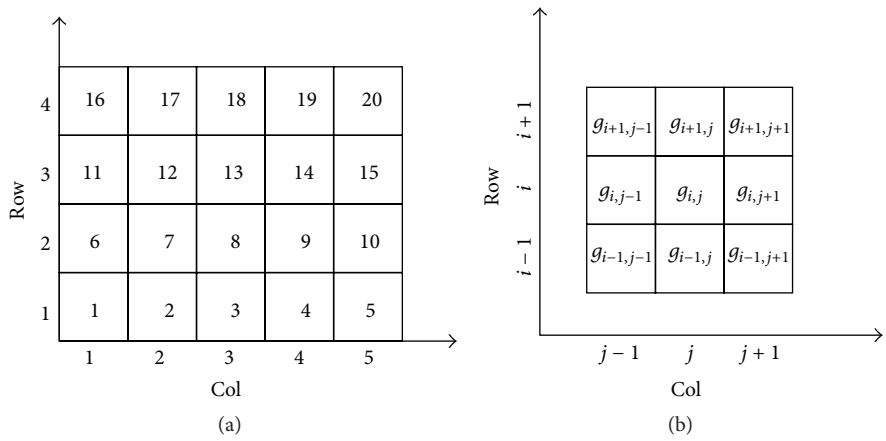


FIGURE 1: Regional meshing and coding: (a) meshing cell coding; (b) neighbor cells of g_{ij}.

perform difference processing to achieve local stationarity. Then, we build the ARIMA model and construct the time attribute predictive function.

(a) Piecewise Representation of Time Attribute Sequence. We use extreme point detection based on the slope change and piecewise representation of the time attribute sequence according to local extreme values in the sequence (the beginning and end value of each curve). The method calculates the slope difference $||t_i - t_{i-1}| - |t_{i+1} - t_i||/\Delta T$ of the line segment formed by sequence value t_i, $1 < i < n$, and its neighbor points t_{i-1}, t_{i+1}, where ΔT is the time interval of the access request. Then, we compare the slope difference with a predefined threshold. If it is greater than or equal to the predefined threshold, we assume that t_i is a local extremum. Finally, by using local extrema, we can piecewise represent the time attribute sequence as follows:

$$T = \{(t_{1L}, t_{1R}), (t_{2L}, t_{2R}), \ldots, (t_{kL}, t_{kR})\}, \quad (5)$$

where t_{iL} is the starting value of $i, i \in k$, segment, t_{iR} is the end value of $i, i \in k$, piecewise, and k is the number of piecewise segments.

(b) Construction of Predictive Function. With the abovementioned piecewise representation and difference processing, we can realize local stationarity of the time attribute sequence and build ARIMA to construct the time attribute predictive function Pre(t). By introducing the k-step lag operator $B^k t_{h_n} = t_{h_n - k}$ and d-order difference $w_n = \Delta^d t_{h_n} = (1 - B)^d t_{h_n}$, $d = 0, 1, 2$, the standard ARIMA(p, d, q) model can be expressed as follows:

$$w_n = \varphi_1 w_{n-1} + \varphi_2 w_{n-2} + \cdots + \varphi_p w_{n-p} + \delta + u_t + \theta_1 u_{t-1} + \theta_2 u_{t-2} + \cdots + \theta_q u_{t-q}, \quad (6)$$

where $w_n = \Delta^d t_n = (1 - B)^d t_n$ is the difference order, $\varphi_1, \varphi_2, \ldots, \varphi_p$ are the autoregressive parameters, $\theta_1, \theta_2, \ldots, \theta_q$ are the moving average parameters, δ is a constant that indicates that the sequence is nonzero mean, and u_t is white noise sequence.

Suppose that the right-and-left local extreme value of j, $1 < j < k$, piecewise is $t_{jL} = t_m$, $t_{jR} = t_n$. Then, according to formula (5), we can piecewise represent it as $(t_{jL}, t_{jR}) = t_m, t_{m+1}, \ldots, t_n$, and, through d-order difference processing, it can be stationary. Judging from the fact that the user access is restricted by the predefined application, the change trend of the time attribute sequence can be only linear and regular, which means that it remains unchanged in terms of cycle and step size, so the parameters are $p = 1$, $\varphi_1 = 1$. At the same time, the time attribute sequence of access request is not affected by external random interference, so the parameters are $u_t = 0$, $q = 0$.

Finally, we can build ARIMA$(1, d, 0)$ as $w_n = w_{n-1}$. Combined with the lag operator $B^k t_{h_n} = t_{h_n - k}$ and d-order difference, the time attribute predictive function Pre(t) can be expressed as follows:

$$\mathrm{Pre}(t_n) = \begin{cases} t_{n-1}, & d = 0 \\ 2t_{n-1} - t_{n-2}, & d = 1 \\ 3t_{n-1} - 3t_{n-2} + t_{n-3}, & d = 2. \end{cases} \quad (7)$$

3.1.2. Construction of Conjoint Attribute Predictive Function. Because the access requests with cross-correlation of spatiotemporal attributes account for a very small proportion of the total number of requests, it is difficult to jointly analyze the attributes. Therefore, we analyze only the access requests that have special cross-correlation of spatiotemporal attributes, that is, if the sequence of location attributes $P_l = \langle p_i, p_{i+1}, \ldots, p_{i+l} \rangle$ and the sequence of type attributes $S_l = \langle s_{i+1}, s_{i+2}, \ldots, s_{i+l} \rangle$ remain unchanged and the length reaches the minimum threshold $l = 3$, as expressed by the following equation:

$$p_{i+1} = p_{i+2} = \cdots = p_{i+l}, \quad s_{i+1} = s_{i+2} = \cdots = s_{i+l}, \quad l \geq 3. \quad (8)$$

Then, we assume that the location attribute and the type attribute remain unchanged in the next access request, and

the time attribute can be predicted by $\mathrm{Pre}(t)$. Finally, the conjoint attribute predictive function $\mathrm{Pre}''(p,s,t)$ can be constructed as

$$\mathrm{Pre}''(p_n,s_n,t_n) = \begin{cases} (p_{n-1},s_{n-1},t_{n-1}), & d=0 \\ (p_{n-1},s_{n-1},2t_{n-1}-t_{n-2}), & d=1 \\ (p_{n-1},s_{n-1},3t_{n-1}-3t_{n-2}+t_{n-3}), & d=2. \end{cases} \quad (9)$$

3.2. Prediction Method for Access Requests. The purpose of this section is to show how to use the predictive function to predict the user's next access request based on the current one.

Suppose that the current user access request can be expressed as the sequence $B = \langle b_1, b_2, b_3, \ldots, b_m\rangle$, and spatiotemporal attributes sequences are $B = \langle (p_1,s_1,t_1),(p_2,s_2,t_2),\ldots,(p_m,s_m,t_m)\rangle = \{P_m, S_m, T_m\}$, where each $b_i = (p_i,s_i,t_i)$, $b_i \in B$, represents one user access request. First, we parameterize it and extract the spatiotemporal attributes to form the spatiotemporal attribute sequence. Then, according to the access request predictive functions $\mathrm{Pre}'(p,s,t)$ and $\mathrm{Pre}''(p,s,t)$, we take the spatiotemporal attribute sequence as the input, and the output $\widehat{b}_{m+1} = (\widehat{p}_{m+1},\widehat{s}_{m+1},\widehat{t}_{m+1})$ is predicted as the access request.

We define a sliding and adaptive observation window with initial size w, and we take the spatiotemporal attribute sequence $\{P_W, S_W, T_W\}$, which falls into the observation window, as the input of the prediction function. Then, we judge whether $\{P_W, S_W, T_W\}$ can satisfy formula (8). If it satisfies (8), we use the conjoint predictive function; otherwise, we use the independent attribute predictive function. As a result, $\widehat{p}_{m+1}$, $\widehat{s}_{m+1}$, and $\widehat{t}_{m+1}$ can be predicted. Here,

$$\begin{aligned} P_W &= (p_{m-w+1}, p_{m-w+2}, \ldots, p_m) \\ S_W &= (s_{m-w+1}, s_{m-w+2}, \ldots, s_m) \\ T_W &= (t_{m-w+1}, t_{m-w+2}, \ldots, t_m). \end{aligned} \quad (10)$$

3.2.1. Independent Attribute Predictive Function. For the access requests that dissatisfy formula (8), we predict the spatiotemporal attribute according to independent attribute prediction $\mathrm{Pre}'(p,s,t) = \{\mathrm{Pre}(p), \mathrm{Pre}(s), \mathrm{Pre}(t)\}$, and then we form the predictive access request $\widehat{b}_{m+1} = (\widehat{p}_{m+1},\widehat{s}_{m+1},\widehat{t}_{m+1})$.

(1) Location Attribute Prediction. Because regional meshing is used for the entire geographic area, before prediction, we need to judge whether the coordinate points belong to the same cell or neighbor cells according to formula (2). If the points belong to the same cell, we trigger the prediction; otherwise, we forgo prediction. The pseudocode for prediction of the location attribute is shown in Pseudocode 1.

