Hindawi
Scientific Programming
Volume 2021, Article ID 9921551, 7 pages
https://doi.org/10.1155/2021/9921551

Research Article

Design and Implementation of Human-Computer Interaction System in Parallel Digital Library System Based on Neural Network

Jun Cao

Zaozhuang University Library, Zaozhuang 277100, China

Correspondence should be addressed to Jun Cao; 101024@uzz.edu.cn

Received 9 March 2021; Revised 26 March 2021; Accepted 18 April 2021; Published 17 May 2021

Academic Editor: Shah Nazir

Information and communication technologies are well thought-out as probable assets for the development of socioeconomics in developing countries. Studies have shown that enhanced infrastructure of telecommunication has facilitated means for underserved population development by various ways. Among the existing applications of ICT, the digital library systems provide with better solutions and respond to a variety of unmet needs of research institutions, scientific communities, and developments. With the development of digital library technology, the parallel database system has become the main tool for efficient information processing in the digital library system. On this basis, based on the parallel environment of the computer cluster, by coordinating the communication in the parallel environment, the coordinator, the collection machine, and the query processor can complete the operation of distribution, load, and maintenance, which has high efficiency and saves much precious time, supports the digital library to meet user requirements effectively, and meets the digital library's performance requirements for data, and also, the key problem in the parallel algorithm has been solved. The experimental results show that this parallel technique has very good performance and efficiency.

1. Introduction

Information and communication technologies (ICT) are well thought-out as possible assets for the development of socioeconomics in developing countries. Recent studies have exposed that higher infrastructure of telecommunication has facilitated means for underserved population development by various ways [1]. Among the existing applications of ICT, the digital library systems provide with better solutions and respond to a variety of unmet needs of research institutions, scientific communities, and developments. With the development of information technology, more and more information needs to be stored and disseminated, and the types and forms of information are more and more multiple. The mechanism of the traditional library is obviously unable to meet these needs [2]. Therefore, people put forward the idea of the digital library. The digital library is an electronic information storage, which can store a large number of various forms of information. Users can easily access it through the network to get these information, and its information storage and user access are not limited by geographical restrictions [3]. The digital library integrates all kinds of information, such as the data of information, storage management, and query and message posting, in which multimedia is involved so that information can be spread on the Internet to make information be used to maximum [4]. Through multimedia database technology and hypermedia technology and aiming at the characteristics of various media in the digital library, an effective and feasible management retrieval scheme in image retrieval, video-on-demand, and literature has been proposed [5, 6]. The digital library is an innovation based on the traditional library in the information era. It contains not only the traditional library functions, providing the corresponding service to the public, but also integrates some functions of other information resources (such as museums and archives); some of the features provide comprehensive public information service [7]. In other words, the digital library will become the public information center and hub of the future society [8].

Based on the parallel environment of the computer cluster, the communication in the parallel environment is controlled by the coordinator, so that the coordinator, collection machine, and query processor can complete the operation of distribution, load, and maintenance, which has high efficiency and saves much precious time, supports the digital library to meet user requirements effectively, and meets the digital library's performance requirements for data, and also, the key problem in the parallel algorithm has been solved [9]. The proposed study has considered the design and implementation of the human-computer interaction system in the parallel digital library system based on the neural network [10].

The paper is organized as follows. Section 2 shows the related work to the design and implementation of the human-computer interaction system in the parallel digital library system. Section 3 describes the methodology section of the paper. Section 4 briefly describes the results' analysis and discussion of the paper. The paper is concluded in Section 5.

2. Related Work

Our digital library workers have also done some work in the digital library, such as the research of the personalized active service system in the digital library system and query optimization algorithm. Although there are some works having been done in the field of digital library at home and abroad, the work is relatively scattered and preliminary [11]. The research that seemed the digital library as an independent and universal system tool has not been done and lacks of a comprehensive understanding of the digital library [12]. A lot of research is based on the use of the traditional database management system to implement the digital library system. It is noteworthy that there is no such report about the research of the parallel digital library [13]. Now, loading is a new research topic in the world. The research on parallel text loading is still at the initial stage, both at home and abroad. Although there are already some data-loading system prototypes at home and abroad, the research is mainly focused on the realization of two-dimensional relational data-loading technology of the single machine, and there is no parallel text-loading system. Moreover, the data loaded by these existing systems are mainly relational tables whose structure has been determined. No system can load fixed structure data according to the tables of the changing structure [14].

