International Journal of Knowledge and Innovation Studies
https://www.acadlore.com/journals/IJKIS

Development and Evaluation of a Parallel K-means Algorithm for Big Data Analysis in Google MapReduce Environment

Junwei Zhao[1], Xuexu Yuan[1*], Qingtao Hou[2, 3*], Hanyu Gao[4], Chunyu Gao[4], Yuanyuan Zhang[1*]

[1] Department of Computer Engineering, North China Institute of Science and Technology, 101601 Yanjiao Beijing-East, China
[2] Inspur Software Co., Ltd., 271002 Tai'an, China
[3] Shandong Provincial Key Laboratory of Artificial Intelligence Technology for Public Service, 271002 Tai'an, China
[4] SanHe ChenHao Geological Exploration Technology Service Co., Ltd, 065201 Yanjiao Beijing-East, China

[*] Correspondence: Xuexu Yuan (yuanxuexu2023@163.com); Qingtao Hou (hqt119119@163.com); Yuanyuan Zhang (zhangyuanyuan@ncist.edu.cn)

Received: 06-20-2024 **Revised:** 08-05-2024 **Accepted:** 08-14-2024

Citation: J. W. Zhao, X. X. Yuan, Q. T. Hou, H. Y. Gao, C. Y. Gao and Y. Y. Zhang, "Development and evaluation of a parallel K-means algorithm for big data analysis in Google MapReduce environment," *Int J. Knowl. Innov Stud.*, vol. 2, no. 3, pp. 147–154, 2024. https://doi.org/10.56578/ijkis020303.

Abstract: The challenge of executing iterative big data analysis algorithms within the Google Cloud MapReduce environment has been addressed by developing a parallel K-means algorithm capable of leveraging the distributed computing power of the platform. Traditional K-means, which requires iterative steps, is adapted into a parallel version using MapReduce to enhance computational efficiency. This parallel algorithm is structured into multiple super-steps, each of which executes in parallel but is processed sequentially across super-steps. Each super-step corresponds to one iteration of the serial K-means algorithm, with parallel computation carried out at each node to determine the mean of each cluster center. Experimental evaluations have demonstrated that the parallel K-means algorithm performs effectively and accurately. Notably, for a dataset of 450 water samples, a parallel speedup factor of 20.8 was achieved, significantly reducing the time required for data analysis. This substantial reduction in processing time is critical in time-sensitive applications, such as coal mine rescue operations, where quick decision-making is essential. The results indicate that the proposed parallel K-means algorithm is both a feasible and efficient solution for handling large-scale datasets within cloud environments, providing substantial benefits in both computational speed and practical application.

Keywords: Parallel computing; Cloud computing; K-means algorithm; Super-step

1 Introduction

The primary characteristic of big data is its enormous volume. The scale of data that needs to be processed in a timely manner to extract useful information has escalated from the TB level to the PB and even EB levels [1–3]. Secondly, big data holds significant commercial value, but its value density is low—individual data points have little value on their own. It is only when large volumes of data are aggregated and processed that the value of big data analysis becomes evident, as it can help predict future trends based on historical data [4]. This requires appropriate hardware platforms and software algorithms for processing big data. Many algorithms used in big data analysis are complex and consist of multiple iterative steps [5–7]. Although Google Cloud's MapReduce environment is a widely used hardware platform for big data analysis, it cannot directly run data analysis algorithms with iterative steps [8–13]. To solve this problem, this paper uses the K-means algorithm as an example, upgrading the serial K-means algorithm to a parallel K-means algorithm that can run in Google Cloud's MapReduce environment.

MapReduce abstracts the parallel computing process running on large-scale clusters into two functions, Map and Reduce, adopting a "divide and conquer" strategy to enable parallel computation [14–17]. A large-scale dataset stored in a distributed file system is split into many independent chunks (splits), which can be processed in parallel by multiple Map tasks [18]. There is no communication between different Map tasks, and no information exchange

occurs between different Reduce tasks. Users cannot explicitly send messages from one machine to another; all data exchanges are handled by the MapReduce framework itself. Due to this performance, MapReduce is only suitable for relatively simple parallel computations, and it struggles with more complex algorithms, such as the K-means algorithm, which requires multiple iterations [19, 20]. To address this issue, this paper proposes a new explicit parallel K-means algorithm that can run in the MapReduce environment and produce correct results. This parallel algorithm is composed of multiple super-steps, each involving parallel computation and completed by a single Map and Reduce function. However, the super-steps are executed sequentially. One super-step corresponds to one iteration of the serial K-means algorithm. Within each super-step, each computational node (processor, worker) concurrently computes the average value of a cluster center.

