Hindawi
Complexity
Volume 2021, Article ID 9989471, 8 pages
https://doi.org/10.1155/2021/9989471

Research Article

A Parallel Attribute Reduction Method Based on Classification

Deguang Li [1] **and Zhanyou Cui** [2]

[1]*School of Information Technology, Luoyang Normal University, Luoyang 471934, China*
[2]*College of Mechanical and Electrical Engineering, Zhengzhou Institute of Industrial Technology, Zhengzhou 451150, China*

Correspondence should be addressed to Zhanyou Cui; zhanyou_cui@126.com

Received 10 March 2021; Revised 29 March 2021; Accepted 30 March 2021; Published 10 April 2021

Academic Editor: Danilo Comminiello

Parallel processing as a method to improve computer performance has become a development trend. Based on rough set theory and divide-and-conquer idea of knowledge reduction, this paper proposes a classification method that supports parallel attribute reduction processing, the method makes the relative positive domain which needs to be calculated repeatedly independent, and the independent relative positive domain calculation could be processed in parallel; thus, attribute reduction could be handled in parallel based on this classification method. Finally, the proposed algorithm and the traditional algorithm are analyzed and compared by experiments, and the results show that the proposed method in this paper has more advantages in time efficiency, which proves that the method could improve the processing efficiency of attribute reduction and makes it more suitable for massive data sets.

1. Introduction

With the rapid development of network technology and storage technology, especially the development of cloud computing and big data, many complex problems have emerged, which not only involve a large amount of computing but also deal with a large scale of data, namely, the so-called massive data processing. The explosive growth of all kinds of data has pushed human society into the era of big data. Big data has a series of characteristics such as high dimension, strong dynamic, and randomness, which reflects the uncertainty of big data [1]. How to obtain valuable information from highly uncertain large-scale dynamic data is currently one of the most important research contents in the field of big data. In order to further improve the speed or performance of the computer, relying only on improving the performance of the computer can no longer meet the requirements. Another method is to introduce parallel processing technology at different levels from the aspect of computer architecture. Parallel data mining [2, 3], a technology combining parallel computing and data mining, has been widely used in all aspects of society. The application of parallel computing to data mining, especially data mining based on rough set theory, is a problem worth studying.

Rough set theory was proposed by Polish mathematician Pawlak [4–6] for the development of automatic rule generation systems and the study of soft computing problems. The theory mainly uses upper and lower approximation operators to describe inaccurate knowledge, which is an intelligent mathematical tool to deal with inconsistent and uncertain information. It can effectively analyze and process all kinds of information with characteristics such as inaccuracy, incompleteness, and inconsistency and reveal potential laws from the information. This theory not only can model nonlinear and discontinuous relations but also has high objectivity, which can effectively deal with big data. At the same time, it also has achieved great success in the fields of decision analysis, pattern recognition, and data mining [7, 8]. The remarkable feature of rough set is to derive the decision or classification rule of the problem through attribute reduction and value reduction while keeping the classification ability unchanged. Rough set attribute reduction theory, combined with other theories, can effectively deal with high-dimensional dynamic massive data. At present, in the field of big data, the research of rough set theory mainly focuses on model extension [9, 10] and attributes reduction algorithm design [11–14].

Attribute reduction, also known as dimension reduction or feature selection, comes from machine learning and is one of the core research contents of rough sets. Its purpose is to delete the attributes that are not relevant to classification in the data set and improve the performance of knowledge discovery of data. At present, attribute reduction has been widely used in the fields of pattern recognition and data mining. People have done a lot of research on attribute reduction in decision tables. Since the attributes in the decision table are not equally important, there are a number of redundant attributes in the attribute set. These redundant attributes have no effect on the decision results but reduce the efficiency of decision making. Therefore, it is necessary to remove these redundant attributes. Therefore, the main idea of attribute reduction is to eliminate those redundant attributes while keeping the classification ability of existing knowledge unchanged, which can reduce the size of the data set and improve the efficiency of knowledge discovery. Therefore, reducing the dimension of data through attribute reduction algorithm can reduce the training time and improve the quality of decision-making. Rough set attribute reduction algorithm can not only deal with small-scale data but also deal with large data effectively.

Parallel computing, in which multiple computing resources are used to solve computing problems at the same time, is an effective means to improve the computing speed and processing capacity of a computer system, which is also suitable for solving large-scale problems with large-scale and complex processes [15]. The basic idea of parallel computing [16] is to use the divide-and-conquer strategy to decompose the whole problem into several independent parts, and each part is processed in parallel by an independent processor. A parallel computing system can be either a specially designed supercomputer with multiple processors or a cluster of several independent computers that are interconnected in some way. In order to reduce the dimension of massive data, many researchers combine parallel computing with rough set theory, divide the tasks that need to be repeatedly calculated in the process of attribute reduction into multiple subtasks, and then deploy each subtask to different processors for processing at the same time. Finally, the final reduction result can be obtained by combining the intermediate reduction results with synthesis technology, and many achievements have been made in theoretical research and platform implementation [17–21].