3. Methodology

3.1. Algorithm of the Collection Machine When the Digital Library System Adds Class. As the amount of data increases in the system, a new category needs to be added to the system at a certain stage. At this time, the classification system is dynamic. Which processor the new category should belong to is not predetermined, so we must first determine which processor the new category should belong to [15]. When the processor that a new class belonging to is determined, data related to the new class is sent to the corresponding processor instead of sending any message to other processors. When adding class, a special case is needed to be solved: the case of a minimum class on many data acquisition machines. In this way, each query processor should derive the thread according to the data acquisition machine, and these threads need to simultaneously receive data collected from different machines, more importantly, when the data is inserted into the same Oracle table, because multiple threads cannot simultaneously insert data to the same table, so it is necessary to with the help of intermediate files. These intermediate files are not merged into one big file until all threads input the data to the intermediate file, respectively; then, data merged is inserted into the Oracle table [16]. If the file on the query processor is a sharing file, such as many data acquisition machines input data into a certain file on the query processor at the same time, it also needs to input data to temporary files separately; finally, these temporary files are merged to form the final file [17].

The idea of the algorithm is that the data acquisition machine does not do any operation if the addition is not the smallest class, that is, it is not the leaf node in the classification pattern table. If the class is a leaf node, when the acquisition machine is receiving command of the adding class, it can also receive which query processor the new class should belong to, which is judged by the front-end machine. After the data is extracted, the extracted data is sent to the related query processor [18].

The function of the algorithm is to add new classes from the acquisition machine to the query processor. Input is a class of coding, processor number belonging to, and a string of number in the class composed of the number of text. The output returns 0 when succeeds. The process is to separate the parameters into class code, the processor number that the class should belong to, and the number of text in the class and store it in the array list of the structure [19]:

$$P(X_i, Y_i)P(Y_{i+1}|X_{i+1}) = P(X_i, Y_{i+1})P(Y_I|X_I). \quad (1)$$

Then, read the network configuration file; the processor that the new add class should belong to is read, connecting to the IP address of the processor. If a new type of class needing to add files is only one, send the file name and the contents of the file in order; if a new type of class to add the files is multiple, first find the first file. After loading the text, you can view the detailed steps of the processor operation [20]:

$$\mathrm{sim}(u_i, u_j) = \frac{\sum_{c\in I_{ij}}(R_{ic} - \overline{R})}{\sqrt{\sum_{c\in I_{ij}}(R_{ic} - \overline{R})^2}}. \quad (2)$$

For the analysis of algorithmic complexity, the number of minimum classes in the system is set as n, and the average amount of data under each minimum class is x, and the number of data acquisition machines is c. The number of the query processors is p. The time complexity of the worst case of the algorithm is o(x), and the time complexity in the best case is O(x/cP):

$$P(S) = \frac{\mathrm{Sim}(T_{rw}, S_{rw})}{\mathrm{Sim}(T_{uw}, S_{uw})}. \tag{3}$$

With a random process $\{X_n, n \in T\}$, if for any integer $n \in T$ and arbitrary $i_{0,} i_1, \cdots, i_n \in I$, the conditional probability satisfies

$$\begin{aligned} &P\{X_{n+1} = i_{n+1} * X_0 = i_0, X_1 = i_1, \ldots, X_n = i_n\} \\ &\quad = P\{X_{n+1} = i_{n+1} * X_n = i_n\}. \end{aligned} \tag{4}$$

Then, $\{X_n, n \in T\}$ would be called the MARKOV chain, which indicates that the probability of the system will be transferred to j at the moment of $n + 1$ when the system is in the state of i at time n; put P_{ij} in order to obtain the following matrix:

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ & & & \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix}. \tag{5}$$

This matrix is called the transition probability matrix. The state space I of any system can be decomposed into the following disjoint subsets:

$$I = C_1 + C_2 + \cdots + N. \tag{6}$$

In which, P_{ij} is composed of all the very return to the state set and $C_i\,(i = 1, 2, \cdots)$ is the reciprocal of the return of the state that is often composed of the closed set. If j is the aperiodic normal return state, then

$$\lim_{n \longrightarrow \infty} P_{ij}^{(n)} = \frac{1}{U_t}. \tag{7}$$

The variable U_j in this formula is the average return time of the state j.