In Google's MapReduce environment, this paper processes the initial 47,000 coal mine water sample test data [21–24]. After extraction, cleaning, and integration, high-quality data consisting of 450 water samples and 3,600 data points were obtained. Subsequently, the parallel K-means algorithm developed in this paper was applied to mine and analyze this data. Both theoretical and experimental results demonstrate that the parallel K-means algorithm yields the same results as the serial algorithm, with a parallel speedup ratio of 20.8. This significantly increases the processing speed of water sample analysis, thus saving valuable time for coal mine rescue operations. The following sections will introduce this new parallel algorithm.

2 Serial K-means Algorithm

Clustering is a method of grouping two observation data points based on the similarity calculated from their distance (without labeled samples). The K-means algorithm, also known as the K-average algorithm or K-means algorithm, is one of the most widely used clustering algorithms [25]. The K-means algorithm uses K as a parameter to divide N samples (objects) into K classes (clusters), such that the similarity within each class is high, while the similarity between classes is low. The similarity is calculated based on the average value of the objects (samples) within a class. First, K samples are randomly selected, each of which initially represents the average value or center of a class. For each remaining sample, based on the distance to each class (cluster) center, it is assigned to the nearest class. Then, the average value of each class is recalculated. This process is repeated until the criterion function E converges, that is, until the generated result classes are as compact and independent as possible [26]. The criterion is as follows:

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} \left| X - \bar{X}_i \right|^2$$

where, E is the total sum of squared errors for all samples to be classified, x is a point in the space representing a given data sample, and $\bar{X}$ is the average (center) of class C_i. The number of input classes k and the dataset with n samples are given, and the output consists of k classes that minimize the squared error criterion.

In summary, the K-means algorithm process is as follows:

K-means Algorithm Process

Step 1: Randomly select k objects from the n data objects as the initial cluster centers.

Step 2: Calculate the distance between each object and these center objects (mean of the cluster), and reassign each object to the corresponding cluster based on the smallest distance.

Step 3: Recalculate the mean (center object) for each cluster (if there is any change).

Step 4: Repeat steps 2 and 3 until no further changes occur in the clusters, i.e., until the squared error criterion is minimized.

3 Google Cloud MapReduce

The computing model of cloud computing is a programmable parallel computing framework that requires high scalability and fault tolerance support. MapReduce is a parallel programming model proposed by Google, which runs on top of the Google File System (GFS) [1, 2]. The design philosophy behind MapReduce is to divide and conquer the problem. The original data source of the user is first split into chunks, which are then assigned to different Map tasks for processing. The MapReduce framework adopts a Master/Slave architecture, consisting of one Master and several Slaves. The JobTracker runs on the Master, and the TaskTracker runs on the Slaves.

As shown in Figure 1, MapReduce follows a "divide and conquer" strategy [27]. A large-scale dataset stored in a distributed file system is split into many independent splits, which can be processed in parallel by multiple Map tasks. The specific steps are as follows:

(1) Data segmentation;

(2) Master assigns Map tasks;

(3) Workers assigned with Map tasks read and process the relevant splits;

(4) The Master coordinates the assignment of Reducers to fetch data from the appropriate Mappers, during which the shuffle process occurs, including a sorting step by key;

(5) The Reducer reads and processes the Value list corresponding to each key;

(6) The Reducer writes the processed data into the HDFS output file.

The shortcomings of Google's MapReduce are as follows:

(1) Programmers must manually and explicitly specify the parallelism of the computation tasks before the Map tasks can be assigned;

(2) It cannot automatically perform iterative computations;

(3) There is no communication between different Map tasks;

(4) There is no information exchange between different Reduce tasks;

(5) Users cannot explicitly send messages from one machine to another; all data exchanges are handled by the MapReduce framework itself.

In summary, the parallel MapReduce computing platform provided by Google Cloud is a basic and simple parallel computing platform. To perform more advanced parallel computing tasks on this platform, further work is required.