Based on the idea of divide-and-conquer [22], this paper divides the whole problem into several problems and then applies the divide-and-conquer strategy on them. The rough set knowledge reduction method based on divide-and-conquer [23] can effectively reduce the complexity of problem processing and greatly improve the efficiency of knowledge reduction. However, the core attribute of divide-and-conquer method is based on the idea of divide-and-conquer and recursion; in fact, it is difficult to convert recursion to non-recursion in the program designing, and there is no concurrency in the process of divide-and-conquer, so it is difficult to achieve attribute reduction in parallel. Aiming at the shortcomings of divide-and-conquer method which cannot be implemented by parallel computing, we improve the method of divide-and-conquer and design a classification method that supports parallel attribute reduction. Specifically, it makes the relative positive domain which needs to be calculated repeatedly independent, and the independent relative positive domain calculation can be performed in parallel. Finally, the proposed algorithm is implemented and tested, and experimental results show that the proposed algorithm is not as efficient as the divide-and-conquer method in the non-parallel environment, but it is obviously better than the divide-and-conquer method in the parallel environment.

2. Related Concepts and Definitions

For the convenience of writing and description, some basic concepts and definitions of rough set theory are listed below.

Definition 1. Decision table. Let $S, S = <U, A = C \cup D, V, f>$ be a decision table; here U is the set of objects, also known as a universe, $A = C \cup D$ is an attribute set, C and D are called condition attribute set and decision attribute set, respectively, D cannot be empty; that is,$D \neq \varnothing$, $V$is a collection of property values, and $f: U \times A \longrightarrow V$is an information function, which specifies the attribute value of each object x in U.

Definition 2. Unclear relation. Let $S(S = <U, A = C \cup D, V, f>)$ be a decision table, for each subset of attributes,$B(B \subseteq A)$ defining an unclear relation for B as$\mathrm{IND}(B)$: $\mathrm{IND}(B) = \{(x, y) | (x, y) \in U \times U, \forall b \in B(b(x) = b(y))\}$.

Definition 3. Upper and lower approximation sets. For a decision table, for each subset of attributes $X(X \subseteq U)$ and unclear relation$\mathrm{IND}(B)$, the upper and lower approximation sets of X are defined by the basic set of Bas follows:

$$B^{-}(X) = \bigcup_{Y_i \in (U/\mathrm{IND}(B)) \wedge Y_i \cap X \neq \varnothing} Y_i,$$
$$B_{-}(X) = \bigcup_{Y_i \in (U/\mathrm{IND}(B)) \wedge Y_i \subseteq X} Y_i. \quad (1)$$

Definition 4. Positive region. Let U be a domain, P and Q are equivalence relational clusters defined on U, and the positive region P of Q is defined as$Pos_P(Q)$: $Pos_P(Q) = \cup_{X \in (U/Q)} P_{-}(X)$.

Definition 5. Relative reduction. Let Ube a domain, and P are Q equivalence relational clusters defined on U, let $S(S \subset P)$ be an independent subset of P relative to Q, if there exists$Pos_S(Q) = Pos_P(Q)$; thus, S is relative reduction of P relative to Q.

Definition 6. Core attribute. Let U be a domain, and$P$ are Q equivalence relational clusters defined on U; if $Pos_P(Q) = Pos_{P-\{r\}}(Q)$, r is unnecessarily relative to Q in P; all the necessary attributes in P relative to Q form a set; the set is called Q core of P, named as $\mathrm{CORE}_Q(P)$.

Definition 7. Necessary attributes. Let U be a domain, and P and Q are equivalence relational clusters defined on U; if all properties of P are necessarily relative to Q, P is relatively independent relative to Q.

Next, the related concepts of parallel computing are given below.

Definition 8. Parallel computer is a computer with multiple processors capable of parallel processing.

Definition 9. Parallel processing is an efficient form of information processing that emphasizes concurrent operations on data elements that belong to one or more processes that solve a single problem.

Definition 10. Parallel algorithm is a collection of simultaneous processes that interact and coordinate to achieve a solution to a given problem.

Definition 11. Data parallelism means that the data is divided into several blocks and mapped to different processors, respectively. Each processor runs the same processing program to process the assigned data. If the overhead associated with parallelism is not added, the processing speed of the feature increases by k times, and the throughput of the system increases by k times.