We call the probability distribution $\{\pi_j, j \in I\}$ a smooth distribution of MARKOV chains, where I is the state space if it satisfies the following conditions:

$$\pi_i = \sum_{i \in I} \pi_i P_{ij}, \tag{8}$$

$$\sum_{i \in I} \pi_j, \pi_j > \; = 0. \tag{9}$$

If $\{\pi_j, j \in I\}$ is the smooth distribution of the MARKOV chain, then

$$\lim_{n \longrightarrow \infty} P_j(n) = \frac{1}{U_j} = \pi_j. \tag{10}$$

So from the data sequence changes, Start at time t_1 of the state, only the state at time t_n can predict the probability, the data sequence is divided into several states, recorded as $i_1, i_2, \cdots, i_n$, the probable transfer time be recorded as $t_1, t_2, \cdots, t_n$, the transition probability of the data sequence of the state at time i_k transitioning to step is expressed as $P_{ij}^{(m)}$:

$$P_{ij}^{(m)} = \frac{M_{ij}^{(m)}}{M_j}, \tag{11}$$

where M_{ij} is the number of times the state i_k transitions to the state i_y after m steps and M_i is the number of occurrences of the state i_k. As the uncertainty of the final state of the data sequence is steering, the last one data should be removed when calculating M_1.

3.2. The Maintenance of Data in the Parallel Digital Library System. The parallel data manipulation subsystem (PDOP)S provides basic query operations based on multiple data distribution strategies and parallel storage structures, such as one-dimensional data partition, multidimensional data partition, and compressed multidimensional array storage structure and attribute partitioning storage structure [21]. The parallelism of all algorithms is based on parallel data, and it is easy to be implemented in the computer parallel cluster environment. All the implementation environment of the parallel algorithm is the computer cluster parallel environment that is composed of several ordinary PC computers; through the parallel high-speed, network processors are connected together, one of them as the front-end processor and coordination machine. The random processor is used as the back-end data acquisition machine, and the arbitrary processor is the final processor of the system administrator [22]. The front-end machine does not store any data and is only used to receive operation requests from multiple users and coordinate the execution of commands by threads on each back-end machine. The data is transferred from the acquisition machine to store in the query processor on the back-end machine according to a certain data distribution method. The back-end machine is responsible for the specific execution. When the parallel algorithm is executed, the scheduling module on the front-end machine will coordinate all the back-end query processors and the operation execution modules on the data acquisition machine to work in parallel. This parallelism is achieved by performing the same operation on different data items, so it is a data parallel [23].

The implementation of the parallel data-loading algorithm is divided into two phases. The first phase is the data division phase. The second phase is the operation execution phase. In the first phase, the scheduling module first receives the execution information from the system administrator and calculates the data distribution strategy locally. The purpose of data distribution is to uniformly distribute data objects on a certain data acquisition machine to multiple query processors so that parallelization of the system can be fully realized during query processing. Data distribution is an important and active field in the research of the parallel database system at present. There are several methods of data distribution in a parallel database system. The one-dimensional data distribution method is the simplest way of data distribution. By partitioning the domain values of one attribute, the whole relationship is partitioned, and a set of subrelationships is obtained, and then, these subrelationships are distributed

among multiprocessors. At present, the one-dimensional data distribution method mainly includes Round—Robin, Hash, Range—partition, and Hybrid—Range—Partition. The one-dimensional data distribution method has a common problem: it is not able to effectively support queries with choice predicates on nonpartition attributes. In order to solve this problem, some multidimensional distribution methods have been put forward, including CMD method, ECC data distribution method, BM data distribution method, FX data distribution method, data distribution method based on the iHlbert curve, and BERD multidimensional data distribution method. Under the control of data distribution strategy, the threads of operation execution modules are derived from all the back-end query processors and data acquisition machines, and the operation execution information is broadcast to the threads of each back-end machine. After the operation information is received by the thread of the operation execution module on the data acquisition machine, the local data is transferred to the query processor.