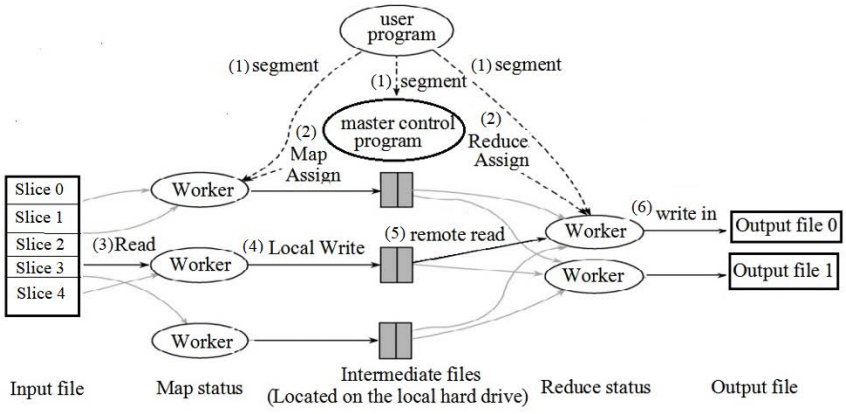

Figure 1. Google cloud MapReduce parallel computing platform working principle

4 Development Process and Pseudocode of Parallel K-means Algorithm in MapReduce Framework

4.1 Parallel K-means Algorithm Steps

To address the issue where the widely used K-means algorithm in big data analysis fails to yield correct results when executed in the Google MapReduce environment, a new explicit parallel K-means algorithm has been proposed that can run correctly in the MapReduce environment. The parallel algorithm is composed of multiple super-steps. Each super-step involves parallel computation, but the super-steps themselves are executed sequentially. A super-step corresponds to one iteration of the serial K-means algorithm. Within each super-step, each computational node (processor, worker) concurrently computes the mean value of a cluster center. Initially, the K cluster centers are selected randomly. Each computational node then calculates the distance between each sample point and the current new cluster centers, and assigns each sample to the nearest cluster based on the distance. During the synchronization phase, the computational nodes communicate with each other by exchanging the newly computed cluster center averages and the classification information of each sample, ensuring that each node has the same updated classification data and information. The super-step computation is repeated multiple times until the cluster centers no longer exhibit significant changes. The Map function is used to allocate parallel tasks, and the Reduce function is used to aggregate the computation results.

In summary, the steps of the parallel K-means algorithm are as follows:

Step 1: Randomly select k samples from n data samples as the initial cluster centers.

Step 2: Parallel classification. Each computational node (processor, worker) concurrently computes the mean value for one cluster center. It calculates the distance between each sample and these center samples and reassigns the samples to the nearest center based on the smallest distance. This step is executed in parallel on each computational node (processor, worker).

Step 3: Parallel computation of class means. Recalculate the mean (center sample) for each (changed) cluster. This step is executed in parallel on each computational node (processor, worker).

Step 4: During the synchronization phase, the computational nodes communicate with each other by exchanging the newly computed cluster center averages and the classification information of each sample, ensuring that each computational node has the same updated classification data and information.

Step 5: After completing one super-step of parallel computation, repeat steps 2, 3, and 4 until the clusters no longer change, that is, until the squared error criterion is minimized.

4.2 Graphical Representation of Parallel K-means Algorithm Steps

The process of the parallel K-means algorithm can be graphically represented as shown in Figure 2.

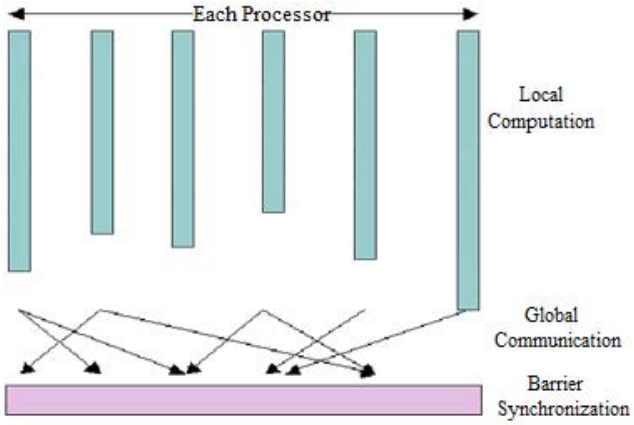

Figure 2. One "super-step" of the parallel K-means algorithm

The operation of the parallel K-means algorithm introduces the concept of "super-step". Its execution is based on super-steps, which are the fundamental units of parallel computation in the algorithm. The parallel K-means algorithm consists of several super-steps, and each super-step is further divided into the following three steps:

(1) Local Computation by Processing Nodes: Each processing node concurrently computes the mean value of one cluster center. There are k cluster centers, so k processing nodes are required to participate in the parallel computation.