Definition 12. Acceleration coefficient. $S(n) = t_s/t_n$. t_s is the execution time of the system with a single processor, t_n is the time required for execution using a system with n processors, and acceleration coefficient is a standard to measure the performance of a system with multiple processors.

3. Application of Divide-and-Conquer Method in Attribute Reduction

The design idea of divide-and-conquer is to divide a big problem that is difficult to solve directly into some smaller identical subproblems, divide and conquer. The subproblems generated by the divide-and-conquer method are often smaller models of the original problem, and then the subproblem is reduced to a point where it is easy to find its solution directly. Literature [22] proposed an attribute core solution method based on the divide-and-conquer method, and experiments proved the efficiency of the algorithm and its suitability for processing large data sets.

The divide-and-conquer method to find the core attribute is based on the algorithm of finding the positive domain. In the division of the domain, the decision table S is divided into $k(k = \mathrm{IND}(U/\{c\}))$, sub-decision-tables as $S_1, S_2, \ldots, S_k$, and it can be seen from Definition 5 that the core attribute of the child decision table must be a subset of the core attributes of the parent decision table, but it is not necessarily a true subset; that is, it is impossible to judge whether attribute c belongs to the core attribute of the parent decision table, according to the definition of the previous core attribute; if the relative positive field of removing the attribute c is different from that of not removing the attribute c, then the attribute c is the core attribute. Therefore, in the recursive algorithm of finding the positive domain, we should first judge whether the attribute c is the core attribute and then recursively find the positive domain of the subdomain and the core attribute.

4. Attribute Reduction Based on Classification

The divide-and-conquer method of seeking core attributes is based on the idea of divide-and-conquer and recursion. While it is difficult to convert recursion to no recursion in program designing, there is no concurrency in the process of divide-and-conquer, and it is difficult to achieve parallel attribute reduction. In view of the disadvantages of the divide-and-conquer method which cannot be paralleled, we improved the method and designed a classification method that supports parallelism.

Finding the positive domain of decision table is actually a process of classification; all the instances in the decision table are classified according to given attribute set, and then the classification results are processed as follows: all the cases in each category are compared by their decision attributes, and if all are the same, then the instances in the category are added to positive domain. How to classify quickly determines the performance of the algorithm and how to realize fast classification determines the speed of algorithm performance.

4.1. Finding Positive Domain Based on Classification. Let $S(S = <U, C \cup D, V, f>)$ be a decision table, $n = |U|$ and $m = |C|$, and if the decision table classification uses the direct sorting method, the time complexity of the sorting algorithm is at least $O(m * n * \lg(n))$. In the classification algorithm in this article, for each attribute in the domain, we use the index method to traverse the attribute value of all instances corresponding to the attribute, so that the attribute values corresponding to all instances can be processed in $O(n)$ time, and then classify each classification result according to the remaining attributes until the last condition attribute. After the classification is completed, we also need to process the decision attributes of all instances of each classification. From the properties of positive domain, we can know that, for the instances in the decision table with exactly the same condition attributes, if their decision attributes are also the same, then they should be incorporated into the relative positive domain. Next, we give the positive region acquisition algorithm based on classification (Algorithm 1).

In the process of finding the positive domain by the classification method, considering the space and time complexity, a preallocated structure array is used, its size is $2 * |U|$, one is used to store the classification set to be divided, and the other is used to store the division results, and both of them are used alternately. At the same time, the array is chained so that the time and space complexity are greatly reduced.

4.2. Finding Core Attribute by Classification. According to the definition of core attribute, let $S(S = <U, R = C \cup D, V, f>)$, and if $c(c \in C)$ is a core attribute, then the

(i) Input: Decision table $S = <U, A = C \cup D, V, f>$
(ii) Output:$Pos_C(D)$Positive region of the decision table.
(iii) Step 1: let$Pos_C(D) = \varnothing$, domain$O$ Set $= U$, $i = 0$.
(iv) Step 2: Initialize the attribute index arrayAttand instance index arrayEntryof the domain in the decision table, initialize a new domain asDomain, add all instances in the new domain to the initial domain.
(v) Step 2.1 for each att_i inAtt, classify each attribute in an order.
Step 2.2 for each domain inDomain, classify the domain according to the attributeatt_i.
Step 2.3 for eachentryinDomain, insert it into the corresponding domain according to its corresponding attribute value.
(vi) Step 2.4 Domain = Domain $- >$ nextDomainif(Domain! = null) goto Step 2.1
(vii) Step 2.5$i + +$; $if(i<|\text{Att}|)$ goto Step 2.2
(viii) Step 3: for eachdomain$[i]$, traverse all the lastDomain.
(ix) Step 4: if$(!(\forall x, y \in \text{domain}[i] \wedge x \neq y \&\& \text{DecValue}(x) = \text{DecValue}(y)))$goto Step 6
(x) Step 5:positive+ = domain$[i]$if (Domain! = null) goto Step 3
(xi) Step 6: returnpositive

Algorithm 1: Positive region acquisition algorithm based on classification.

inequality $\text{POS}_{C\setminus\{c\}}(D) < \text{POS}_C(D)$is true; that is to say, the positive domain value of the conditional attribute set C removed attribute c relative to the decision attribute D is less than the relative positive domain value when it is not removed, and we call that the conditional attribute c of C could not be omitted from D, so it is a core attribute.