Here, w' is the minimum observation window, $R'(s_{ij},\phi_{ij})$ is a temporary ruleset to store matched associate items and the *confidence*, and $P_{W-1} = \langle p_{m-w+2}, p_{m-w+3}, \ldots, p_m\rangle$ is a location attribute sequence with an observation window of one-bit duration. $\mathrm{Match}(P_W, R(g_{ij},\phi_{ij}))$ is a rule matching function, whose output is an associate rule item matching successfully with P_W, and the output is NULL when the match fails. For example, if the coordinate points of P_W belong to the same cell g_{ij} or neighbor cell, we use the rule matching function $\mathrm{Match}(P_W, R(g_{ij},\phi_{ij}))$ to scan associate rulesets $R(g_{ij},\phi_{ij})$ for three matched associate items:

$$\begin{aligned} (p_1,p_2,\ldots,p_m) &\longrightarrow (p'_{m+1},\phi_1) \\ (p_1,p_2,\ldots,p_m) &\longrightarrow (p''_{m+1},\phi_2) \\ (p_1,p_2,\ldots,p_m) &\longrightarrow (p'''_{m+1},\phi_3), \end{aligned} \quad (11)$$

where the *confidence* satisfies $\phi_1 + \phi_2 + \phi_3 = 1$. If $\phi_1 = \max(\phi_1,\phi_2,\phi_3)$, the location attribute of the predicted access request is $\widehat{p}_{m+1} = p'_{m+1}$.

(2) Type Attribute Prediction. The type attribute prediction is similar to the location attribute prediction. Scan associate rulesets $R(s_{ij},\phi_{ij})$ according to the predictive function $\mathrm{Pre}(s)$. Find the associate rules matched with the type attribute sequence in the current observation window. Then, select the confidence rules with the largest confidence as the output results. Suppose that $(s_1,s_2,\ldots,s_m) \rightarrow (s'_{m+1},\phi_1)$ is the associate rule matched successfully with S_W, and ϕ_1 is the largest; then, the type attribute of the predicted access request is $\widehat{s}_{m+1} = s'_{m+1}$.

(3) Time Attribute Prediction. The time attribute prediction is based on the predictive function $\mathrm{Pre}(t)$. First, we perform d-order difference processing for T_W to achieve stationarity. Then, we use ARIMA to calculate the time attribute of the predicted access request; the result is given by

$$\widehat{t}_{m+1} = t'_{m+1} = \begin{cases} t_m, & d=0 \\ 2t_m - t_{m-1}, & d=1 \\ 3t_m - 3t_{m-1} + t_{m-2}, & d=2. \end{cases} \quad (12)$$

3.2.2. Conjoint Attribute Predictive Function. For the access requests that satisfy formula (9), we predict the next access request $\widehat{b}_{m+1}$ according to the conjoint attribute prediction function $\mathrm{Pre}''(p,s,t)$ and then form the predictive access request:

$$\widehat{b}_{m+1} = (\widehat{p}_{m+1},\widehat{s}_{m+1},\widehat{t}_{m+1}) = \begin{cases} (p_m,s_m,t_m), & d=0 \\ (p_m,s_m,2t_m-t_{m-1}), & d=1 \\ (p_m,s_m,3t_m-3t_{m-1}+t_{m-2}), & d=2. \end{cases} \quad (13)$$

3.3. Data Prefetching. Data prefetching is performed to load data into the cache in accordance with the predicted request. To avoid unnecessary consumption of memory and computing resources, we built two data structure queues of length λ. One is used to store the actual access requests and the other to

Algorithm $\widehat{p}_{m+1} = \text{pre}(P_W)$
Input
P_W: location attribute sequence of the observation window
w': the minimum observation window
$R(g_{ij}, \phi_{ij})$: associate rulesets of a regional cell
$R'(g_{ij}, \phi_{ij})$: temporary associate rulesets
Output
$\widehat{p}_{m+1}$: predicted location attribute
while $(w \geq w')$
  if $(\text{Match}(P_W, R(g_{ij}, \phi_{ij})))$ // find a matching associate rules
    $R'(g_{ij}, \phi_{ij}) \Leftarrow \text{Match}(P_W, R(g_{ij}, \phi_{ij}))$ // put the matched rules into the temporary rulesets
    break
  else
    $w \leftarrow w - 1; P_W \leftarrow P_{W-1}$
end while
for $(\text{Scan}(R'(g_{ij}, \phi_{ij})))$ // scan the temporary rulesets
  if $\phi_1 = \max(\phi_{ij})$ // find the largest degree of confidence
    $\widehat{p}_{m+1} \leftarrow R'(g_{ij}, \phi_1)$
end for

Pseudocode 1: Pseudocode for predicting location attribute.

store the predicted requests. By calculating the consistent rate of the two queues, we can judge the degree of credibility of the current predicted requests. If the consistent rate achieves the predefined threshold, we assume that the predicted request is credible, and accordingly we carry out the data prefetching.

Suppose that the actual access request stored in a queue of length λ is $\{b_{m-\lambda+1}, b_{m-\lambda+2}, \ldots, b_m\}$ and the predicted request is $\{\widehat{b}_{m-\lambda+1}, \widehat{b}_{m-\lambda+2}, \ldots, \widehat{b}_m\}$. If they satisfy formula (14), we assume that the predicted request $\widehat{b}_{m+1} = (\widehat{p}_{m+1}, \widehat{s}_{m+1}, \widehat{t}_{m+1})$ is credible, and we load the corresponding data d_{n+1} into the cache:

$$\begin{aligned} &\widehat{p}_{m-\lambda+1} = p_{m-\lambda+1}, \widehat{p}_{m-\lambda+2} = p_{m-\lambda+2}, \ldots, \widehat{p}_m = p_m \\ &\widehat{s}_{m-\lambda+1} = s_{m-\lambda+1}, \widehat{s}_{m-\lambda+2} = s_{m-\lambda+2}, \ldots, \widehat{s}_m = s_m \\ &\widehat{t}_{m-\lambda+1} = t_{m-\lambda+1}, \widehat{t}_{m-\lambda+2} = t_{m-\lambda+2}, \ldots, \widehat{t}_m = t_m. \end{aligned} \tag{14}$$

4. Experiment

This section consists of three parts. The first part introduces the performance evaluation metrics for our prefetching scheme. The second part describes the experimental data and methods. The last part presents and discusses the results of the experiments.

4.1. Evaluation Metrics. We propose five criteria for performance evaluation of the proposed prefetching scheme in terms of accuracy, efficiency, and effectiveness.

Prediction Accuracy. It is the correct number of predicted requests as a percentage of the total number of requests.

Prediction Coverage. It is the number of predicted requests as a percentage of the total number of requests.

Pattern Mining Time. It is the time consumed for mining user access patterns from the history of user access requests.

Request Prediction Time. It is the average time for predicting an access request.

Average Response Time. It is the average response time to obtain a single data item.

4.2. Experimental Data and Methods. The experimental data was obtained from Wuhan smart city network application demonstration platform, which includes 14 types of sensors in different regions; it has been generating sensor data since January 1, 2010, and provides 20 types of predefined applications to the public. We obtained the historical user access information from the user access log in the server for the period from September 1, 2014, to February 16, 2015. After processing, we generated 1,819,008 data access requests. The initial 1,628,183 requests formed the training set for mining user access patterns and constructing the predictive function. The remaining 190,825 requests formed the test set for testing the performance of the prefetching scheme.

The performance of the prefetching scheme is determined by the initial size of the observation window, regional meshing level, support threshold, and confidence threshold. Considering that the pattern mining is performed offline, the objective of the prediction is to choose the association rules with the maximum confidence. In order to choose the maximum number of rules and improve the trigger probability of prediction, we set the *support threshold* at 0.05% of the total number of access requests; that is, $\delta_p = \delta_s = 0.05\%$, and the *confidence threshold* was 0.01; that is, $\phi_p = \phi_s = 0.01$. Experiments were conducted to determine the changes in the evaluation metrics with different initial sizes of the observation window (w = 2, 3, 4, 5, and 6) and different regional

Table 1: Location attribute prediction.

Regional meshing	Accuracy/coverage (%) Window size				
	2	3	4	5	6
1×1	86.95/93.92	89.11/93.92	89.19/93.92	89.23/93.92	89.27/93.92
50×50	81.04/91.11	85.93/91.11	85.96/91.11	86.01/91.11	86.08/91.11
80×80	80.26/90.32	85.11/90.32	85.17/90.32	85.21/90.32	85.26/90.32
100×100	79.70/89.79	84.58/89.79	84.63/89.79	84.72/89.79	84.75/89.79
120×120	79.22/89.27	84.07/89.27	84.13/89.27	84.18/89.27	84.24/89.27
150×150	78.43/88.49	83.27/88.49	83.33/88.49	83.39/88.49	83.43/88.49

meshing levels (row×col = 1×1, 50×50, 80×80, 100×100, 120×120, and 150×150), and the average response time for users to access a single data item was recorded. Finally, we compare our prefetching algorithm (STAP, spatial-temporal attributes prediction) with associate rules discovery prefetching algorithm (ARP) proposed by [5] and neighbor selection Markov Chain prefetching algorithm (MCP) proposed by [15].

4.3. Experimental Results

4.3.1. Prediction Accuracy and Coverage

(1) Spatiotemporal Attribute Prediction

(a) Location Attribute Prediction. The prediction accuracy and coverage of the location attribute with different regional meshing levels and observation window sizes are summarized in Table 1. As can be seen, for the same observation window size, when the meshing level increases (the area of the cell becomes smaller), the accuracy and coverage decrease. This is because the regional meshing results in the loss of the association rules between nonneighbor cells and leads to unsuccessful trigger prediction of some access requests. In contrast, for the same regional meshing level, as the observation window size increases, the accuracy gradually increases while the coverage remains unchanged. This can be explained as follows. On the one hand, the larger the window size, the greater the amount of available prediction information. On the other hand, the short rules form a subset of the long rules, and the current observation window cannot trigger prediction; the size of the window will adaptively decrease for further prediction until the minimum size is reached.

(b) Type Attribute Prediction. The prediction of the type attribute is related only to the observation window size. As can be seen from Table 2, as the observation window size increases, the prediction accuracy gradually increases from 94.38% to 96.05%, while the prediction coverage remains unchanged at 96.76%. This can be explained as follows. The larger the observation window size, the greater the amount of available prediction information, and the adaptive observation window size achieves exactly the same prediction coverage.