(2) Local Computation with Remote Memory Access: The processing nodes perform their local computations using the information in their local memory. During this phase, the nodes may asynchronously issue remote memory access and message-passing operations. However, these operations will not be executed immediately. The communication network will handle the operations issued in the previous step.

(3) Global Synchronization (Barrier Synchronization): Finally, all processing nodes perform global barrier synchronization. The communication operations of the current super-step become effective after barrier synchronization.

4.3 Pseudocode for Parallel K-means Algorithm in MapReduce Framework

The pseudocode for the parallel K-means algorithm is as follows:

S1 assign initial value for means; /* Randomly select k samples as the initial cluster centers */
S2 REPEAT /* Repeat parallel computation within each super-step */
forall

$$(i = 1; i \leq k; i++)$$

/* Start parallel computation within each super-step */
{
S3 FOR j=1 to n DO assign each to the cluster which has the closest mean;
/* Assign each sample to the most similar cluster based on the mean of the cluster */
S4 FOR i=1 to k DO

$$X_i = |C_i| \sum_{x \in C_i} X$$

/* Update the mean of each cluster by computing the average of the samples in that cluster */
S5 Compute

$$\sum_{x \in C_i} \left| X - \bar{X}_i \right|$$

/* Compute the criterion function E */
} /* End parallel computation within each super-step */
S6 UNTIL no significant changes occur in E.

4.4 More Detailed Pseudocode for Parallel K-means Algorithm in MapReduce Framework

In each super-step, one iteration of the K-means algorithm is implemented using MapReduce. The Map and Reduce functions are as follows:

(1) Map Process

Each worker compute node loads the sample data set, where the input to the Map is <key, value>, with the key being the sample row number (i.e., the sample ID), and the value being the sample's attributes and label (i.e., the class it belongs to out of the k classes). The output of the Map is <key, list(value)>, where the key is the sample row number (i.e., the sample ID), and the value is the sample's label (i.e., the class it belongs to out of the k classes) and the distance (the distance from the sample to each of the k class centers). The pseudocode for the Map function is as follows:

```
for i in test\_data            /* For each sample to be classified */
    for j in cluster\_center\_data  /* For each cluster center */
                Extract label L from sample i
                Compute the distance D between sample i and cluster center j
           context.write(sample row number (ID), vector (L, D))
    end for
end for
```

where, L represents the cluster label, namely the cluster ID; and D represents the distance between the sample and the cluster center L.

(2) Reduce Process

For the input key-value pairs with the same key, sort the (L, D) pairs by D, take all k label-L (i.e., the distances of a sample to the k cluster centers), and calculate the minimum distance. The result for this key is the sample's assigned cluster center ID, which is the cluster to which the sample belongs.

5 Experimental Results

5.1 Size, Characteristics, and Applicability of Experimental Data

The initial dataset to be analyzed consisted of 47,000 data points. The most prominent feature of these data was incompleteness, meaning the data contained some missing information and erroneous data, i.e., dirty data. After extraction, cleaning, and integration, a high-quality dataset of 3,600 data points was obtained. This dataset was then mined and analyzed using the parallel K-means algorithm developed in this paper within the MapReduce environment.

The size of the high-quality experimental dataset used for mining and analysis was 3,600 data points, i.e., 450 water samples ×8 analytical indicators = 3,600 data points.

Applicability of the Experimental Data: To ensure the dataset's representativeness and broad applicability, data from 35 representative and widely applicable coal mines, owned by the Lu'an Group, were selected [19–27]. These data represented water quality chemical analysis indicators from various mine source waters. The dataset consists of four types of water sources, which are: water in ground spring, wate in sandstone aquifers, water in limestone aquifers of the Taiyuan series, and water in limestone aquifers of the ordovician system. These four water types are typical sources of water-related hazards encountered in coal mining operations in China, making them both representative and widely applicable.

The characteristics of the experimental data: incompleteness (dirty data), timeliness (quick analysis results are required), reliability (high practical value), complexity (the knowledge contained is difficult to discover), authenticity (the data reflects reality), and large data volume, approximately 36,000 raw data points [20–27].