Therefore, in the process of finding the core attributes. We first need to find the positive domain of all attributes; the purpose of this is to prevent inconsistencies in the decision table and then find out the positive domain which is named asPositive$_c$after removing attribute C in turn; finally we could verify whether the attribute is a core attribute by comparing their sizes. Detailed algorithm is shown as follows (Algorithm 2).

4.3. Algorithm Complexity Analysis. In the process of finding positive domain based on classification, for each attribute, we use the index method to traverse all categories $k(k = |\text{IND}(U/\{c\})|)$, and the time complexity is k. Each category index needs to be initialized first, the initialization time is complex k, for each category, traverse the instances in the category, and since the number of instances is n, the time complexity is n; thus the time complexity of the algorithm can be easily expressed as $T(n, m) = (k + k^2 + n) * m$ and also as $T(n, m) = n * m + C * m$; therefore, the time complexity of the algorithm based on classification is $O(n * m) + C$, and the spatial complexity is $O(3 * n) = O(n)$. In the classification method for finding core attribute, each attribute requires a positive domain, so the time complexity is $O(n^2 * m)$, and the spatial complexity is $O(n)$.

5. Parallel Attribute Reduction Algorithm Implementation Based on MPI

5.1. Concurrency in Attribute Reduction. The core of concurrent computing is to seek concurrency. We improve the traditional serial decision table attribute reduction algorithm and divide the attribute reduction process into three parts: the core attribute of decision table calculation stage, attribute expansion stage, and attribute compression stage. These three parts all need to repeatedly calculate the relative positive domain of the attribute set, and the process of calculating the relative positive domain is relatively independent, so they all have good concurrency and can be implemented by parallel algorithms.

Although the divide-and-conquer method divides the original problem into small independent problems for solving, the recursive idea and dynamic partition cannot be well realized by parallel algorithm, while the classification algorithm calculates the relative positive domain of each attribute, and they are independent of each other, so it can be realized by parallel computing. In the process of attribute expansion and attribute compression of decision table, the relative positive domain of attribute set should be calculated repeatedly regardless of divide-and-conquer and classification, so this part can be realized by parallel computation.

5.2. Parallel Attribute Reduction Algorithm. According to the concurrency in the attribute reduction process, the following parallel attribute reduction algorithm can be obtained and the specific implementation is shown in Algorithm 3. In order to balance the number of attributes allocated between processes, each process is utilized to the maximum extent according to the drawer principle, and we divide the attributes to be allocated equally, so that the difference of the number of attributes allocated by each process is not more than one; that is, the attributes to be assigned in the core attribute, attribute expansion, and attribute compression stages must be assigned according to the drawer principle, which guarantees maximized concurrency.

Let S be a decision table, C is a conditional attribute set, D is decision attribute set, rank is process number, and size is the number of processes.

5.3. Algorithm Complexity Analysis. The time complexity of parallel attribute reduction algorithm based on MPI is related to the number of parallel processes. In this paper, we assume that the number of parallel processes is p; according to the algorithm's parallel processing process, each process is first

(i) Step 1: find the positive domain of all attributes$\mathrm{POS}_C(D)$
(ii) Step 2: $\forall c | c \in C$, find positive domain$\mathrm{Positive}_{C\setminus\{c\}}(D)$after removing the attribute c
(iii) Step 3: $f(\mathrm{POS}_C(D) \neq \mathrm{POS}_{C\setminus\{c\}}(D))$ do CORE = CORE $\cup\, c$

Algorithm 2: Find core attribute by classification.