(c) Time Attribute Prediction. The time attributesare predicted by the ARIMA model, because the change trend of the time attribute sequences comprises only three situations (remaining unchanged, changing periodically, and changing in step length). Furthermore, the adaptive observation window size makes the ARIMA model available for all the time attribute sequences. Therefore, the prediction errors appear only in the case of inconsistent change trends of sequence, and all requests falling within the observation windows can be predicted. The experimental results show that the prediction accuracy of the time attribute with different observation windows sizes is 88.63%, while the prediction coverage is 100%.

Table 2: Type attribute prediction.

Window size	2	3	4	5	6
Accuracy (%)	94.38	95.28	95.52	95.80	96.05
Coverage (%)	96.76	96.76	96.76	96.76	96.76

(2) Access Request Prediction. The final prediction request must include three basic attributes: location, time, and type. Therefore, we should synthesize the spatiotemporal attributes predicted previously to form the access request.

The prediction accuracy and coverage of the user access requests with different regional meshing levels and observation window sizes are shown in Figures 2 and 3. We can see that, without meshing for the same observation window size, the prediction accuracy and coverage are the highest, and as the meshing level increases, the prediction accuracy gradually decreases. This is because the regional meshing can result in the loss of association rules between nonneighbor cells and reduce the probability of triggering prediction. In contrast, at the same regional meshing level, the prediction accuracy at different observation window sizes varies slightly, except for a window size of 2, where the prediction accuracy is significantly lower. And the curves for different window sizes overlap completely. These results can be explained as follows. First, the rules mined from user access are very similar when the rule length is greater than 2. Second, when the window size $w = 2$, it will fail to trigger joint property prediction. Finally, as stated before, the adaptive observation window size has no effect on the coverage.

Figures 4 and 5 show the prediction accuracy and coverage of the user access of the proposed prefetching scheme STAP and another two schemes, ARP [5] and MCP [15],

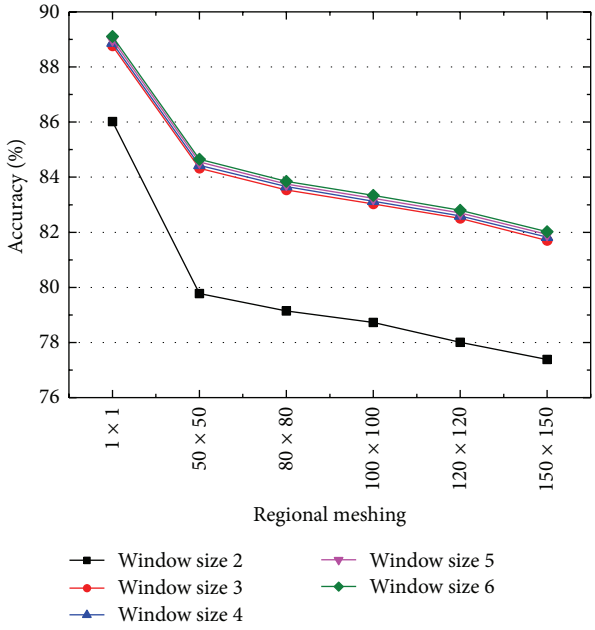


Figure 2: Request accuracy.

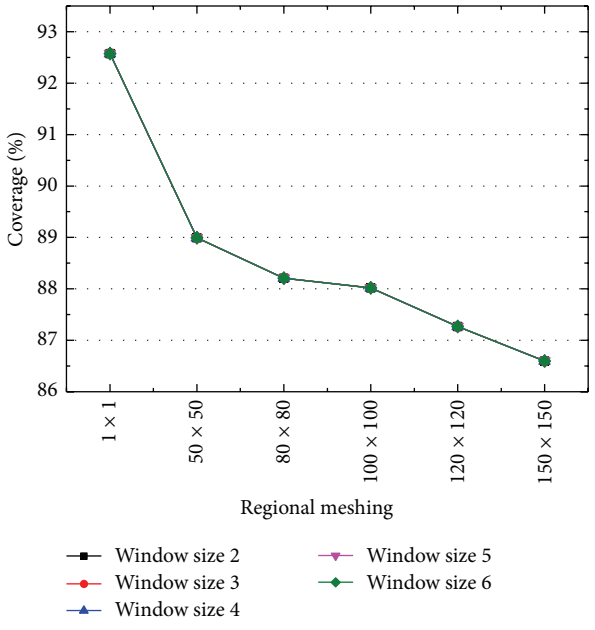


Figure 3: Request coverage.

with different observation window sizes. Regional meshing levels were set as 50 ∗ 50 in the experiment, the observation window size is the active session window size in ARP, and the number of previous movements was monitored in MCP. As shown in Figure 4, the STAP achieves the highest prediction accuracy compared with the other two prefetching schemes. This is because the ARP and MCP can only predict user access requests that have appeared in history. When the observation window size increases gradually, prediction accuracy of the three prefetching schemes are all improved. The prediction coverage of the user access requests with different observation

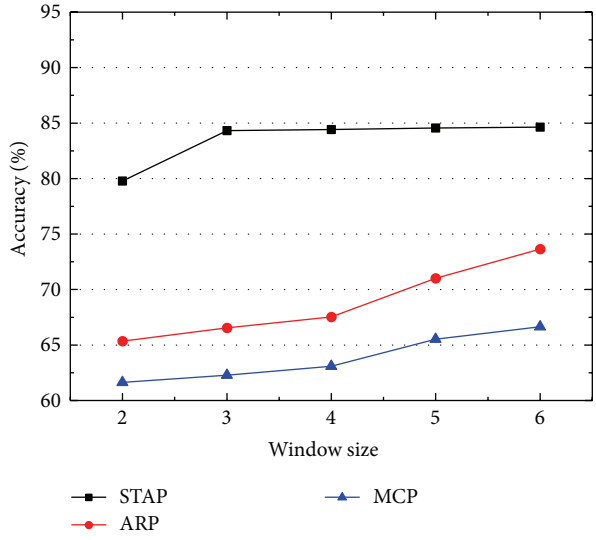


Figure 4: Request accuracy.

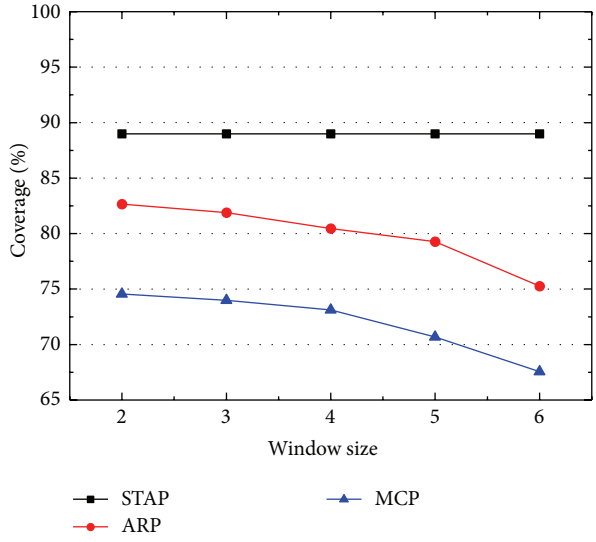


Figure 5: Request coverage.

window size is shown in Figure 5. The prediction coverage of ARP and MCP is lower than STAP, because it cannot trigger the prediction when the user's access request has not happened in history. At the same time, as the observation window size increases gradually, the prediction coverage of ARP and MCP decreases, while STAP remains unchanged because of its adaptability.

4.3.2. Pattern Mining and Prediction Times. Figure 6 shows the time consumed for mining user access patterns from the history of user access requests. As can be seen, the time of construction decreases from 430,401 s to 21,237 s; it falls drastically as the regional meshing level increases. This is because the calculation of associate rules for the coordinate points of the entire geographic area is disaggregated to the calculation of cells and neighbor cells based on regional meshing, and

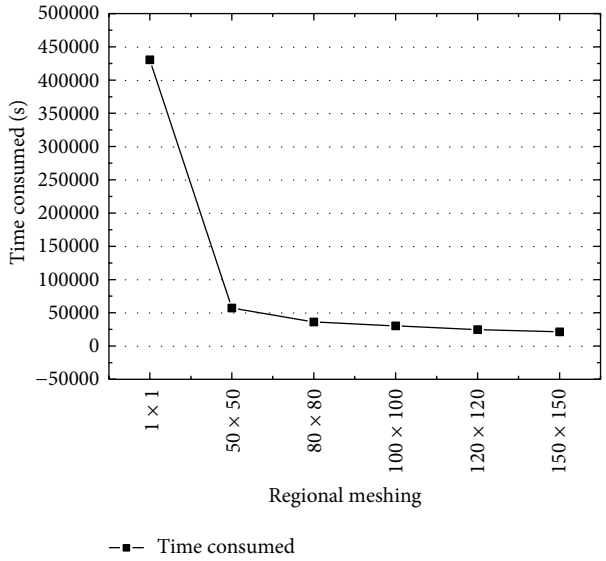


Figure 6: Patterns mining time.