The experimental data was collected from various sources of water samples, and the chemical indicators of water quality were analyzed to establish a representative water sample database. The indicators include: K^+, $\mathrm{M_g^{2+}}$, $\mathrm{C_a^{2+}}$, $\mathrm{Cl^-}$, $\mathrm{SO_4^{2-}}$, $\mathrm{HCO_3^-}$, Total hardness, and pH value [19].

5.2 Experimental Results

In the Google MapReduce environment, the initial 47,000 experimental data points were processed through extraction, cleaning, and integration to produce a high-quality dataset of 3,600 data points. This dataset was then mined and analyzed using the parallel K-means algorithm developed in this paper, with the task of classifying the 450 water samples. Each water sample's class was known. During the experiment, it was assumed that the class of each water sample was unknown in order to test whether the parallel algorithm could correctly classify the 450 water samples. The accuracy of the classification results was 99% (i.e., the classification accuracy of the parallel K-means algorithm), thus proving that the parallel K-means algorithm developed in this paper is correct.

Table 1. Analysis results and data processing for sample water samples (Partial listing due to space constraints)

Sampling Point	Correctness of Analysis	M_g^{2+}	C_a^{2+}	Cl^-	SO_4^{2-}	pH
Water_1 in ground spring	Correct	74.8	60.0	145.0	719.5	7.9
Water_2 in ground spring	Correct	90.9	206.8	384.8	530.6	7.6
Water_1 in sandstone aquifers	Correct	72.4	80.0	124.5	540.0	8.2
Water_2 in sandstone aquifers	Correct	74.9	186.8	301.3	212.3	8.0
Water_1 in limestone aquifers of the Taiyuan series	Correct	5.1	17.2	126.5	1,779.8	7.9
Water_2 in limestone aquifers of the Taiyuan series	Correct	62.7	117.2	195.9	1,843.6	7.6
Water_1 in limestone aquifers of the ordovician system	Correct	13.9	34.3	978.3	105.2	7.9
Water_2 in limestone aquifers of the ordovician system	Correct	7.6	14.7	821.5	101.3	8.4

Table 1 shows part of the experimental data and analysis results (due to space limitations, only a selection of typical representative data is listed in Table 1).

Figure 3 is the visualization of the classification result for these 450 water samples. Since this experiment used 8 water quality test indicators, it corresponds to an 8-dimensional space. Plotting the classification result in an 8-dimensional space is difficult, so principal component analysis (PCA) was used in this paper to take the first two principal components and plot the classification result in a 2-dimensional space, as shown in Figure 3. These two principal components explain 76.4% of the information of the original 8 water quality test indicators (also referred to as variables or features). From Figure 3, it can be seen that the parallel K-means algorithm developed in this paper, under the Google MapReduce environment, divides the 450 water samples into four categories. This classification result is consistent with the actual situation, which also proves that the parallel K-means algorithm developed in this paper is correct.

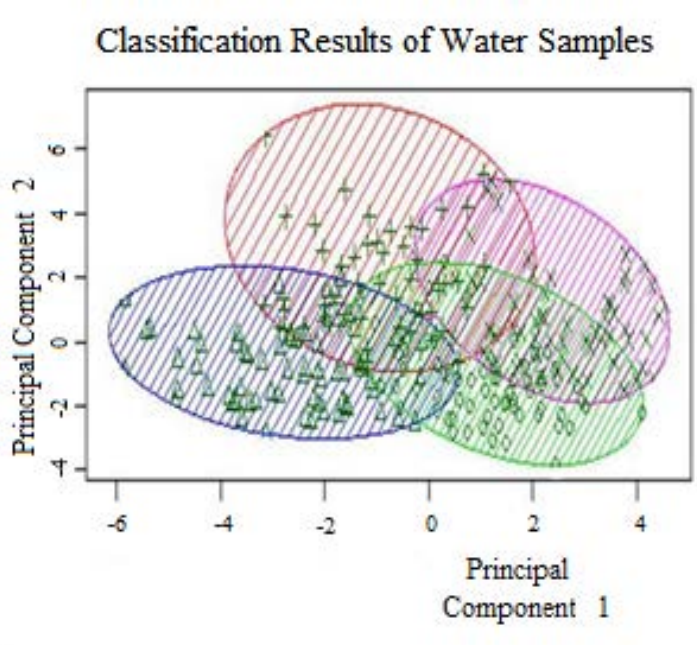

Figure 3. 2D space classification results for 450 experimental water samples

Note: These two principal components explain (that is, represent) 76.4% of the information of the original 8 indicators.

