mathematics

MDPI

Article

Efficient Breadth-First Reduct Search

Veera Boonjing [1] and Pisit Chanvarasuth [2,*]

[1] Department of Computer Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Ladkrabang, Bangkok 10520, Thailand; veera.bo@kmitl.ac.th

[2] School of Management Technology, Sirindhorn International Institute of Technology, Thammasat University, Pathum Thani, Bangkok 10200, Thailand

* Correspondence: pisit@siit.tu.ac.th; Tel.: +66-2-501-3505-20

Received: 14 April 2020; Accepted: 18 May 2020; Published: 21 May 2020

Abstract: This paper formulates the problem of determining all reducts of an information system as a graph search problem. The search space is represented in the form of a rooted graph. The proposed algorithm uses a breadth-first search strategy to search for all reducts starting from the graph root. It expands nodes in breadth-first order and uses a pruning rule to decrease the search space. It is mathematically shown that the proposed algorithm is both time and space efficient.

Keywords: reduct; subset; feature selection; breadth-first search; rooted graph

1. Introduction

In machine learning, feature selection is a process that selects relevant features used as the input in learning models. Its intention is to obtain optimal features so that the model can be used to accurately predict the output. Such optimal features are known as reducts in rough set theory. A reduct of an information system with conditional attributes and a decision attribute is defined as a minimal subset of a set of conditional attributes, in which its degree of dependency on the decision attribute is the same as the set of conditional attributes; (see [1] for a formal definition). A reduct could be any nonempty subset of the conditional attributes with the degree of dependency. Hence, the number of possible reducts is exponential with respect to the number of conditional attributes. With the aim of computational efficiency, a number of algorithms are proposed as a solution to finding a single reduct or multiple reducts without exhaustively investigating all of these possibilities. Therefore, many heuristic algorithms [2–6] and metaheuristic algorithms [7–14] have been proposed, as discussed in [15]. This class of algorithms is known as approximate algorithms. They give a single reduct or multiple reducts but not an exhausted list of reducts as the exact algorithms do. As the results of approximate algorithms requiring a parameter setting, they may produce different reducts in different runs. Moreover, they could create reducts that are not optimal. The exact algorithms are necessary if we are interested in computing the best reduct, with a given criterion, out of all reducts. However, finding all reducts involves generating and examining all possible reducts. In the literature, this can be done based on a discernibility matrix [16] and a power set tree [17,18]. Therefore, these exact algorithms have a time complexity exponential with respect to the number of conditional attributes.

In this work, we propose an exact algorithm to find all reducts without generating and examining all possible reducts. A simple representation of the possible reducts called a solution rooted graph is proposed. The rooted graph is formed by possible subsets of conditional attributes and their connections. Its root node is an n-subset attribute. The node is connected to its $(n-1)$-subset nodes. Each $(n-1)$-subset node is connected to its $(n-2)$-subset nodes. This continues until reaching a 0-subset node. Hence, each node of the rooted graph except for a 0-subset node is a possible reduct. Furthermore, there are n node types based on their cardinalities in the search space: n-subset, $(n-1)$-subset, $(n-2)$-subset, ... , and 1-subset. The proposed algorithm searches on the solution

rooted graph with the breadth-first search. This work adopts the breadth-first search because it is complete, i.e., it assures finding all reducts. This algorithm still involves generating and examining all possible reducts from n-subset type to 1-subset type, type by type. We know that any node with a degree of dependency less than the graph root is not a reduct and its subsets are not reducts according to "the monotonic property of dependency". Then, all of these subsets can be eliminated from consideration without losing any optimal reducts. The proposed algorithm is equipped with this rule of elimination as the pruning rule, which is the basis of its efficiency.

This paper is organized as follows. Section 2 briefly gives a sufficient background on reducts in terms of the degree of dependency. Section 3 describes the new efficient breadth-first reduct search algorithm. Analysis of the algorithm is given in Section 4. We conclude the paper in Section 5. An illustrative example is shown in Appendix A.

2. Basic Concepts

This section gives the background on reducts in terms of the degree of dependency. More details on reducts and rough sets can be found in [1].