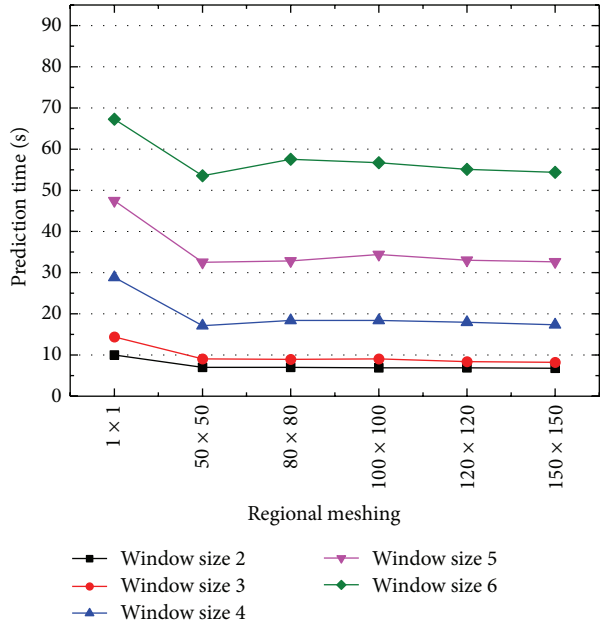


Figure 7: Request prediction time.

partial and incremental solutions of the association rules of the location attribute are achieved.

Figure 7 shows the time for predicting 190,825 access requests with different regional meshing levels and observation window sizes. As can be seen, without meshing for the same observation window, maximum time is consumed. Then, as the meshing level increases, the time consumed decreases gradually and reaches a stable level. In contrast, for the same regional meshing level, the prediction time clearly varies with the size of the observation window. The smaller the window size, the short the prediction time. This is because, without regional meshing, the entire regional rulesets need to be scanned for matching to predict requests.

Table 3: The average response time (ms).

Nonprefetching	MCP	ARP	STAP
0.378	0.258	0.245	0.209

However, with meshing, only the association rules belonging to the cell are to be scanned, and it is clear that the larger the window size, the longer the prediction time.

4.3.3. Average Response Time. From the abovementioned experimental results, we can see that, in the request prediction phase, although the prediction accuracy and coverage of requests decrease under the regional meshing, the time consumed for pattern mining is effectively reduced, and more importantly, regional meshing avoids the numerous calculations required for updating the location attribute. In the data prefetching phase, it is clear that the larger the length of the buffer queue, the more credible the request for the previous prediction, and the prefetching data is more accurate. However, it also means that some data of the predicted request fail to be prefetched.

Therefore, to compare the average response time, we set the regional meshing as row × col = 50 × 50, the initial size of the observation window as $w = 3$, and the length of the buffer queue as $\lambda = 3$ and then test the average response time for users to access a single data item of the four schemes: STAP, ARP, MCP, and nonprefetching. From Table 3, we can see that the average response time for users to access a single data item is 0.378 ms under the nonprefetching scheme. When the prefetching mechanism is employed, the average response time is reduced obviously; the proposed scheme STAP gets the minimum average response time, with a 44.71% reduction over nonprefetching mechanism.

5. Conclusion

In this study, we exploited the spatiotemporal features of user access for spatiotemporal data in a smart city. We mapped the history of user access requests to the spatiotemporal attribute domain to perform correlation analysis and identify variation rules, mined the user access patterns, and developed a simple and efficient prefetching scheme. Specifically, the regional meshing methods use the spatial locality of user access; thus, they not only achieve partial and incremental solutions of association rules but also reduce the computation considerably. Furthermore, the ARIMA model uses the time stationarity of user access and realizes accurate prediction of the time attribute. Experimental results showed that our prefetching scheme is simple yet effective, and it can reduce the user access latency significantly.

Finally, the proposed concept of access pattern mining in the spatiotemporal domain for spatiotemporal data not only has a significant effect on spatiotemporal data prefetching in a smart city but also can be widely used for user-personalized recommendation, active pushing of information, and other network applications based on location services.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key Basic Research and Development Program of China (no. 2011CB302306), the National Natural Science Foundation of China (no. 61471162), and the Open-End Foundation of Hubei Collaborative Innovation Center for High-Efficiency Utilization of Solar Energy (no. HBSKFMS2014032).

References

[1] S. P. Vanderwiel and D. J. Lilja, "Data prefetch mechanisms," *ACM Computing Surveys*, vol. 32, no. 2, pp. 174–199, 2000.

[2] G. Pallis, A. Vakali, and J. Pokorny, "A clustering-based prefetching scheme on a Web cache environment," *Computers and Electrical Engineering*, vol. 34, no. 4, pp. 309–323, 2008.

[3] M. Wan, A. Jönsson, C. Wang, L. Li, and Y. Yang, "Web user clustering and web prefetching using random indexing with weight functions," *Knowledge & Information Systems*, vol. 33, no. 1, pp. 89–115, 2012.

[4] M. Khosravi and M. J. Tarokh, "Dynamic mining of users interest navigation patterns using naive Bayesian method," in *Proceedings of the IEEE 6th International Conference on Intelligent Computer Communication and Processing (ICCP '10)*, pp. 119–122, Cluj-Napoca, Romania, August 2010.

[5] M. Bamshad, H. Dai, T. Luo, and N. Miki, "Effective personalization based on association rule discovery from web usage data," in *Proceedings of the 3rd International Workshop on Web Information and Data Management (WIDM '01)*, pp. 9–15, Atlanta, Ga, USA, November 2001.

[6] S. G. Matthews, M. A. Gongora, A. A. Hopgood, and S. Ahmadi, "Web usage mining with evolutionary extraction of temporal fuzzy association rules," *Knowledge-Based Systems*, vol. 54, pp. 66–72, 2013.

[7] C. Jianxi, W. Qingsong, C. Cheng, and F. Dan, "Adaptive prefetching scheme for storage system in multi-application environment," *IEEE Transactions on Magnetics*, vol. 49, no. 6, pp. 2762–2767, 2013.

[8] Y. Chen, S. Byna, and X. Sun, "Data access history cache and associated data prefetching mechanisms," in *Proceedings of the ACM/IEEE Conference on Supercomputing*, pp. 1–12, Reno, Nev, USA, 2007.

[9] S. Ahmad and S. Hsien-Hsin, "Data prefetching mechanism by exploiting global and local access patterns," in *Proceedings of the 1st International Journal of Instructional Level Parallelism Data Prefetching Championship (DPC-1 '09)*, Raleigh, NC, USA, February 2009.

[10] Y. Chen, H. Zhu, H. Jin, and X.-H. Sun, "Algorithm-level Feedback-controlled Adaptive data prefetcher: accelerating data access for high-performance processors," *Parallel Computing*, vol. 38, no. 10-11, pp. 533–551, 2012.

[11] S. Jiang, X. Ding, Y. Xu, and K. Davis, "A prefetching scheme exploiting both data layout and access history on disk," *ACM Transactions on Storage*, vol. 9, no. 3, pp. 317–318, 2013.

[12] Y. Chou, "Low-cost epoch-based correlation prefetching for commercial applications," in *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '07)*, pp. 301–313, IEEE, Chicago, Ill, USA, December 2007.

[13] H. Tang, X. Zou, J. Jenkins et al., "Improving read performance with online access pattern analysis and prefetching," in *Euro-Par 2014 Parallel Processing*, vol. 8632 of *Lecture Notes in Computer Science*, pp. 246–257, Springer, Basel, Switzerland, 2014.

[14] D.-J. Park and H.-J. Kim, "Prefetch policies for large objects in a web-enabled GIS application," *Data & Knowledge Engineering*, vol. 37, no. 1, pp. 65–84, 2001.

[15] H. L. Dong, J. S. Kim, S. D. Kim, K. C. Kim, Y. S. Kim, and J. Park, "Adaptation of a neighbor selection Markov chain for prefetching tiled web GIS data," in *Advances in Information Systems*, vol. 2457 of *Lecture Notes in Computer Science*, pp. 213–222, Springer, Berlin, Germany, 2002.

[16] S. Yeşilmurat and V. Işler, "Retrospective adaptive prefetching for interactive Web GIS applications," *GeoInformatica*, vol. 16, no. 3, pp. 435–466, 2012.

[17] R. Li, R. Guo, Z. Xu, and W. Feng, "A prefetching model based on access popularity for geospatial data in a cluster-based caching system," *International Journal of Geographical Information Science*, vol. 26, no. 10, pp. 1831–1844, 2012.

[18] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pp. 1–12, Dallas, Tex, USA, May 2000.

[19] P. G. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, no. 17, pp. 159–175, 2003.

[20] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, "ARIMA models to predict next-day electricity prices," *IEEE Transactions on Power Systems*, vol. 18, no. 3, pp. 1014–1020, 2003.

[21] P. Areekul, T. Senjyu, H. Toyama, and A. Yona, "Combination of artificial neural network and ARIMA time series models for short term price forecasting in deregulated market," in *Proceedings of the Transmission & Distribution Conference & Exposition: Asia and Pacific*, pp. 1–4, IEEE, Seoul, The Republic of Korea, October 2009.

[22] Y. L. Huai, S. X. Chang, and Y. Liu, "A new method of pefetching I/O requests," in *Proceedings of the 2nd International Conference on Networking, Architecture, and Storage (NAS '07)*, pp. 217–224, IEEE, Guilin, China, July 2007.

[23] N. Tran and D. A. Reed, "Automatic ARIMA time series modeling for adaptive I/O prefetching," *IEEE Transactions on Parallel & Distributed Systems*, vol. 15, no. 4, pp. 362–377, 2004.