(i) Input: decision table$S(S = <U, A = C \cup D, V, f>)$.
(ii) Output: A relative attribute reduction R of decision table$S$.
(iii) Stage 1:Find core attribute in parallel
(iv) Step 1: Each process assigns attributes as $\mathrm{Att}_{\mathrm{rank}}$ according to ranklabel and drawer principle, and meets$|\mathrm{Att}_i| - |\mathrm{Att}_j| \leq 1 \wedge |\mathrm{Att}_i| \cap |\mathrm{Att}_j| = \varnothing \wedge |\mathrm{Att}_1| \cup \cdots \cup |\mathrm{Att}_j| = C$.
(v) Step 2: Each process computes the core attribute $\mathrm{CORE}_{\mathrm{rank}}$ in$\mathrm{Att}_{\mathrm{rank}}$.
(vi) Step 3: Each process exchanges the core attribute with each other, and obtains the final core attribute.$\mathrm{CORE} = \mathrm{CORE}_1 \cup \cdots \cup \mathrm{CORE}_{\mathrm{size}}$.
(vii) Stage 2: Prejudge whether the core attribute is the result of reduction
(viii) $If(\mathrm{POS}_{\mathrm{CORE}}(D)) = \mathrm{POS}_C(D)$ returnCORE
(ix) else goto Stage 3. Step 1
(x) Stage 3: Attribute expansion stage
(xi) Step 1: Calculate the attributes to be addedAtt_left = $C\setminus$CORE,Att_need = CORE.
(xii) Step 2: Each process assigns attributes as according to rank label and drawer principle, and meets the following condition $|\mathrm{Att}_i| - |\mathrm{Att}_j| \leq 1 \wedge \mathrm{Att}_i \cap \mathrm{Att}_j = \varnothing \wedge \mathrm{Att}_1 \cup \cdots \cup \mathrm{Att}_j = \mathrm{Att\_lef}$.
(xiii) Step 3: Each process computes the best attributes to be added $C_{\mathrm{select}} | \forall i \in \mathrm{Att}_{\mathrm{rank}} \wedge i \neq \mathrm{select}$, $|\mathrm{POS}_{C_{\mathrm{select}} \cup \mathrm{Att\_need}}(D)| > |\mathrm{POS}_{C_i \cup \mathrm{Att\_need}}(D)|$.
(xiv) Step 4: Each process sends the results of the calculations in Step3 to the main process.$\mathrm{SendMsg}(C_{\mathrm{select}}, \mathrm{POS}_{C_{\mathrm{select}} \cup \mathrm{Att\_need}}(D))$.
(xv) Step 5: The main process receives calculation results of each process and calculates the attribute that is best to be added.$C_{\mathrm{select}} | \forall i \leq \mathrm{size}, |\mathrm{POS}_{\mathrm{Att\_need} \cup C_{\mathrm{select}}}(D)| > |\mathrm{POS}_{\mathrm{Att\_need} \cup C_i}(D)|$.
(xvi) Then main process distributes the results,$\mathrm{SendMsg}(C_{\mathrm{select}}, \mathrm{POS}_{C_{\mathrm{select}} \cup \mathrm{Att\_need}}(D))$.
(xvii) Step 6: Each process accepts the calculation results of the main process and updates the reduction results. Att_left = Att_left$\setminus C_{\mathrm{select}}$,Att_need = Att_need $\cup\, C_{\mathrm{select}}$;
(xviii) Step 7: If $\mathrm{POS}_{\mathrm{Att\_need}}(D) = \mathrm{POS}_C(D)$ goto Stage4
(xix) else goto Stage 3. Step 2;
(xx) Stage 4:Attribute compression stage
(xxi)Step 1: Calculate the properties needed to be checked, Att_add = Att_need/CORE.
(xxii)Step 2: Each process checks whether C_{rank} can be removed according to itsrank. if $(\mathrm{POS}_{\mathrm{Att\_need}C_{\mathrm{rank}}}(D) = \mathrm{POS}_{\mathrm{Att\_need}}(D))$, send C_{rank} to main process
(xxiii)else Send -1 to main process.
(xxiv)Step 3: The main process receives the calculation result of the sub process, select one attribute to compress and distribute the results.
(xxv)Step 4: Each process updates the compression results Att_add = Att_add$\setminus C_{\mathrm{select}}$; Att_need = Att_need$\setminus C_{\mathrm{select}}$;
(xxvi)Step 5: if $(|\mathrm{Att\_add}| = \varnothing)$ return Att_need Att_need
(xxvii)else goto Stage 4.Step 2.

Algorithm 3: Parallel attribute reduction algorithm.

assigned attributes, and then each process processes the results according to the assigned attributes, and, finally, the main process processes the results that each process sends to the main process. In the attribute allocation stage, if the number of attributes is more than the number of processing processes, each process could be assigned attributes and conducted parallel processing. If the number of attributes is less than the number of processing processes, then some processes will not be assigned attributes. In extreme cases, that is, in the later stage of algorithm processing, as the number of attributes to be processed decreases, there will certainly be some processes that cannot be assigned attributes. At this time, the parallel algorithm degenerates into a nonparallel algorithm. In this way, the time complexity of the parallel attribute reduction algorithm is shown as formula (2), and, in the formula, the algorithm time constant C is the time required for parallel program message transmission, which is related to the network environment and cluster settings.

$$T(m,n) = \begin{cases} \dfrac{O(m^3 * n)}{p + C(p < m)}, \\ O(m^3 * n)(p > m). \end{cases} \tag{2}$$

6. Experiment and Result

In order to verify the effectiveness of the parallel attribute reduction algorithm proposed in this paper, we conducted

Table 1: Attribute reduction results of the two algorithms in nonparallel testing.