Definition 1. *Any 4-tuple IS = <U, A = C ∪ D, V, f>; C ∩ D = Φ is called an information system; where U is a finite set of objects; A is a finite set of attributes; C is a finite set of conditional attributes; D is a finite set of decision attributes; V = $\cup_{p \in A} Vp$, where V_p is a domain of the attribute p, and f: U × A → V is a function called an information function f(x_i, q) for every q ∈ A and x_i ∈ U. An information system is denoted by IS = (U, A).*

Example 1. *Let us consider the simple information system shown in Table 1. We adopt this table to illustrate the basic concepts in the following examples.*

Table 1. Example dataset.

	C_1	C_2	C_3	C_4	C_5	D
x_1	1	2	1	3	1	2
x_2	3	3	2	3	1	2
x_3	3	3	2	3	1	1
x_4	2	1	1	2	1	1
x_5	2	2	1	1	1	2
x_6	3	2	3	1	1	3
x_7	2	2	3	1	1	3
x_8	1	1	2	2	1	1
x_9	3	3	3	1	1	3
x_{10}	1	1	1	1	1	1

From Table 1, let P be C_1. We have

$$U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\},$$

$$A = \{C_1, C_2, C_3, C_4, C_5, D\},$$

$$VP = \{1, 2, 3\},$$

$$f(x_4, C_1) = \{2\},$$

$$and\ f(x_{10}, D) = \{1\}.$$

Definition 2. *Let R be any nonempty subset of A. An R-indiscernibility relation, denoted by IND(R), is defined as IND(R) = {(x_i, x_j): (x_i, x_j) ∈ U × U, a ∈ R, f (x_i, a) = f (x_j, a)}.*

Note that if $(x_i, x_j) \in$ IND(R), then objects x_i and x_j are called indiscernible with respect to R. The relation IND(R) is an equivalence relation. Therefore, it forms a partition U/IND(R); the equivalence classes of the R-indiscernibility relation.

Example 2. *Let R be {C_1,C_2,C_3,C_4,C_5}. We have*

$$IND(R) = \{(x_2, x_3)\},$$

$$IND(\{C_1\}) = \{(x_1, x_8), (x_1, x_{10}), (x_8, x_{10}), (x_4, x_5), (x_4, x_7), (x_5, x_7), (x_2, x_3), (x_2, x_6), (x_2, x_9), (x_3, x_6), (x_3, x_9), (x_6, x_9)\},$$

$$U/IND(R) = \{\{x_2, x_3\}, \{x_1\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\}, \{x_8\}, \{x_9\}, \{x_{10}\}\},$$

$$and\ U/IND(\{C_1\}) = \{\{x_1, x_8, x_{10}\}, \{x_4, x_5, x_7\}, \{x_2, x_3, x_6, x_9\}\}.$$

Definition 3. *Let R be any nonempty subset of A and X be any nonempty subset of U. The R-lower approximation of X, denoted by $\underline{R}X$, is defined as $\cup \{Y \in U/IND(R): Y \subseteq X\}$. The R-lower approximation of X contains all objects with the known values of R that belong to X. The P-lower approximation of X contains all objects that with the knowledge of attributes P can be classified as belonging to concept X.*

Example 3. *Let R be {C_1,C_2,C_3,C_4,C_5}. We have*

$$U/IND\ (D) = \{\{x_1, x_2, x_5\}, \{x_3, x_4, x_8, x_{10}\}, \{x_6, x_7, x_9\}\},$$

$$X_1 = \{x_1, x_2, x_5\}, X_2 = \{x_3, x_4, x_8, x_{10}\}, X_3 = \{x_6, x_7, x_9\},$$

$$U/IND(R) = \{\{x_2, x_3\}, \{x_1\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\}, \{x_8\}, \{x_9\}, \{x_{10}\}\},$$

$$\underline{R}X_1 = \{x_1\} \cup \{x_5\} = \{x_1, x_5\},$$

$$\underline{R}X_2 = \{x_4\} \cup \{x_8\} \cup \{x_{10}\} = \{x_4, x_8, x_{10}\},$$

$$and\ \underline{R}X_3 = \{x_6\} \cup \{x_7\} \cup \{x_9\} = \{x_6, x_7, x_9\}.$$