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 412418, 9 pages
http://dx.doi.org/10.1155/2015/412418

Research Article

Factor Knowledge Mining Using the Techniques of AI Neural Networks and Self-Organizing Map

Pao-Kuan Wu[1] and Tsung-Chih Hsiao[2]

[1]*Department of Urban Planning, College of Architecture, Huaqiao University, No. 668 Jimei Avenue, Jimei District, Xiamen, Fujian 361021, China*
[2]*College of Computer Science and Technology, Huaqiao University, No. 668 Jimei Avenue, Jimei District, Xiamen, Fujian 361021, China*

Correspondence should be addressed to Tsung-Chih Hsiao; hsiaotc@gmail.com

Received 5 June 2015; Revised 14 September 2015; Accepted 15 September 2015

Academic Editor: Yingshu Li

This paper offers a hybrid technique combined by artificial neural networks (ANN) and self-organizing map (SOM) as a way to explore factor knowledge. ANN and SOM are two kinds of pattern classification techniques based on supervised and unsupervised mechanisms, respectively. This paper proposes a new aspect to combine ANN and SOM as NNSOM process in order to delve into factor knowledge other than pattern classification. The experimental material is conducted by the investigation of street night market in Taiwan. NNSOM process can yield two results about factor knowledge: first, which factor is the most important factor for the development of street night market; second, what value of this factor is most positive to the development of street night market. NNSOM process can combine the advantages of supervised and unsupervised mechanisms and be applied to different disciplines.

1. Introduction

Since the technology of geographic information system (GIS) has been more popular and improved recently, how to apply suitable artificial intelligence (AI) algorithms with GIS that can assist researchers in solving their environmental problems is an important trend in urban-related domains nowadays [1–5]. There are two kinds of AI algorithm mainly discussed and applied in this paper: artificial neural networks (ANN) and self-organizing map (SOM).

ANN is a kind of AI technique based on the theory of neural science with useful advantages, such as the endurance of data noise, optimization and approximation, and great prediction ability [6, 7]. Because of these advantages, ANN has been applied to lots of comprehensive applications like land use changes [1, 2], regional labor market [8], traffic management [9], and regional economic activities [10], even though ANN has already been developed for quite a long time. Basically, ANN is a pattern classification technique based on supervised learning mechanism operated by statistic mathematics [7, 11]. Therefore, most of the ANN applications emphasized pattern classification performance. The discussion about the input factor analysis is still lacking because of the inherent limitation of ANN. The combination of SOM can bring out a new way for the issue of factor analyzing.

SOM also known as Kohonen map is another neural network technique based on unsupervised learning mechanism [12]. SOM is a kind of clustering technique which operated in dimensional space. The development of SOM is later than ANN but still has been widely applied in many research tasks like clustering data, monitoring process, and identifying particular pattern [13–16]. There are two important features of SOM: abstraction and visualization, which can make SOM unique clustering technique out of others [12, 17]. Depending on these two advantages, SOM not only can do clustering task but also can display the complicated distribution of data on limited dimensional space established by the assigned input factors. From this point of view, SOM can be seen as a kind of analyzing technique operated by specific input factors.

Overall, SOM can demonstrate data distribution through the assigned input factors with limited dimensional space, and then we can observe the density of data in order to learn

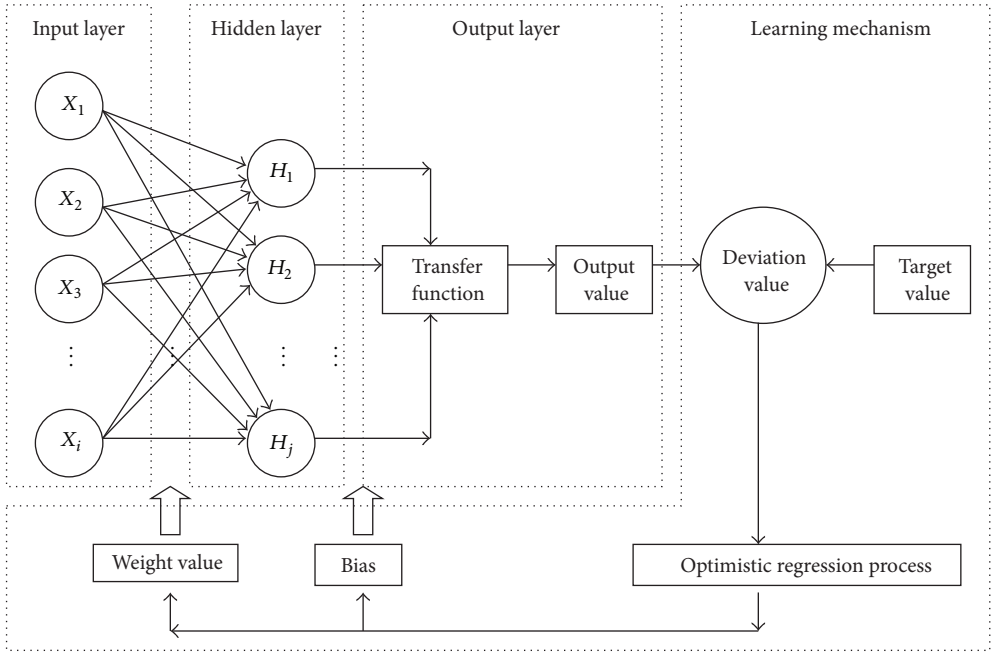


Figure 1: The structure of ANN consists of input layer, hidden layer, and output layer.

where sectors of the limited dimensional space could be more sensitive to the assigned input factors after clustering process. However, because of the essence of unsupervised mechanism, there is no information about target pattern along the way of SOM procedure. Therefore, we do not have enough information to clarify these sensitive sectors which related to the assigned input factors that are positive or negative for particular pattern. On the other hand, the outcome value of ANN calculated by the process of supervised mechanism can indicate the tendency of target pattern in order to do pattern classification performance. But there is no such operation in ANN procedure that can assist us in displaying the detailed distribution of data by the assigned input factors and help us answer question like the following: what values of the assigned input factors could be more sensitive with target pattern? Hence, ANN and SOM are both lacking the ability to delve into detailed factor knowledge between the assigned input factors and target pattern traditionally. But the interesting thing is that the lack of ability of each one can be found out from the other one. That is the main motivation of this paper for combining ANN and SOM.

The main purpose of this paper is to combine the advantages of ANN and SOM in order to conquer the problems mentioned above. This paper offers an idea: look at ANN outcome as one particular input factor which can indicate the impact of target pattern and then use this ANN factor with other input factors to conduct SOM procedure. By doing so, we can perform the clustering process with the information of target pattern, and then deeply analyzing the correlations between the assigned input factors and target pattern can be possible.

As the notion mentioned above, this paper presents a combination technique of ANN and SOM named NNSOM as a specific factor analyzing process. In order to demonstrate the abilities of NNSOM, we practice an experiment related to urban space study. The experimental material is a kind of commercial development on the streets, Taiwanese street night market. The application of NNSOM can solve two questions after the experiment: first, which factor is the most important factor for target pattern; second, what value of this factor is most positive to target pattern. NNSOM process can combine the advantages of supervised and unsupervised mechanisms and be applied to different disciplines for exploring factor knowledge.

2. Methodologies and Techniques

2.1. Artificial Neural Networks (ANN). ANN are well known as perceptron logic which derived from the imitation of human perception. The mechanism of ANN is like the function of our brain which can receive outside stimulation by neurons and then produce the best reaction by comprehensive consideration. For research application, the set of input factors is like the recipient part which can define and receive lots of information, then through a series of neural process to produce ANN outcome or pattern classification result [7, 11].

Normal ANN consists of three components. As Figure 1 shows, those are "input layer," "hidden layer," and "output layer." Hidden layer is particularly a sensitive part among those components. Linear ANN refers to no hidden layer that has clearest correlation between target pattern and input factors, even though this model might not have well ANN performance. In contrast, nonlinear ANN refers to at least one hidden layer existing that can easily get higher ANN performance but lose crucial information between target pattern and input factors and that is the reason for why multilayer

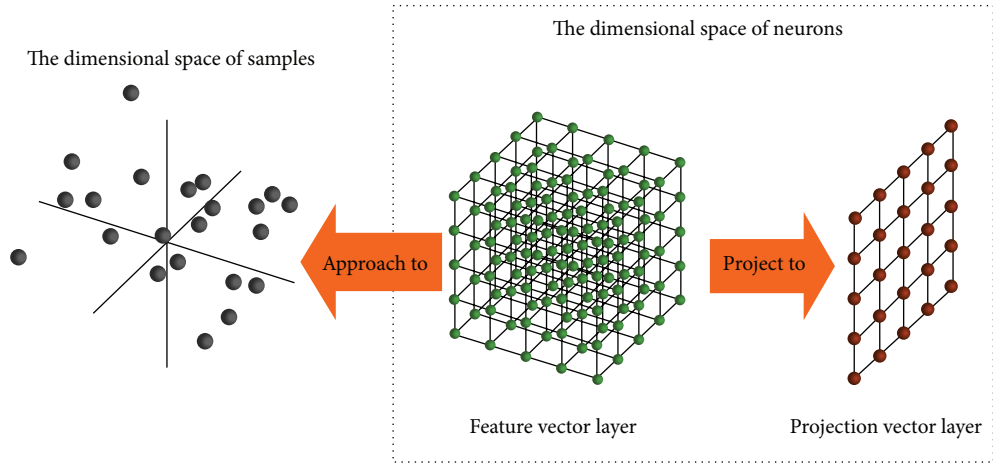


Figure 2: The samples recorded in the dimensional feature space of three dimensions for depicting the concept and structure of SOM.

ANN is criticized as a kind of "black box" technique [18–20]. This paper emphasized delving into the detailed information between target pattern and input factors. Therefore we implement linear ANN as the reason for simplification in order to emphasize the abilities of NNSOM process.

Through supervised learning process, ANN can do pattern classification according to reality record. The reality record is so-called target pattern which is the standard for ANN output. Minimizing the deviation between ANN output and target pattern is the goal of ANN training iterations. The ANN outcome can be calculated by (1) after ANN training iteration:

$$Y_j = \sigma\left(\sum_{i=1}^{n} W_{ij} X_i + \beta\right), \quad (1)$$

where Y_j is the ANN outcome value of jth sample; X_i is ith factor value; W_{ij} is the ANN weight value; σ is the transfer function; and β is a bias value.