Number of data sets	Running time of divide-and-conquer (s)	Running time of classification (s)	Reduction result
97968	35.424784	40.115938	9/41
195936	64.141747	74.01378	15/41
293904	72.875936	90.239001	19/41
391872	89.184722	113.409269	19/41
489840	105.541708	136.741713	19/41
587808	121.805281	159.759949	19/41
685776	139.719959	183.720615	19/41
783744	159.584528	212.715617	20/41
881712	176.411368	263.181288	20/41
979680	260.774013	329.194148	21/41

multiple sets of comparative experiments for testing. The data set is KDDCUP99 intrusion detection data [24], which has a total of 4,898,432 records, and each record contains 41 attributes. We randomly select 20% of the data sets, about 1 million pieces of data for testing.

The test is divided into two stages. The first stage tests the nonparallel performance of the original divide-and-conquer algorithm and the classification-based attribute reduction algorithm in this paper, as well as the difference between them. We first randomly select 10%, 20%,. . ., 100% from the 1 million records to generate the new data sets. Then we test the performance of the two algorithms in positive domain, attribute core, and attribute reduction on different data sets and draw conclusions through comparative analysis. The second stage of the experiment is to test the performance of the two algorithms in the parallel environment, and the data set obtained in the first stage is still used for testing, and then the performance of the two algorithms in the positive domain, attribute core, and attribute reduction on different data sets is tested, respectively, and the conclusion is drawn through comparative analysis.

The experiment is carried out on five computers by using MPICH.NT.1.2.5 and VS 2008 development tools in Windows environment. The system is configured with Windows XP, CPU P4, main frequency of 2.93 GHz, and 1G memory.

6.1. Nonparallel Testing. Nonparallel testing is mainly to test the performance of two algorithms and analyze the reasons for the difference in performance between them. According to the previous description, the 1 million pieces of data are selected at a rate of 10%. In order to reduce unnecessary errors, the two algorithms were, respectively, run 5 times on different data sets, and then the average running time was taken as the running time of this data set. Table 1 shows the attribute reduction results and running time of nonparallel programs and Figure 1 is the time comparison of the two algorithms.

It can be observed from Table 1 and Figure 1 that when there are 100,000 pieces of data, the difference in completion time between the two algorithms is very small. With the continuous increase of the data sets, the performance of the divide-and-conquer algorithm is gradually better than that of the classification algorithm. Especially when there are 900,000 data sets, the completion time of the classification algorithm increases by nearly 49.5% compared with that of the divide-and-conquer algorithm. Through experimental analysis, the algorithm performance of divide-and-conquer method is higher than that of taxonomy in nonparallel operation. Therefore, divide-and-conquer method has more advantages than taxonomy in nonparallel environment.

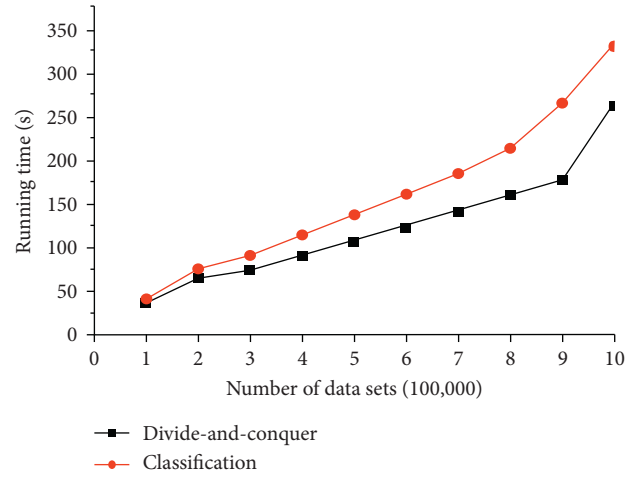

Figure 1: Running time of the two algorithms in nonparallel testing on different set.

6.2. Parallel Testing. In parallel testing, in order to maintain consistency with nonparallel testing, the data of the nonparallel test is still used, and the MPI communication protocol is used to run on the PC with the same performance, and the average running time is calculated by 5 times in the same test. Table 2 shows the attribute reduction results and running time in the parallel running environment, and Figure 2 shows the time analysis diagram of the two algorithms. Finally, in order to further compare the performances of the two algorithms, the comparison diagram of the two algorithms in the parallel and nonparallel environment is given as shown in Figure 3.