Definition 4. *Let R be any nonempty subset of A and X be any nonempty subset of U. Then, the positive region of the partition U/IND (D) with respect to R, denoted by POS_R (D), is defined as $\cup_{X \in U/IND\ (D)} \underline{R}X$.*

Example 4. *From Example 3, we have*

$$POS_R(D) = \underline{R}X_1 \cup \underline{R}X_2 \cup \underline{R}X_3 = \{x_1, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}.$$

Theorem 1 ([19]). *Let IS = (U, A = C ∪ D). If B ⊆ C, then we have: POS_B (D) ⊆ POS_C (D).*

Proof. For each x ∈ POS_B(D), there is y ∈ U/IND(D) such that $[x]_{IND(B)} \subseteq$ y where $[x]_{IND(B)}$ is the equivalence class of x with regard to equivalence relation IND(B). Since B ⊆ C, we have $[x]_{IND(C)} \subseteq [x]_{IND(B)}$. Thus, $[x]_{IND(C)} \subseteq$ y, then x ∈ POS_C (D). Therefore, POS_B (D) ⊆ POS_C (D). □

Definition 5. *Let R be any nonempty subset of C. D depends on R in a degree k (0 ≤ k ≤ 1) where*

$$k = \gamma_R\ (D) = |POS_R\ (D)|/|U|.$$

Example 5. *From Example 4, we have $k = \gamma_R$ (D) = $|POS_R(D)|/|U| = |\{x_1, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}|/|U| = 8/10 = 0.8$. Then, D depends on R with degree 0.8.*

Example 6. *Let R be {C_1,C_2}. We have*

$$U/IND(D) = \{\{x_1, x_2, x_5\}, \{x_3, x_4, x_8, x_{10}\}, \{x_6, x_7, x_9\}\},$$

$$X_1 = \{x_1, x_2, x_5\}, X_2 = \{x_3, x_4, x_8, x_{10}\}, X_3 = \{x_6, x_7, x_9\},$$

$$U/IND(R) = \{\{x_2, x_3, x_9\}, \{x_1\}, \{x_4\}, \{x_5, x_7\}, \{x_6\}, \{x_8, x_{10}\}\},$$

$$\underline{R}X_1 = \{x_1\} = \{x_1\},$$

$$\underline{R}X_2 = \{x_4\} \cup \{x_8, x_{10}\} = \{x_4, x_8, x_{10}\},$$

$$\underline{R}X_3 = \{x_6\} = \{x_6\},$$

$$POS_R(D) = \underline{R}X_1 \cup \underline{R}X_2 \cup \underline{R}X_3 = \{x_1, x_4, x_6, x_8, x_{10}\},$$

$$and\ k = \gamma_R(D) = |\{x_1, x_4, x_6, x_8, x_{10}\}|/|U| = 5/10 = 0.5.$$

Definition 6. *Let C′ be any nonempty subset of C. C′ is a D-reduct (reduct with respect to D) of C, if C′ is a minimal subset of C such that $\gamma_C(D) = \gamma_{C'}(D)$.*

Example 7. *Let R be {C_1,C_2,C_3}. We have*

$$U/IND(D) = \{\{x_1, x_2, x_5\}, \{x_3, x_4, x_8, x_{10}\}, \{x_6, x_7, x_9\}\},$$

$$X_1 = \{x_1, x_2, x_5\}, X_2 = \{x_3, x_4, x_8, x_{10}\}, X_3 = \{x_6, x_7, x_9\},$$

$$U/IND(R) = \{\{x_2, x_3\}, \{x_1\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\}, \{x_8\}, \{x_9\}, \{x_{10}\}\},$$

$$\underline{R}X_1 = \{x_1\} \cup \{x_5\} = \{x_1, x_5\},$$

$$\underline{R}X_2 = \{x_4\} \cup \{x_8\} \cup \{x_{10}\} = \{x_4, x_8, x_{10}\},$$

$$and\ \underline{R}X_3 = \{x_6\} \cup \{x_7\} \cup \{x_9\} = \{x_6, x_7, x_9\},$$

$$POS_R(D) = \underline{R}X_1 \cup \underline{R}X_2 \cup \underline{R}X_3 = \{x_1, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\},$$

$$and\ k = \gamma_R(D) = |\{x_1, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}|/|U| = 8/10 = 0.8.$$

Since each nonempty subset R′ ∈ {{C_1},{ C_2},{ C_3},{ C_1,C_2},{ C_1,C_3},{ C2, C3}} of R gives $\gamma_{R'}$(D) < γ_R(D), then R = {C_1,C_2,C_3} is a D-reduct of C.