Normal ANN result is the performance of pattern classification transferred by the ANN outcome. Therefore ANN outcome can be seen as an indicator which can reveal the tendency of target pattern. By this aspect, we can do special factor analysis as we combine the ANN outcome into SOM.

2.2. Self-Organizing Map (SOM). SOM or Kohonen map is based on unsupervised mechanism. The operation of SOM is conducted in dimensional feature space to process the distribution of data [12]. Normal SOM consists of two kinds of dimensional space; those are "the dimensional space of samples" which contains all samples recorded by their feature vectors and "the dimensional space of neurons" which contains all neurons with regular distribution. In addition, "the dimensional space of neurons" consists of two kinds of SOM vector layer; those are "SOM feature vector layer" and "SOM projection vector layer." The number of dimensions in "SOM feature vector layer" is the same as "the dimensional space of samples." But the number of dimensions in "SOM projection vector layer" can be any (normally using two dimensions for planar display). Figure 2 is the illustration for the concept and structure of SOM.

SOM can do clustering task by competitive learning process. There are two kinds of neurons that can gain momentum to get closer to the objective sample in order to perform clustering result: BMU neuron and the neurons affected by "neighborhood effect" of BMU neuron [14, 16, 21]. For one neuron where the distance between itself and the objective sample is shorter than the other neurons, this neuron is the best matching unit (BMU). Besides BMU neuron, some other neurons whose positions are near to BMU neuron can be affected by "neighborhood effect." We set a fixed radius centered by BMU neuron for the reason of simplification, if some other neurons within this radius can be affected by fixed "neighborhood effect." BMU neuron and the neurons affected by "neighborhood effect" can change their positions for approaching the objective sample by the calculation of

$$\underline{\omega}_j(t+1) = \underline{\omega}_j(t) + \delta\left(\underline{x}_i - \underline{\omega}_j(t)\right), \quad (2)$$

where $\underline{\omega}_j$ is jth neuron's feature vector; $\underline{x}_i$ is ith objective sample's feature vector; t refers to tth iteration; δ is the parameter which refers to momentum; δ is calculated by

$$\delta = \mu * s * d(t), \quad (3)$$

where μ is the learning rate set in the range 0~1; s is the parameter of "neighborhood effect"; if one calculated neuron is BMU neuron, s should be set as 1; if one calculated neuron is affected by "neighborhood effect," s should be set less than 1, where $d(t)$ refers to a continuously decreased parameter; while the SOM iteration is running, $d(t)$ will be continuously decreased. $d(t)$ is calculated by

$$d(t) = 1 - \left(\frac{t}{N}\right), \quad (4)$$

where N is the number of iterations.

Once finishing SOM procedure, the latest positions of neurons in "SOM feature vector layer" can be projected onto

"SOM projection vector layer" for abstracting and visualizing the complicated distribution of samples. The demonstration of "SOM projection vector layer" can reveal which sectors have high density of samples by the assigned features. If we combine the ANN outcome into SOM as one kind of feature vector, then extracting the information from "SOM projection vector layer" with the information of target pattern can be possible. That is a unique concept of this paper compared to other applications of the combination of ANN and SOM.

2.3. The Unique Concept for Combining ANN and SOM as NNSOM. Some researchers have tried to combine the techniques of SOM and ANN for application research [22, 23], but this paper has different concept to combine both techniques. Most applications are applying SOM as a preliminary step of pattern classification. They often implement the clustering process of SOM to find out potential patterns first and then use these potential patterns as target patterns to conduct ANN for improving pattern classification performance. Thus, pattern classification is the main purpose of this sort of applications. In contrast, the main purpose of this paper is the application of factor analyzing by SOM mechanism; ANN is applied as a referential factor to indicate the impact of target pattern. Therefore, the operational steps in this paper are ANN procedure first and then SOM procedure in sequence. Because of the different motivation, this paper offers a unique concept to combine ANN and SOM as NNSOM process.

The features of SOM are clustering and abstracting the complicated distribution of data [12, 17]. The demonstration of "SOM projection vector layer" can extract detailed information with the assigned input factors. On the other hand, the ANN outcome can be seen as an indicator which can reveal each sample's tendency of target pattern. The concept of NNSOM is that if we use the ANN outcome as one kind of dimensional feature and operate with SOM in competitive learning process, then hire the ANN outcome and one objective factor as two feature dimensions to establish and demonstrate "SOM projection vector layer." By doing so, we can deeply reveal detailed correlation between the ANN outcome and the objective factor. That means NNSOM process can explore detailed information between target pattern and the objective factor. Therefore, NNSOM can have the efficiency to delve into the detailed factor knowledge. For demonstrating the abilities of NNSOM, we practice an experiment in the following content.

3. Experimental Material and Operation Settings

The experimental material is Feng Chia night market, Taichung, Taiwan, which is a kind of Taiwanese night market developed along the streets around Feng Chia University. This kind of urban commercial phenomenon has some features like complexity, uncertainty, and being flexible and dynamic [24, 25]. The experiment is to investigate what environmental factors can affect a number of locations on the streets being occupied by informally commercial activities. The definition of informally commercial activities is various commercial activities or vendors occupying the locations of arcade, sidewalk, or road. These locations cannot be allowed to do any kinds of businesses by Taiwanese regulations, and this phenomenon is criticized as "out of place" generally [26].

The experimental area is set on the streets of Feng Chia night market, and the target pattern is the locations on the streets occupied by informally commercial activities. In order to define and record each location exactly, all the street areas have been transformed into lattice formation with GIS platform, as Figure 3 shows. Each lattice represents one grid-sample.

The number of grid-samples is 10940, and that contains 1717 occupied grid-samples (the target pattern) and 9223 unoccupied grid-samples. Each grid-sample is described by several environmental factors which derived from related literatures [26–29]. The descriptions of each factor are listed in Table 1, and all the factors have been dealt with through normalization by scaling of the range 0~1.

The experiment consists of two experimental phases. The first experimental phase is ANN operation. In order to simplify the operation and focus on the purpose of factor analyzing, single-layer ANN is hired in this experimental phase. There are two results needed to be done in this experimental phase. The first result is to acquire the ANN outcome of each grid-sample, and this result can be calculated by (1). The second result is to find out which factor mentioned in Table 1 is the most important factor to affect the target pattern, and the second result can be done by analyzing the ANN weight values. All ANN experimental results in this experimental phase are then handing over to the second experimental phase.

The second experimental phase is NNSOM operation, which is mainly operated by SOM procedure and is including previous experimental results of ANN. There are eight features defined by the environmental factors of Table 1 and the ANN outcome for establishing the dimensional space of SOM. "SOM projection vector layer" is constructed by two-dimensional features which are the ANN outcome and the most important factor both derived from previous ANN operation. The formation of "SOM projection vector layer" is set by 35×35 grid formation; the vertical axis refers to the value of ANN outcome, and the horizontal axis refers to the value of the most important factor. Each SOM iteration is calculated by (2)~(4), and several SOM parameter settings are listed in Table 2. The next chapter is the results and discussions of NNSOM experiment.

4. Experimental Results and Discussions

4.1. The Results of ANN. The pattern classification performance of ANN is demonstrated in Table 3. Total accuracy is 84.23%. ANN procedure can derive the ANN outcome of each grid-sample as Figure 4 shows; dotted line is the classifying boundary; this boundary line is acquired by the average of 300 times of ANN performances. Any grid-samples where the ANN outcomes are above this boundary can be assigned as the target pattern.

We can observe the result of ANN weight values to evaluate which factor is the most important one for ANN conducting pattern classification. The ANN weight values of all factors are demonstrated in Table 4; checking this result

Table 1: The descriptions of environmental factors.

Factor denotation	Description
Grid location (GL)	For denoting that one grid-sample with its street location has certain impact of informally commercial attraction
Intersection factor (IntF)	The distance measured from one grid-sample to the nearest street intersection
Street width (SW)	The width of one street on which one grid-sample is located
Background building width (BBW)	The width of one building which is the background for one grid-sample
Core zone-1 (C1)	The distance measured from one grid-sample to core zone-1; core zone-1 is the main gate of Feng Chia University
Core zone-2 (C2)	The distance measured from one grid-sample to core zone-2; core zone-2 is the location of first bustling street intersection
Core zone-3 (C3)	The distance measured from one grid-sample to core zone-3; core zone-3 is the location of second bustling street intersection

Table 2: Parameter settings of SOM.