Metadata is an important part of the digital library. The quality of metadata determines the quality management of the whole digital library. Metadata is stored in the metadata table. It is not only an important data material of the digital library to be used by workers but also can be used for querying of users so that the structure of the digital library can be better understood and its use level can be improved. The idea of the algorithm is that when modifying metadata, new metadata of modified metadata is stored in the Oracle tables of each back-end machine query processor, and new values in the table replace the old values of metadata tables. Because the new value table and metadata table both exist on all query processors, the algorithm only involves every query processor, and the algorithm on each query processor is exactly the same.

The algorithm modifies the back-end machine metadata information, and input is empty. The output returns 0 when it succeeds. Then, the name of the machine is obtained. Open the network configuration file and read it from the second records. If the name of the machine is the same as the name of a certain machine recorded of the network configuration file, the logical name of the machine is obtained. Using a cursor from the machine to select data for the table prepared for the update metadata, the condition is that the logical name of the machine is the same as the logical name of the machine. When data can be extracted from the cursor, update the title of the machine by using the following items, including document identifier, abstract, author, department, other information of the author, publisher, publication time, input time, page number, ISBN, and category code. Finally, return successfully.

4. Result Analysis and Discussion

4.1. Test One. The first experiment of the multithread parallel text-loading algorithm and serial algorithm in the query processor number is fixed, and the experimental results when the amount of data changing occurs is in the first experiment, and the back-end machine we use includes two sets of data acquisition machine and four sets of query processors; each back-end machine configuration is IG memory and 70G hard disk. The amount of data in this group of experiments is constantly changing, and the number of the back-end machines is fixed. The purpose of the experiment is to compare the efficiency of serial and parallel algorithms when the amount of data is increased. The following are the data of the test and the performance analysis table according to the test data. Performance analysis table is shown in Table 1.

Table 1 is the experimental data of the multithread parallel text-loading algorithm described in this article on 4 node machines. Row is the data volume, line is the parallel, serial different loading algorithms, and table content describes the running time of the algorithm. Comparison of parallelism and serialization when data quantity changes is shown in Figure 1.

We can see from the chart that when using the serial algorithm, although we use four queries for data processor loading at the same time, the query processor is working in the serial mode, that is to say, when the work of a query processor is over, the other one can work. In this way, the total load time will be the sum of the loading time of all processors, not only does the speed do not improve but also the communication between the processors should also be considered. When using parallel algorithms, four query processors will work at the same time, so the total time of loading will be the slowest processor loading time. As we can see in the graph, when the number of processors is constant, the parallel algorithm is much faster than the serial algorithm with the increase of data volume, and the total cost is about 1/3 of the serial algorithm.

4.2. Test Two. Experiment two is the experimental results of a multithread parallel text-loading algorithm and a serial algorithm when the data is fixed and the number of the query processor changes. When the amount of data to be loaded and the number of the query processors change, we use the 19.7 GB data, and the results are shown in the chart below. Line is the number of processors, row is parallel, serial different loading algorithms, and the table content describes the running time of the algorithm. Algorithm running time is shown in Table 2. Comparison of multiple threads and single thread when number of processors changes is shown in Figure 2.

The experimental results show that when using the serial text-loading algorithm, the more the number of the query processor is used, the lower the efficiency of the algorithm is. Since using the serial algorithm, the query processor still needs to load data one by one, and the latter query processor must wait until the last query processor is finished. And, the overhead of the algorithm also includes the communication time between the processors. But when using the parallel algorithm, the more the query processor is used, the higher the efficiency of the algorithm is because all the data query processor will be loaded at the same time, and the loading time is less than the communication time between processors, so increasing the communication overhead between

Table 1: Performance analysis table.