Therefore, we find a reduct by obtaining a minimal subset of conditional attribute (C) so that the decision attributes (D) depend on it in the same degree as depending on C. A brute force approach to the problem is to check all subsets of C, with a minimal condition, satisfying the dependency of D on C (γ_C(D)). Each minimal subset obtained (reduct) requires checking all of its $2^n - 2$ subsets—excluding itself and an empty set; where n is its cardinality.

3. Efficient Breadth-First Reduct Search Algorithm

Let C = {C_1,C_2,C_3, … , C_n} be a conditional attribute of an information system IS(C, D). There are n + 1 types of subsets of C: n-subset, (n − 1)-subset, (n − 2)-subset, … , 1-subset, and 0-subset. Reducts could be any types of these subsets except for the 0-subset. The breadth-first reduct search algorithm orderly investigates the following $2^n - 1$ subsets, type by type: n-subset, (n − 1)-subset, (n − 2)-subset, … , and 1-subset. Thereafter, these subsets are referred to as reduct candidates. For each subset C′ in reduct candidates, if $\gamma_{C'}$(D) = γ_C(D), then C′ is a new element of the reduct set, and all of its supersets in the reduct set, if they exist, are eliminated. If $\gamma_{C'}$(D) < γ_C(D), all subsets of C′ are not

reduct candidates. Each nonreduct C′ reduces $2^{|C'|} - 2$ elements of the reduct candidates, according to Theorem 1. A reduct set is obtained once all reduct candidates are investigated.

The algorithm implements the idea above by using three data structures: Candidate Queue (the first-come first-served structure), Reduct_Set, and NonReduct_Set to maintain the reduct candidates, reducts, and nonreducts, respectively. Initially, it calculates k = $\gamma_C(D)$, adds C to Candidate Queue, and sets both Reduct_Set and NonReduct_Set to an empty set. It loops to update the data of Candidate Queue, Reduct_Set, and NonReduct_Set if Candidate Queue is not empty. For each loop, it gets an element C′ from Candidate Queue and calculates $\gamma_{C'}(D)$. If $\gamma_{C'}(D)$ = k, then it adds C′ to Reduct_Set using the updateReduct_Set(C′) procedure. It also generates all (|C′| − 1)-subsets and insert them into Candidate Queue using the updateCandidate Queue(C′) procedure. If $\gamma_{C'}(D)$ < k, it inserts C′ and all its subsets into NonReduct_Set using the updateNonReduct_Set(C′) procedure. The algorithm (in detail) is as shown in Figure 1.

Input : Decision Table T(C, D)
C is a finite set of conditional attributes,
D is a finite set of decision attributes.
Output : Reduct_Set
Step 1:
Calculate k = $\gamma_C(D)$
Step 2:
Add C to Candidate_Queue, set Reduct_Set= {}, and set NonReduct_Set = {}
Step 3:
While (Candidate_Queue is not empty)
Loop
Get C′ from Candidate_Queue
If $\gamma_{C'}(D)$ = k then updateReduct_Set(C′) and updateCandidate_Queue(C′)
else updateNonReduct_Set(C′)}
End Loop
Step 4:
Return Reduct_Set.

Figure 1. Efficient breadth-first reduct search algorithm.