Parameter	Setting
The formation of "SOM projection vector layer"	35×35 grid formation
Learning rate (μ)	0.009
Neighborhood effect radius (centered by BMU neuron)	$1.45D$ (D refers to the initial distance of two adjacent neurons)
Neighborhood effect for BMU neuron (s)	1
Neighborhood effect for other neurons (s)	0.3
The number of iterations (N)	600

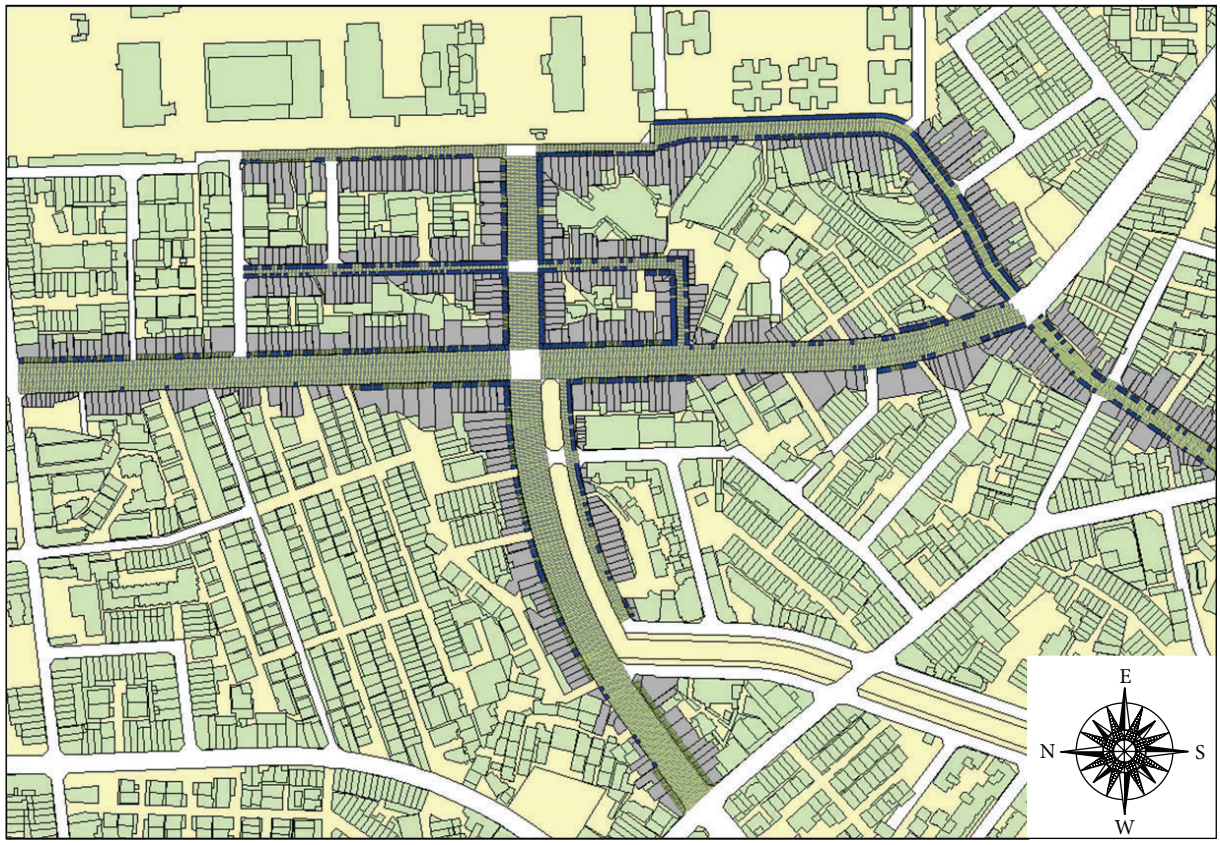


Figure 3: The distribution of target pattern recorded from reality; blue lattices are the grid-samples occupied by informally commercial activities, and they are the target pattern.

Table 3: The classification performance of ANN.

	Total grid-samples	Occupied grid-samples (target pattern)	Unoccupied grid-samples
The number of grid-samples	10940	1717	9223
The number of missed classifications	1725	102	1623
Accuracy rate	84.23%	94.06%	82.4%

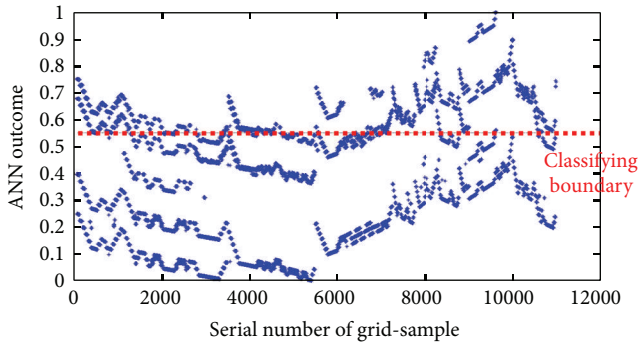


Figure 4: The result of ANN outcome.

in absolute value can figure out which factor is the most important one. If one factor has higher ANN weight value, this factor is more important for pattern classification [6, 7]. As Table 4 shows, the highest ANN weight value is "GL"; that means "GL" is the most important factor to assist ANN in judging some grid-samples as the target pattern. The next experimental phase of NNSOM can derive more detailed information between "GL" and the target pattern.

4.2. The Result of NNSOM. According to the previous result of ANN, we can then do advanced factor analyzing by NNSOM process. Because the most important factor is "GL," "SOM projection vector layer" has to be established by the ANN outcome and "GL" in order to dig out detailed factor information. The final result of "SOM projection vector layer" is demonstrated in Figure 5(a); the crosses are the latest positions of neurons, and the additive circles are the marks for denoting specific neurons that are surrounded by at least 1000 grid-samples within the radius of $0.4D$ (D refers to the initial distance of two adjacent neurons) in dimensional space. These additive circles are the cluster centers which can reveal the high density areas of grid-samples.

There is a graphically statistic figure bar chart hired in NNOSM phase for analyzing detailed information between "GL" and the target pattern, as Figure 5 shows. The dotted line in Figure 5(a) is the ANN classifying boundary which is the same as the previous description of Figure 4. There are two bar charts: "bar chart 1" demonstrates the correlation between the "GL" value and the number of cluster centers below the ANN boundary line and "bar chart 2" demonstrates the correlation between the "GL" value and the number of cluster centers above the ANN boundary line. Particularly for "bar chart 2," this figure can determine what value of "GL" can be most positive to the target pattern with the distribution of cluster centers upon the ANN classifying boundary. As Figure 5 shows, the result of bar chart 2 can reveal that the most positive value of "GL" for the target pattern is 0.58. That means if one grid-sample's "GL" value is closer to 0.58, this grid-sample should have higher tendency to become the target pattern.

"GL" factor refers to different grid-sample locations on the streets having different levels of informally commercial attraction. The setting of "GL" values and descriptions is listed in Table 5, and 0.58 is close to the sixth "GL" level. This analyzing result can reveal that this kind of street location should have higher attraction for informally commercial activities than other "GL" conditions. This result can give researcher detailed implication for better understanding of this specific phenomenon. The analyzing process of NNSOM can apply to analyze other environmental factors as well for deeply exploring the detailed factor knowledge.

5. Conclusion

Computational modeling can assist researchers in simulating the research objective in order to learn the possible trend with deductive logic. ANN and SOM are two kinds of computational modeling techniques that have different concept to do pattern simulation. How to get higher simulating accuracy when applying these two computational techniques is always the important issue; but factor analysis is still lacking to discuss even though there is a lot of potential information hidden in factor assumption that can be revealed by efficient way of factor analyzing. Offering a combination of ANN and SOM for the application of factor analysis is the main purpose of this paper.

ANN belongs to supervised mechanism, and the main feature of ANN is the orientation of target pattern. SOM belongs to unsupervised mechanism, and the main feature of SOM is abstraction and visualization of data distribution by the assigned input factors. This paper offers a unique concept to combine ANN into SOM as NNSOM process; that is using the ANN outcome as one of the SOM input factors which can indicate the impact of target pattern and then doing SOM competitive learning process. The result of NNSOM can display clustering result with one objective factor and the ANN outcome, and then analyzing sectors with high density of data distribution can deeply reveal detailed correlation between the objective factor and the ANN outcome. Therefore, the application of NNSOM can solve two questions related to factor knowledge: first, which factor is the most important factor for target pattern; second, what value of this factor is most positive to target pattern. Because of the unique concept to combine ANN and SOM, NNSOM can implement factor analysis for extracting potential information other than pattern classification.

This paper practices an experiment for demonstrating the abilities of NNSOM. The experimental material is the phenomenon of street night market, and the target is to analyze a number of locations on the streets occupied by informally

Table 4: The ANN weight values of each factor.

Factor	GL	IntF	SW	BBW	C1	C2	C3
Original ANN weight value	0.503729	0.112207	−0.12697	−0.02611	−0.17324	0.197353	−0.18364
Absolute value	0.503729	0.112207	0.12697	0.02611	0.17324	0.197353	0.18364