	624 mega bytes	1380 mega bytes	1710 mega bytes	1970 mega bytes
Parallelism serialization	5.65 minutes and 12.55 minutes 3886 mega bytes	21.5167 minutes and 38.05 minutes 5668 mega bytes	23.2833 minutes and 50.433 minutes 13877 mega bytes	31.9833 minutes and 117.45 minutes 19564 mega bytes
Parallelism serialization	49.67 minutes and 142.8 minutes	73.54 minutes and 216.433 minutes	135.85 minutes and 396.7 minutes	233.76 minutes and 723.4 minutes

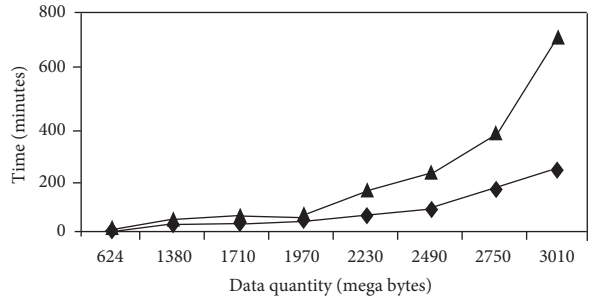

Figure 1: Comparison of parallelism and serialization when data quantity changes.

Table 2: Algorithm running time.

	Multithreads	Single thread
2	50.1	59.733
4	31.983	64.1
6	26.088	73.45
8	22.57	83.77
10	19.66	95.89

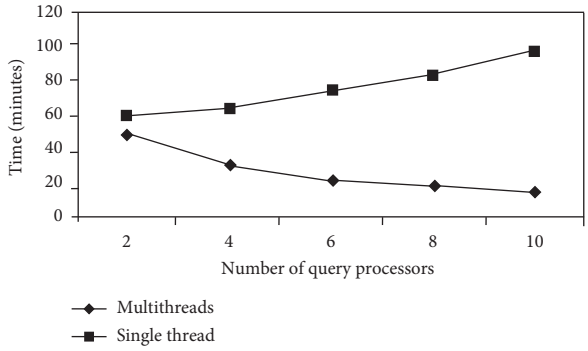

Figure 2: Comparison of multiple threads and single thread when number of processors changes.

processors is worth it. As we can see from the results, the efficiency of the parallel algorithm will be significantly improved when the number of processors increases.

4.3. Test Three. The experimental results of the multithreaded text parallel loading algorithm and single-thread loading algorithm are fixed in the number of query processors and data volume, and the number of data acquisition machine changes. When the amount of data to be loaded and the number of query processors are fixed and the number of data acquisition machines changes, we use 19.7 GB data and 4 query processors. The experimental results are shown below. Among them, line is the number of data acquisition machines, the row is whether each data acquisition machine and query processor can derive multiple threads and single-threaded loading algorithms, and tabular content describes the running time of the algorithm. The running time of the algorithm is shown in Table 3. Comparison of multiple threads and single thread when number of processors changes is shown in Figure 3.

The experimental results show that when the single threaded and multithreaded text parallel loading algorithms are used, the total execution time decreases with the increase of the number of data acquisition machines. When using a single thread loading algorithm, four sets of query processors are used; each query processor only derived a thread to receive the data of a data acquisition. Each data acquisition machine is derived from a thread to a query processor data; after a data acquisition machine finishes sending, the query processor receives the data from second sets; when using the multithread loading algorithm, each query processor derived M threads according to the data acquisition machine number; each data acquisition machine derived N threads according to the number of query processor so that each data acquisition machine with N threads simultaneously sends data to N query processors. And, each query processor with M threads simultaneously receives M data acquisition data, so the multithread loading algorithm will be much faster than the single thread algorithm. The running time prediction of the algorithm is shown in Table 4. Processor number prediction results were compared and are shown in Figure 4.

Table 3: The running time of the algorithm.