There are three major procedures in our algorithm: updateReduct_Set(C′), updateCandidate Queue(C′), and updateNonReduct_Set(C′).

updateReduct_Set(C′)

Each element in Reduct_Set is not a reduct if we can find any of its subsets that are also reducts. Therefore, we have to test whether each reduct in Reduct_Set is a superset of a new reduct. If it is a superset, we eliminate it from Reduct_Set before putting the new reduct into it, to gain the reduct minimal condition. For example, let Reduct_Set be {{C_1,C_2,C_3,C_4,C_5}}, let C′ be {C_1,C_2,C_3}, and $\gamma_{C'}(D)$ = $\gamma_C(D)$; therefore, a new reduct is C′. However, there is a reduct {C_1,C_2,C_3,C_4,C_5} in Reduct_Set that is a superset of C′. We, therefore, remove {C_1,C_2,C_3,C_4,C_5} from the Reduct_Set and insert C′ into it. This gives the new Reduct_Set = {{C_1,C_2,C_3}}. The procedure (in detail) is as shown in Figure 2.

Procedure updateReduct_Set(C′)
Begin
Remove all supersets of C′ and insert C′ to Reduct_Set
End

Figure 2. updateReduct_Set(C′) procedure.

updateCandidate_Queue(C′)

The procedure generates all (|C′| − 1)-subsets from C′ and tests whether each is a reduct candidate. A reduct candidate is not a subset of any NonReduct_Set element. Such a candidate is appended into Candidate Queue. For example, let C′ be {C_1,C_2,C_3,C_4}, then a set of 3-subsets of C′ is {{C_1,C_2,C_3}, {C_1,C_2,C_4}, {C_2,C_3,C_4}, {C_1,C_3,C_4}}. If NonReduct_Set contains an element {C_2,C_3,C_4,C_5}, then {C_2,C_3,C_4} is not a reduct candidate since it is a subset of {C_2,C_3,C_4,C_5}. Therefore, {C_1,C_2,C_3}, {C_1,C_2,C_4}, and {C_1,C_3,C_4} are appended into Candidate Queue. The details of the procedure are shown in Figure 3.

Procedure updateCandidate_Queue(C′)
Begin
Generate all candidate (|C′|-1)-subsets from C′
For each candidate, if it is not a subset of any set in
NonReduct_Set then add it to Candidate_Queue
End

Figure 3. updateCandidate Queue(C′) procedure.

updateNonReduct_Set(C′)

The property of the positive region as shown in Theorem 1 allows us to reduce the search space, i.e., the number of reduct candidates. We know that if C′ ⊆ C, then we have $POS_{C'}(D) \subseteq POS_C(D)$. This infers that $\gamma_{C'}(D) \leq \gamma_C(D)$. In addition, if $\gamma_C(D) = k$ (the degree of dependency of D on C in the original data), then C′ and its subsets are not reducts. All of the subsets can be eliminated from the candidates. For example, let C be a conditional attribute {C_1,C_2,C_3,C_4,C_5} with $\gamma_C(D) = 0.8$ and let C′ be {C_1,C_2,C_4} with $\gamma_{C'}(D) = 0.6$. Then, C′ and its subsets cannot be reducts according to Theorem 1. We, therefore, do not need to explore these subsets. We then remove all subsets of C′ from Candidate Queue. The proposed algorithm stores these candidates in NonReduct_Set using the procedure updateNonReduct_Set(C′). The procedure (in detail) is as shown in Figure 4.

Procedure updateNonReduct_Set(C′)
Begin
Insert it into NonReduct_Set
Remove all subsets of C′ in Candidate_Queue
End

Figure 4. updateNonReduct_Set(C′) procedure.

4. Analysis of Algorithm

Let C = {C_1,C_2,C_3, … , C_n} be a conditional attribute of an IS(C, D). Additionally, let L_k be a set of k-subsets. We know that $|L_k| = \binom{n}{k}$ and $\sum_{k=0}^{n}\binom{n}{k} = 2^n$. The algorithm searches for reducts from each L_k level by level starting with k = n, k = n − 1, and so on, until k = 1. Therefore, the size of its search space is $\sum_{k=1}^{n}\binom{n}{k} = 2^n - 1$. For the best case, C is the only element of Reduct_Set and each element of L_{n-1} does not satisfy a reduct property. It tests C and all elements of L_{n-1}. The number of tests is $1 + \binom{n}{n-1} = 1 + n$. For the best-case scenario, the time complexity is O(n).