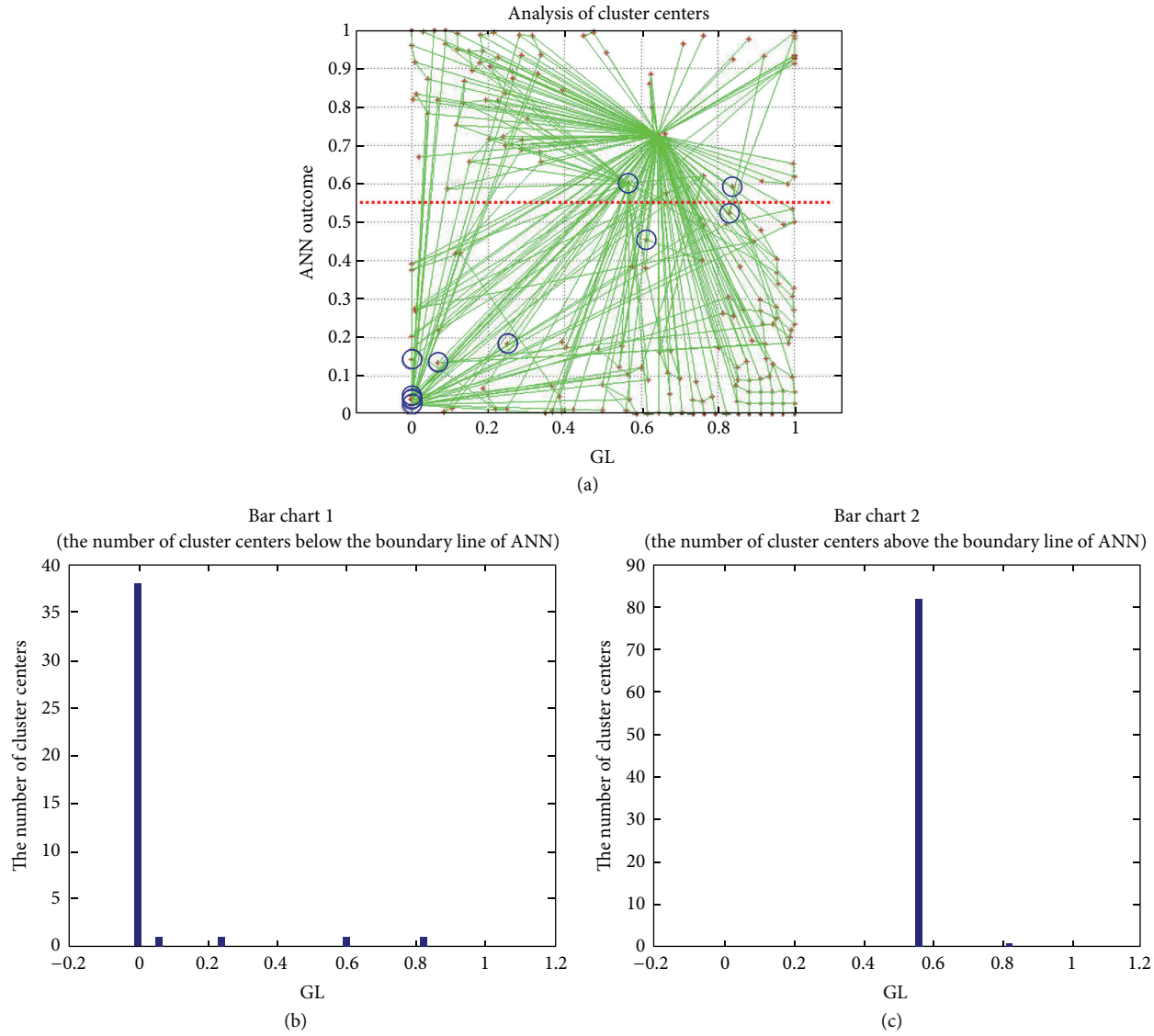


Figure 5: The demonstration of detailed factor analysis by NNSOM.

commercial activities. The experiment contains two experimental phases: the first one is ANN operation, and the second one is NNSOM operation. The first experimental phase can derive two experimental results: the ANN outcome of each sample is the first result; and the second result is that "GL" is the most important factor for the development of informally commercial activities along the streets. Through the analysis of "SOM projection vector layer" established by the ANN outcome and "GL" with bar chart in the second experimental phase, the most positive value of "GL" for the development of informally commercial activities along the streets is 0.58. This result can reveal detailed information about which kinds of locations on the streets can have highest attraction for informally commercial activities gathering. As the experimental demonstration, NNSOM can exactly extract detailed factor knowledge for better understanding of research objective.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Table 5: The description of GL factor setting.

GL level	Definition	Variable value
GL 1	Grid-sample located on the fast car lane	0
GL 2	Grid-sample located on the slow car lane	0.125
GL 3	Grid-sample located on the street intersection	0.25
GL 4	Grid-sample located on the scooter lane	0.375
GL 5	Grid-sample located on the bike lane	0.5
GL 6	Grid-sample located on the bike lane next to the building	0.625
GL 7	Grid-sample located on the pedestrian sidewalk	0.75
GL 8	Grid-sample located on the arcade	0.875
GL 9	Grid-sample located on the special zone of night market	1

Acknowledgments

This research is funded by "The Fund of Scientific Research Projects for Introduction Talents, Huaqiao University" (Project no. Z14Y0009), and The Education Department of Fujian Province (Project no. JA15031).

References

[1] B. C. Pijanowski, A. Tayyebi, J. Doucette, B. K. Pekin, D. Braun, and J. Plourde, "A big data urban growth simulation at a national scale: configuring the GIS and neural network based Land Transformation Model to run in a High Performance Computing (HPC) environment," *Environmental Modelling & Software*, vol. 51, pp. 250–268, 2014.

[2] R. M. Basse, H. Omrani, O. Charif, P. Gerber, and K. Bódis, "Land use changes modelling using advanced methods: cellular automata and artificial neural networks. The spatial and explicit representation of land cover dynamics at the cross-border region scale," *Applied Geography*, vol. 53, pp. 160–171, 2014.

[3] G. Grekousis, P. Manetos, and Y. N. Photis, "Modeling urban evolution using neural networks, fuzzy logic and GIS: the case of the Athens metropolitan area," *Cities*, vol. 30, no. 1, pp. 193–203, 2013.

[4] R. B. Thapa and Y. Murayama, "Urban growth modeling of Kathmandu metropolitan region, Nepal," *Computers, Environment and Urban Systems*, vol. 35, no. 1, pp. 25–34, 2011.

[5] I. Santé, A. M. García, D. Miranda, and R. Crecente, "Cellular automata models for the simulation of real-world urban processes: a review and analysis," *Landscape and Urban Planning*, vol. 96, no. 2, pp. 108–122, 2010.

[6] C. Marzban, "Basic statistics and basic AI: neural networks," in *Artificial Intelligence Methods in the Environmental Sciences*, S. E. Haupt, A. Pasini, and C. Marzban, Eds., pp. 15–47, Springer, Berlin, Germany, 1st edition, 2009.

[7] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University, New York, NY, USA, 1st edition, 1995.

[8] R. Patuelli, P. Nijkamp, S. Longhi, and A. Reggiani, "Neural networks and genetic algorithms as forecasting tools: a case study on German regions," *Environment and Planning B: Planning and Design*, vol. 35, no. 4, pp. 701–722, 2008.

[9] S. F. Kalyoncuoglu and M. Tigdemir, "An alternative approach for modelling and simulation of traffic data: artificial neural networks," *Simulation Modelling Practice and Theory*, vol. 12, no. 5, pp. 351–362, 2004.

[10] A. K. Nag and A. Mitra, "Forecasting daily foreign exchange rates using genetically optimized neural networks," *Journal of Forecasting*, vol. 21, no. 7, pp. 501–511, 2002.

[11] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, John Wiley & Sons, New York, NY, USA, 2nd edition, 2001.

[12] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, pp. 1–6, 1998.

[13] M. Oral, E. L. Oral, and A. Aydin, "Supervised vs. unsupervised learning for construction crew productivity prediction," *Automation in Construction*, vol. 22, pp. 271–276, 2012.

[14] D. Arribas-Bel, P. Nijkamp, and H. Scholten, "Multidimensional urban sprawl in Europe: a self-organizing map approach," *Computers, Environment and Urban Systems*, vol. 35, no. 4, pp. 263–275, 2011.

[15] P. Klement and V. Snášel, "Using SOM in the performance monitoring of the emergency call-taking system," *Simulation Modelling Practice and Theory*, vol. 19, no. 1, pp. 98–109, 2011.

[16] A. M. Kalteh, P. Hjorth, and R. Berndtsson, "Review of the self-organizing map (SOM) approach in water resources: analysis, modelling and application," *Environmental Modelling & Software*, vol. 23, no. 7, pp. 835–845, 2008.

[17] J. Kangas and T. Kohonen, "Developments and applications of the self-organizing map and related algorithms," *Mathematics and Computers in Simulation*, vol. 41, no. 1-2, pp. 3–12, 1996.

[18] M. M. Fischer, "Computational neural networks: a new paradigm for spatial analysis," *Environment and Planning A*, vol. 30, no. 10, pp. 1873–1891, 1998.

[19] S. Openshaw, "Neural network, genetic, and fuzzy logic models of spatial interaction," *Environment and Planning A*, vol. 30, no. 10, pp. 1857–1872, 1998.

[20] G. P. Zhang, "An investigation of neural networks for linear time-series forecasting," *Computers & Operations Research*, vol. 28, no. 12, pp. 1183–1202, 2001.

[21] F. Bação, V. Lobo, and M. Painho, "The self-organizing map, the Geo-SOM, and relevant variants for geosciences," *Computers & Geosciences*, vol. 31, no. 2, pp. 155–163, 2005.

[22] A. Jain and S. Srinivasulu, "Integrated approach to model decomposed flow hydrograph using artificial neural network and conceptual techniques," *Journal of Hydrology*, vol. 317, no. 3-4, pp. 291–306, 2006.

[23] H. Moradkhani, K.-L. Hsu, H. V. Gupta, and S. Sorooshian, "Improved streamflow forecasting using self-organizing radial basis function artificial neural networks," *Journal of Hydrology*, vol. 295, no. 1–4, pp. 246–262, 2004.

[24] A.-T. Hsieh and J. Chang, "Shopping and tourist night markets in Taiwan," *Tourism Management*, vol. 27, no. 1, pp. 138–145, 2006.

[25] J. Chang and A.-T. Hsieh, "Leisure motives of eating out in night markets," *Journal of Business Research*, vol. 59, no. 12, pp. 1276–1278, 2006.

[26] Y. A. Yatmo, "Perception of street vendors as 'out of place' urban elements at day time and night time," *Journal of Environmental Psychology*, vol. 29, no. 4, pp. 467–476, 2009.

[27] D. J. Timothy and G. Wall, "Selling to tourists: Indonesian street vendors," *Annals of Tourism Research*, vol. 24, no. 2, pp. 322–340, 1997.

[28] R. Bromley, "Street vending and public policy: a global review," *International Journal of Sociology and Social Policy*, vol. 20, no. 1-2, pp. 1–28, 2000.

[29] J. Cross, "Street vendors, and postmodernity: conflict and compromise in the global economy," *International Journal of Sociology and Social Policy*, vol. 20, no. 1-2, pp. 29–51, 2000.