It can be clearly observed from Table 2 and Figure 2 that, in a parallel operating environment, the performance of the classification algorithm is better than that of the divide-and-conquer method, and its running time is reduced by nearly 30% on average compared to the divide-and-conquer method. Through comparison, it can be found that the overall running time of the two algorithms in a parallel environment is less than that in a nonparallel environment.

Table 2: Attribute reduction results of the two algorithms in parallel testing.

Number of data sets	Running time of divide-and-conquer (s)	Running time of classification (s)	Reduction result
97968	35.166193	26.079651	9/41
195936	65.103365	45.734677	15/41
293904	80.939468	62.347393	19/41
391872	99.040613	77.326405	19/41
489840	117.440574	87.380716	19/41
587808	135.404997	101.736253	19/41
685776	153.235049	117.03368	19/41
783744	173.126772	133.629863	20/41
881712	198.306259	148.357779	20/41
979680	256.82359	207.596716	21/41

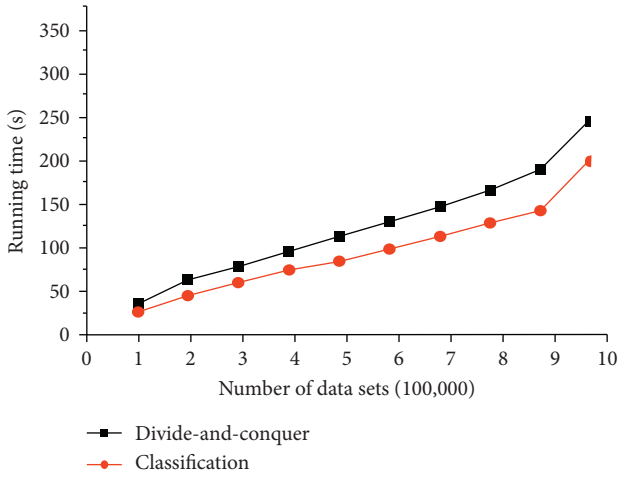

Figure 2: Running time of the two algorithms in parallel testing on different set.

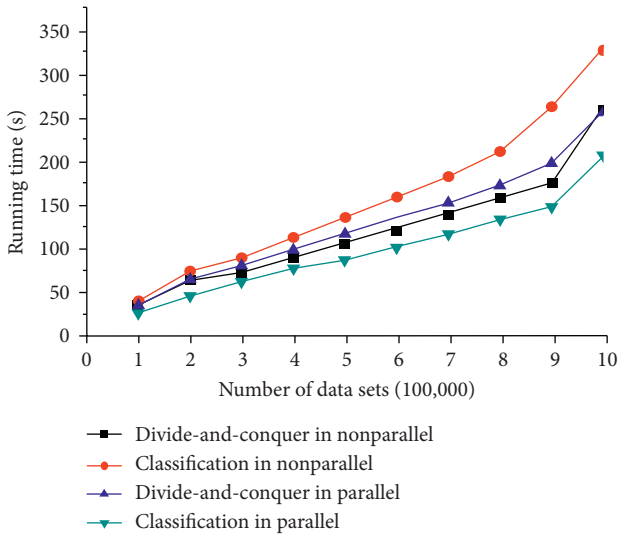

Figure 3: Comparison of the two algorithms in nonparallel and parallel testing.

However, it can also be found in Figure 2 that the reduction performance based on the divide-and-conquer algorithm has a downward trend; as discussed earlier, the divide-and-conquer method cannot be used in parallel when finding core attribute, and each process calculates the core attribute independently, which is no different with the nonparallel method. If the core attribute is the reduction result or close to the reduction result, this parallel algorithm actually reduces operating efficiency. However, it can be clearly observed from Figure 2 that the classification algorithm is superior to the divide-and-conquer algorithm in terms of attribute reduction in the parallel environment, and the performance of the classification parallel algorithm is obviously superior to those of the other three algorithms. This result proves that the classification parallel attribute reduction algorithm proposed in this paper has more advantages in terms of time efficiency.

7. Conclusion

In this paper, starting from improving the efficiency of the rough set knowledge acquisition algorithm, combining the ideas of divide-and-conquer, classification, and parallelism, a parallel attribute reduction algorithm based on rough set is proposed. By improving the divide-and-conquer method without parallelism, a classification method that supports parallel processing in a parallel environment can greatly improve the processing efficiency of attribute reduction. Experimental results show that the algorithm proposed in this paper is more suitable for attribute reduction processing of massive data sets.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the Science and Technology Key Project of Henan Province under Grant no. 202102210370.