For the worst-case scenario, the algorithm gives a 1-subset reduct as an element of Reduct_Set. If all generated subsets satisfy a reduct property, the number of test is $\sum_{k=1}^{n}\binom{n}{k} = \binom{n}{1} + \binom{n}{2} + \ldots + \binom{n}{n-2} + \binom{n}{n-1} + \binom{n}{n}$. Therefore, the worst-case time complexity is $O(\binom{n}{1} + \binom{n}{2} + \ldots +$

$\binom{n}{n-2} + \binom{n}{n-1} + \binom{n}{n}) = O(\binom{n}{m})$, where m = ⌊med⌋ and med is the median of n, n − 1, n − 2, . . . , 1, 0. Since $\binom{n}{m}$ = n (n − 1)(n − 2) . . . (n − m − 1)/m!, then the worst-case time complexity is $O(n^m)$. However, the algorithm applies a property of the positive region to reduce the search space once it finds a nonreduct subset. Each such l-subset could eliminate 2^{l-1} candidate elements in L_k where k = l − 1, l − 2, . . . , 1. These nonreduct subsets are stored in NonReduct_Set. Any subset that is a subset of an element of NonReduct_Set is not included as an element of Candidate Queue.

In general, the space complexity of the breadth-first search algorithm is that all candidates remain in Candidate Queue. Therefore, it depends on the size of the largest Candidate Queue. For the best case, the space complexity is O(n). We observe that $|L_m|$ is the largest among all subset levels; where m = ⌊med⌋ and med is the median of n, n − 1, n − 2, . . . , 1, 0. Since the algorithm generates and tests level by level, its worst-case space complexity is determined by $O(\binom{n}{m}) = O(n^m)$.

5. Conclusions

This paper presents a simple and efficient solution to the problem of finding all reducts in an information system. The problem is formulated as a search problem where the search space is a rooted graph. The rooted graph is a connected graph of possible reducts and their connections. Its root is a set of all conditional attributes. Each of its k-subset nodes is connected by (k − 1)-subset nodes where k is a non-negative integer not larger than the cardinality of the graph root. The proposed algorithm searches this graph using a breadth-first search strategy, starting from the graph root. It expands nodes in breadth-first order. With the monotonic property of the positive region (Theorem 1) as the pruning rule, it can prune all nonreduct nodes in the search space early. An illustrative example is given to demonstrate the algorithm. The algorithm's efficiency is confirmed by the results of the algorithm analysis. Let n be the cardinality of conditional attributes, and m be the floor of the median of n, n − 1, n − 2, . . . , 1, 0; it is shown that both the time and space complexity of the algorithm are O (n) and O (n^m) for the best case and the worst case, respectively.

Author Contributions: Conceptualization, V.B. and P.C.;methodology, V.B.; software, V.B.; validation, V.B. and P.C.; formal analysis, V.B.; investigation, V.B.;resources, P.C.; data curation, V.B.; writing—original draft preparation, V.B.; writing—review and editing, P.C.;visualization, V.B.; supervision, V.B.; project administration, P.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A Illustrative Example

Let us consider an input to the algorithm as an information system IS (C.D) from Table 1 where {C_1,C_2,C_3,C_4,C_5}.

```
    Input: An information system IS(C.D)
Step 1:
    Calculate k = γC(D). We get k = 0.8.
Step 2:
Candidate_Queue = <C>
Reduct_Set= {}
NonReduct_Set = {}
Step 3:
    Iteration 1: Candidate_Queue is not empty.
Loop
   Get an element of Candidate_Queue and assign it to C'.
```