As shown in Table 2, the parallel K-means algorithm developed in this paper achieved a speedup ratio of 20.8 when analyzing 3,600 data points from 450 water samples. This significantly improved the speed of water sample analysis, saving analysis time and providing more time for coal mine emergency rescue operations.

Table 2. Performance comparison between parallel K-means and serial K-means

Sampling Point	Analysis Correctness Rate	Computation Time	Speedup Ratio
K-means Parallel K-means	99%	13 minutes	20.8
Serial K-means (single machine, Lenovo)	99%	270 minutes	

6 Conclusion

Many algorithms used in big data analysis are quite complex and consist of multiple iterative steps [17–19]. Although the Google Cloud MapReduce environment is a widely used big data analysis platform, it cannot directly run algorithms with iterative steps. To address this issue, this paper takes the K-means algorithm as an example and upgrades the serial K-means algorithm to a parallel K-means algorithm that can run in the Google Cloud MapReduce environment.

This paper proposes an explicit parallel K-means algorithm that can produce correct results within the MapReduce environment. The parallel algorithm is composed of multiple super-steps. Within each super-step, parallel computation is performed, but the super-steps themselves are executed serially. A super-step corresponds to one iteration of the serial K-means algorithm. During each super-step, each compute node (processor, worker) parallelly computes the mean value for a cluster center. The parallel computation within each super-step is achieved through Map and Reduce functions. Both theoretical analysis and experimental results show that the parallel K-means algorithm developed in this paper can run correctly within the Google MapReduce environment. Furthermore, the results of the parallel K-means algorithm are consistent with those of the serial K-means algorithm. The speedup ratio achieved for analyzing 3,600 data points from 450 water samples is 20.8, significantly improving the analysis speed and saving time for coal mine rescue operations.

The method proposed in this paper is not only suitable for the parallel K-means algorithm but also applicable to many other big data analysis algorithms that consist of multiple iterative steps. The parallel K-means algorithm presented here has broad application value in fields such as facial recognition, fingerprint recognition, language models, speech recognition, artificial intelligence, and public safety identification for criminal detection.

Data Availability

The data supporting our research results are deposited in H. Y. Gao and S. C. Yu at 200600737ysc@ncist.edu.cn (email).

Conflict of Interests

The authors declare that they have no conflicts of interest.

References

[1] "Apache hadoop." http://hadoop.apache.org/

[2] "Apache hive." https://hive.apache.org/

[3] "Big data." http://en.wikipedia.org/wiki/Big_data.2014

[4] D. Howe, M. Costanzo, P. Fey, T. Gojobori, L. Hannick, W. Hide, D. P. Hill, R. Kania, M. Schaeffer, S. St Pierre, S. Twigger, O. White, and S. Y. Rhee, "The future of biocuration," *Nature*, vol. 455, pp. 47–50, 2008. https://doi.org/10.1038/455047a

[5] "IBM aglets software development kit." http://www.trl.ibm.co.jp/aglets

[6] J. C. Zhou, H. Y. Zhang, W. L. Zha, and Y. B. Chen, "User-aware resource provision policy for cloud computing," *J. Comput. Res. Dev.*, vol. 51, no. 5, pp. 1108–1119, 2014.

[7] S. Ghemawat, H. Gobioff, and S. T. Leung, "The Google file system," in *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, Bolton Landing, NY, USA, 2003, pp. 29–43. https://doi.org/10.1145/945445.945450

[8] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *ACM SIGACT News*, vol. 33, no. 2, pp. 51–59, 2002. https://doi.org/10.1145/564585.564601

[9] P. A. Slcigh, P. H. Gaskell, M. Berzins, and N. G. Wright, "An unstructured finite-volume algorithm for predicting flow in rivers and estuaries," *Comput. Fluids*, vol. 27, no. 4, pp. 479–508, 1998. https://doi.org/10.1016/S0045-7930(97)00071-6

[10] H. R. Seddighi and A. L. Theocharous, "A model of tourism destination choice: A theoretical and empirical analysis," *Tour. Manag.*, vol. 23, no. 5, pp. 475–487, 2002. https://doi.org/10.1016/S0261-5177(02)00012-2

[11] R. O. Reid and B. R. Bodine, "Numerical model for storm surges in Galveston Bay," *J. Waterw. Harb. Div.*, vol. 94, no. 1, pp. 33–57, 1968. https://doi.org/10.1061/JWHEAU.0000553

[12] G. A. Morgan and O. V. Griego, *Easy Use and Interpretation of SPSS for Windows: Answering Research Questions with Statistics*. New Jersey: Lawrence Erlbaum Associations, Inc., 1998.