	Multithreads	Single thread
1	31.983	64.1
2	26.76	57.8
4	19.85	50.77
6	15.7	41.99
8	14.39	34.34
10	9.6	29.37

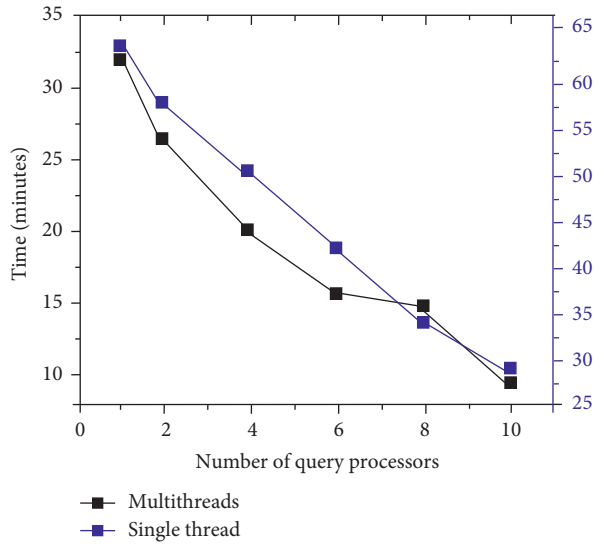

Figure 3: Comparison of multiple threads and single thread when number of processors changes.

Table 4: The running time prediction of the algorithm.

	Multithreads	Single thread
1	33	69.22
2	28.69	62.30
4	25.14	56.15
6	15.69	45.36
8	18.69	36.33
10	18.66	36.21

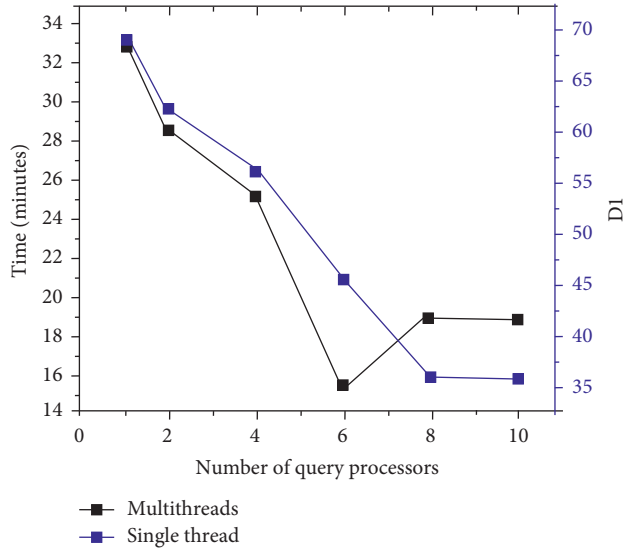

Figure 4: Processor number prediction results were compared.

5. Conclusion

With the development of digital library technology, the parallel database system has become the key tool for efficient information processing in the digital library system. Data-loading operation is a significant part of the digital library, which is well known that the data-loading operations are time-consuming. Data loading, especially the loading of parallel text data, is a new field of research. In this paper, a novel data operation algorithm based on a new parallel digital library is proposed, and all the data operation algorithms are implemented in the prototype system. A multithread parallel text data-loading operation and maintenance algorithm has been proposed in this paper, which has no research about it up to now. A large number of experiments show that the algorithm proposed in this paper is more efficient than the existing algorithms and has high practical value. Considering the performance and price ratio, the parallel algorithm has high practical value and benefit. To sum up, the author thinks it is still a large area having many problems that need to be solved. There are many works to do, hoping that experts and scholars will pay enough attention to it.

Data Availability

The datasets used and/or analyzed during the current study are available from the corresponding author upon reasonable request.

Additional Points

This is a research involving human participants and/or animals.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] N. Yu, S. Li, Y. Zhao et al., "Design and implementation of a dexterous human-robot interaction system based on haptic shared control (in Chinese)," *Chinese Journal of Scientific Instrument*, vol. 38, no. 3, pp. 602–611, 2017.

[2] Y. Park, H. Yang, T. Dinh et al., "Design and implementation of a container-based virtual client architecture for interactive digital signage systems," *International Journal of Distributed Sensor Networks*, vol. 13, no. 7, p. 155, 2017.

[3] Y. J. Zhang, X. J. Meng, and G. Wang, "Design and implementation of 3D electronic sand table system based on gesture interaction," *Command Control & Simulation*, vol. 49, no. 21, pp. 706–712, 2016.

[4] W. K. Liou, K. K. Bhagat, and C. Y. C. Y. Chang, "The design, implementation, and evaluation of a digital interactive globe system integrated into an earth science course," *Educational Technology Research & Development*, vol. 53, no. 11, pp. 1201–1209, 2018.