Then we have Candidate_Queue = <> and C' = C.
Calculate $\gamma_{C'}(D) = 0.8$.
We have $\gamma_{C'}(D) = k$. Then
Reduct_Set= {{C_1,C_2,C_3,C_4,C_5}},
Candidate_Queue = <{C_1,C_2,C_3,C_4}, {C_1,C_2,C_3,C_5}, {C_1,C_2,C_4,C_5}, {C_1,C_3,C_4,C_5},
{C_2,C_3,C_4,C_5}>, and NonReduct_Set = {}.
End Loop
Iteration 2: Candidate_Queue is not empty.
Loop
Get an element of Candidate_Queue and assign it to C'.
Then we have
Candidate_Queue = <{C_1,C_2,C_3,C_5}, {C_1,C_2,C_4,C_5}, {C_1,C_3,C_4,C_5},
{C_2,C_3,C_4,C_5}>
and C' = {C_1,C_2,C_3,C_4},.
Calculate $\gamma_{C'}(D) = 0.8$.
We have $\gamma_{C'}(D) = k$. Then Reduct_Set= {{C_1,C_2,C_3,C_4}}, Candidate_Queue
= <{C_1,C_2,C_3,C_5}, {C_1,C_2,C_4,C_5}, {C_1,C_3,C_4,C_5}, {C_2,C_3,C_4,C_5}, {C_1,C_2,C_3},
{C_1,C_2,C_4}, {C_2,C_3,C_4}, {C_1,C_3,C_4}>,
and NonReduct_Set = {}.
End Loop
Iteration 3: Candidate_Queue is not empty.
Loop
Get an element of Candidate_Queue and assign it to C'.
Then we have
Candidate_Queue = <{C_1,C_2,C_4,C_5}, {C_1,C_3,C_4,C_5}, {C_2,C_3,C_4,C_5}, {C_1,C_2,C_3},
{C_1,C_2,C_4}, {C_2,C_3,C_4}, {C_1,C_3,C_4}>
and C' = {C_1,C_2,C_3,C_5}.
Calculate $\gamma_{C'}(D) = 0.8$.
We have $\gamma_{C'}(D) = k$. Then Reduct_Set = {{C_1,C_2,C_3,C_4}, {C_1,C_2,C_3,C_5}},
Candidate_Queue = <{C_1,C_2,C_4,C_5}, {C_1,C_3,C_4,C_5}, {C_2,C_3,C_4,C_5}, {C_1,C_2,C_3},
{C_1,C_2,C_4}, {C_2,C_3,C_4}, {C_1,C_3,C_4}, {C_1,C_2,C_5}, {C_2,C_3,C_5}, {C_1,C_3,C_5} >,
and NonReduct_Set = {}.
End Loop
Iteration 4: Candidate_Queue is not empty.
Loop
Get an element of Candidate_Queue and assign it to C'.
Then we have
Candidate_Queue = <{C_1,C_3,C_4,C_5}, {C_2,C_3,C_4,C_5}, {C_1,C_2,C_3}, {C_1,C_2,C_4},
{C_2,C_3,C_4}, {C_1,C_3,C_4}, {C_1,C_2,C_5}, {C_2,C_3,C_5}, {C_1,C_3,C_5} >
and C' = {C_1,C_2,C_4,C_5}.
Calculate $\gamma_{C'}(D) = 0.6$.
We have $\gamma_{C'}(D) < k$. Then Reduct_Set= {{C_1,C_2,C_3,C_4}, {C_1,C_2,C_3,C_5}},
Candidate_Queue = <{C_1,C_3,C_4,C_5}, {C_2,C_3,C_4,C_5}, {C_1,C_2,C_3}, {C_2,C_3,C_4},
{C_1,C_3,C_4}, {C_2,C_3,C_5}, {C_1,C_3,C_5} >,
and NonReduct_Set = {{C_1,C_2,C_4,C_5}}.
End Loop
Iteration 5: Candidate_Queue is not empty.
Loop
Get an element of Candidate_Queue and assign it to C'.
Then we have