[13] D. Cramer, *Introducing Statistics for Social Research: Step-by-Step Calculations and Computer Techniques Using SPSS*. United States: Routledge, 1994.

[14] J. J. K. Ó Ruanaidh, W. J. Dowling, and F. M. Boland, "Watermarking digital images for copyright protection," *IEEE Proc. Vis. Image Signal Process.*, vol. 143, pp. 250–256, 1996.

[15] R. Barnett and D. E. Pearson, "Frequency mode LR attack operator for digitally watermarking images," *Electron. Lett.*, vol. 34, no. 19, pp. 1837–1839, 1998. https://doi.org/10.1049/el:19981323

[16] "Amazon," 2015. http://aws.amazon.com/cn/documentation/

[17] S. F. A, N. N. Liu, S. C. Yu, G. X. Yu, and S. C. Yu, "A new method to improve the speed and accuracy of intruder detection in cloud computing environment," *Comput. Appl. Softw.*, vol. 38, no. 4, pp. 311–317, 2021.

[18] F. Li, L. M. Sun, J. X. Wei, J. L. Huang, S. C. Yu, H. Guo, and X. J. Wang, "Research on automatic identification method of mine water source based on artificial intelligence," *J. North China Inst. Sci. Technol.*, vol. 10, no. 2, pp. 17–21, 2013.

[19] J. J. Li, Y. Xue, G. X. Yu, G. L. Yu, and S. C. Yu, "Research on method of automatic recognition of water sources based on support vector machine," *J. North China Inst. Sci. Technol.*, vol. 12, no. 2, pp. 25–29, 2015.

[20] N. B. Yin, J. Hao, and S. C. Yu, "Research on establishment method of mine water source identification model base based on artificial intelligence," *J. North China Inst. Sci. Technol.*, vol. 14, no. 4, pp. 24–28, 2017.

[21] S. F. A, N. N. Liu, S. C. Yu, and Q. Shi, "Research on relative technologies of automatic recognition of water sources based on back propagation neural network improved by immune algorithm," *J. North China Inst. Sci. Technol.*, vol. 14, no. 1, pp. 34–39, 2017.

[22] H. Q. Chen, Y. Zhai, S. L. Hui, X. M. Liu, and X. Y. Wang, "Evaluation on water inrush danger from aquifer under seam floor and water inrush prevention and control countermeasures," *Coal Sci. Technol.*, vol. 39, no. 7, pp. 122–115, 2011. https://doi.org/10.13199/j.cst.2011.07.118.chenhq.028

[23] Y. L. Fu, Z. J. Luo, and Z. H. Wang, "Application of cluster analysis and fuzzy comprehensive judgment in geological environment quality evaluation," *Coal Geol. Explor.*, vol. 27, no. 6, pp. 47–50, 1999.

[24] W. J. Li, R. L. An, Y. F. Fang, and J. L. Han, "Coalmine water inrush source analysis based on biasing weighting method fuzzy integrated evaluation," *Coal Geol. China*, vol. 24, no. 4, pp. 35–37, 2012.

[25] B. S. Yang, C. S. Wang, and C. Y. Yan, "Causes of water inrush in Qidong Coal Mine," *Coal Geol. Explor.*, vol. 31, no. 1, pp. 41–43, 2003.

[26] W. D. Gao, Y. D. He, and X. S. Li, "Application of hydrochemical method in judging the source of water inrush in mines," *Min. Saf. Environ. Prot.*, vol. 28, no. 5, pp. 44–45, 2011.

[27] J. Zhou, X. Z. Shi, and H. Y. Wang, "Water-bursting source determination of mine based on distance discriminant analysis model," *J. Cent. South Univ.*, vol. 35, no. 2, pp. 278–282, 2010. https://doi.org/10.13225/j.cnki.jccs.2010.02.025