[5] R. Habel, F. Silber-Chaussumier, F. Irigoin et al., "Combining data and computation distribution directives for hybrid parallel programming: a transformation system,"

International Journal of Parallel Programming, vol. 44, no. 6, pp. 1–28, 2016.

[6] D. Huang, D. Han, J. Wang et al., "Achieving load balance for parallel data access on distributed file systems," *IEEE Transactions on Computers*, vol. 108, no. 99, p. 1, 2017.

[7] M. Hermanns and E. Cramer, "Likelihood inference for the component lifetime distribution based on progressively censored parallel systems data," *Journal of Statistical Computation and Simulation*, vol. 87, no. 3, pp. 607–630, 2016.

[8] T. C. Pan, P. Flick, C. Jain et al., "Kmerind: a exible parallel library for k-mer indexing of biological sequences on distributed memory systems," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 16, 2019.

[9] X. Shen, L. Yi, Y. Yi et al., "Dynamic identifying protein functional modules based on adaptive density modularity in protein–protein interaction networks," *BMC Bioinform*, vol. 16, no. 12, pp. 5–12, 2015b.

[10] B. P. Tu, A. Kudlicki, M. Rowicka et al., "Logic of the yeast metabolic cycle: temporal compartmentalization of cellular processes," *Science*, vol. 310, no. 5751, pp. 1152–1158, 2005.

[11] S. Wang and W. Guo, "Sparse multigraph embedding for multimodal feature representation," *IEEE Transactions on Multimedia*, vol. 19, no. 7, pp. 1454–1466, 2017a.

[12] S. Wang and W. Guo, "Robust co-clustering via dual local learning and high-order matrix factorization," *Knowledge-Based Systems*, vol. 138, pp. 176–187, 2017b.

[13] J. Wang, X. Peng, M. Li, and Y. Pan, "Construction and application of dynamic protein interaction network based on time course gene expression data," *Proteomics*, vol. 13, no. 2, pp. 301–312, 2013.

[14] J. Wang, J. Liang, X. Zhao, and W. Zheng, "Overlapping protein complexes detection algorithm based on assortativity in PPI network," *Computer Sciences*, vol. 46, pp. 294–300, 2019.

[15] X. Xiao, J. Ji, and C. Yang, "Fireworks algorithm for functional module detection in protein–protein interaction networks," *Journal of Harbin Institute of Technology*, vol. 51, pp. 57–66, 2019.

[16] Y. Zhang, K. Jia, and A. Zhang, "Consistent protein functional module detection from multi-view of biological data," *Acta Electronic Sinica*, vol. 42, no. 12, pp. 2337–2344, 2014.

[17] Y. Zhang, H. Lin, Z. Yang et al., "A method for predicting protein complex in dynamic PPI networks," *BMC Bioinformatics*, vol. 17, no. 7, pp. 229–239, 2016.

[18] B. Zhao, H. Xiong, W. Ni et al., "Improved weighted-network based algorithm for predicting protein complexes," *Computer Science*, vol. 41, no. 6, pp. 231–234, 2014a.

[19] B. Zhao, J. Wang, M. Li et al., "Detecting protein complexes based on uncertain graph model," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 11, no. 3, pp. 486–497, 2014b.

[20] B. Zhao, X. Li, S. Hu et al., "Prediction of protein functions based on essential functional modules mining," *Acta Automatica Sinica*, vol. 44, no. 1, pp. 183–192, 2018.

[21] W. P. Zheng, J. Y. Li, and J. Wang, "Protein complex recognition algorithm based on genetic algorithm," *Journal of Computer Science and Technology*, vol. 12, no. 5, pp. 794–803, 2018.

[22] J. Zhong, J. Wang, W. Peng et al., "A feature selection method for prediction essential protein," *Tsinghua Science and Technology*, vol. 7, no. 10, pp. 491–499, 2015.

[23] W. Zhu, W. Guo, Z. Yu, and H. Xiong, "Multitask allocation to heterogeneous participants in mobile crowd sensing," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 7218061, 10 pages, 2018.