References

[1] G. Wang, J. Yang, and J. Xu, "Granular computing: from granularity optimization to multi-granularity joint problem solving," *Granular Computing*, vol. 2, no. 3, pp. 105–120, 2017.

[2] L. Jian, C. Wang, Y. Liu, S. Liang, W. Yi, and Y. Shi, "Parallel data mining techniques on graphics processing unit with compute unified device architecture (CUDA)," *The Journal of Supercomputing*, vol. 64, no. 3, pp. 942–967, 2013.

[3] G. Laccetti, M. Lapegna, V. Mele, D. Romano, and L. Szustak, "Performance enhancement of a dynamic K-means algorithm through a parallel adaptive strategy on multicore CPUs," *Journal of Parallel and Distributed Computing*, vol. 145, pp. 34–41, 2020.

[4] Z. A. Pawlak, "Rough sets," *International Journal of Computer & Information Sciences*, vol. 11, no. 5, pp. 341–356, 1982.

[5] Z. Pawlak, "Rough set theory and its applications to data analysis," *Cybernetics and Systems*, vol. 29, no. 7, pp. 661–688, 1998.

[6] Z. Pawlak, "Rough set approach to multi-attritute decision analysis," *European Journal of Operational Research*, vol. 72, pp. 443–459, 1994.

[7] Z. Pawlak and A. Skowron, "Rudiments of rough sets," *Information Sciences*, vol. 177, no. 1, pp. 3–27, 2007.

[8] Z. Pawlak and A. Skowron, "Rough sets and Boolean reasoning," *Information Sciences*, vol. 177, no. 1, pp. 41–73, 2007.

[9] C. Liu, P. Edrycz, and J. Qian, "Covering- based multi-granulation decision-theoretic rough set approaches with new strategies," *Journal of Intelligent and Fuzzy Systems*, vol. 1, pp. 1–13, 2018.

[10] Q. Hu, L. Zhang, Y. Zhou, and W. Pedrycz, "Large-scale multimodality attribute reduction with multi-kernel fuzzy rough sets," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 1, pp. 226–238, 2018.

[11] J. Dai Hu, Q. Hu, D. HuangHu et al., "Neighbor inconsistent pair selection for attribute reduction by rough set approach," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 2, pp. 937–950, 2018.

[12] A. K. Das Sengupta and S. S. Bhattacharyya, "A group incremental feature selection for classification using rough set theory based genetic algorithm," *Applied Soft Computing*, vol. 65, pp. 400–411, 2018.

[13] C. Wang, Y. Huang, W. Ding, and Z. Cao, "Attribute reduction with fuzzy rough self-information measures," *Information Sciences*, vol. 549, pp. 68–86, 2021.

[14] C. Wang, Y. Huang, M. Shao, and X. Fan, "Fuzzy rough set-based attribute reduction using distance measures," *Knowledge-Based Systems*, vol. 164, pp. 205–212, 2019.

[15] K. Asanovic, R. Bodik, J. Demmel et al., "A view of the parallel computing landscape," *Communications of the ACM*, vol. 52, no. 10, pp. 56–67, 2009.

[16] C.-F. Tsai and S.-W. Lin, "Big data mining with parallel computing: a comparison of distributed and MapReduce methodologies:a comparison of distributed and MapReduce methodologies," *Journal of Systems and Software*, vol. 122, pp. 83–92, 2016.

[17] H. Chen, T. Li, Y. Cai, C. Luo, and H. Fujita, "Parallel attribute reduction in dominance-based neighborhood rough set," *Information Sciences*, vol. 373, pp. 351–368, 2016.

[18] J. Qian, D. Miao, Z. Zhang, and X. Yue, "Parallel attribute reduction algorithms using MapReduce," *Information Sciences*, vol. 279, pp. 671–690, 2014.

[19] J. Qian, P. Lv, X. Yue, C. Liu, and Z. Jing, "Hierarchical attribute reduction algorithms for big data using MapReduce," *Knowledge-Based Systems*, vol. 73, pp. 18–31, 2015.

[20] C. Wang, Y. Wang, M. Shao et al., "Fuzzy rough attribute reduction for categorical data," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 5, pp. 818–830, 2019.

[21] C. Wang, Y. Huang, M. Shao et al., "Feature selection based on neighborhood self-information," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 4031–4042, 2019.

[22] R. Hannane, A. Elboushaki, and K. Afdel, "A divide-and-conquer strategy for facial landmark detection using dual-task CNN architecture," *Pattern Recognition*, vol. 107, Article ID 107504, 2020.

[23] D. Lianjie, C. Degang, W. Ningling, and L. Zhanhui, "Key energy-consumption feature selection of thermal power systems based on robust attribute reduction with rough sets," *Information Sciences*, vol. 532, pp. 61–71, 2020.

[24] KDD http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.