Candidate_Queue = <{C_2,C_3,C_4,C_5}, {C_1,C_2,C_3}, {C_2,C_3,C_4}, {C_1,C_3,C_4}, {C_2,C_3,C_5}, {C_1,C_3,C_5} >,
and C' = {C_1,C_3,C_4,C_5}. Calculate $\gamma_{C'}(D) = 0.8$.
We have $\gamma_{C'}(D) = k$. Then Reduct_Set= {{C_1,C_2,C_3,C_4}, {C_1,C_2,C_3,C_5}, {C_1,C_3,C_4,C_5}}, Candidate_Queue = < {C_2,C_3,C_4,C_5}, {C_1,C_2,C_3}, {C_2,C_3,C_4}, {C_1,C_3,C_4}, {C_2,C_3,C_5}, {C_1,C_3,C_5}, {C_3,C_4,C_5}>,
and NonReduct_Set = {{C_1,C_2,C_4,C_5}}.
End Loop
Iteration 6: Candidate_Queue is not empty.
Loop
Get an element of Candidate_Queue and assign it to C'.
Then we have
Candidate_Queue = <{C_1,C_2,C_3}, {C_2,C_3,C_4}, {C_1,C_3,C_4}, {C_2,C_3,C_5}, {C_1,C_3,C_5}, {C_3,C_4,C_5}>
and C' = {C_2,C_3,C_4,C_5}.
Calculate $\gamma_{C'}(D) = 0.8$.
We have $\gamma_{C'}(D) = k$. Then
Reduct_Set= {{C_1,C_2,C_3,C_4}, {C_1,C_2,C_3,C_5}, {C_1,C_3,C_4,C_5}, {C_2,C_3,C_4,C_5}},
Candidate_Queue = < {C_1,C_2,C_3}, {C_2,C_3,C_4}, {C_1,C_3,C_4}, {C_2,C_3,C_5}, {C_1,C_3,C_5}, {C_3,C_4,C_5}>,
and NonReduct_Set = {{C_1,C_2,C_4,C_5}}.
End Loop
Iteration 7: Candidate_Queue is not empty.
Loop
Get an element of Candidate_Queue and assign it to C'.
Then we have Candidate_Queue = < {C_2,C_3,C_4}, {C_1,C_3,C_4}, {C_2,C_3,C_5}, {C_1,C_3,C_5}, {C_3,C_4,C_5}>
and C' = {C_1,C_2,C_3 }.
Calculate $\gamma_{C'}(D) = 0.8$.
We have $\gamma_{C'}(D) = k$. Then Reduct_Set= {{C_1,C_3,C_4,C_5},{C_2,C_3,C_4,C_5}, {C_1,C_2,C_3 }},
Candidate_Queue = < {C_2,C_3,C_4}, {C_1,C_3,C_4}, {C_2,C_3,C_5}, {C_1,C_3,C_5}, { C_3,C_4,C_5}, {C_1,C_3 }, { C_2,C_3 }>,
and NonReduct_Set = {{C_1,C_2,C_4,C_5}}.
End Loop
Iteration 8: Candidate_Queue is not empty.
Loop
Get an element of Candidate_Queue and assign it to C'.
Then we have Candidate_Queue = <{C_1,C_3,C_4}, {C_2,C_3,C_5}, {C_1,C_3,C_5}, {C_3,C_4,C_5},{C_1,C_3}, {C_2,C_3 }>,
and C' = {C_2,C_3,C_4}.
Calculate $\gamma_{C'}(D) = 0.8$.
We have $\gamma_{C'}(D) = k$. Then Reduct_Set= {{C_1,C_3,C_4,C_5}, {C_1,C_2,C_3}, {C_2,C_3,C_4}},
Candidate_Queue = < {C_1,C_3,C_4}, {C_2,C_3,C_5}, {C_1,C_3,C_5}, {C_3,C_4,C_5}, {C_1,C_3}, {C_2,C_3}, {C_3,C_4 }>,
and NonReduct_Set = {{C_1,C_2,C_4,C_5}}.
End Loop
Iteration 9: Candidate_Queue is not empty.
Loop
Get an element of Candidate_Queue and assign it to C'.

Then we have Candidate_Queue = <{C_2,C_3,C_5}, {C_1,C_3,C_5}, {C_3,C_4,C_5}, {C_1,C_3 },
{C_2,C_3}, {C_3,C_4}>,
and C' = {C_1,C_3,C_4}.
Calculate $\gamma_{C'}(D)$ = 0.8.
We have $\gamma_{C'}(D)$ = k. Then Reduct_Set= {{C_1,C_2,C_3},{C_2,C_3,C_4}, {C_1,C_3,C_4}},
Candidate_Queue
= <{C_2,C_3,C_5}, {C_1,C_3,C_5},{C_3,C_4,C_5}, {C_1,C_3}, {C_2,C_3},{C_3,C_4}>,
and NonReduct_Set = {{C_1,C_2,C_4,C_5}}.

End Loop

Iteration 10: Candidate_Queue is not empty.

Loop

Get an element of Candidate_Queue and assign it to C'.
Then we have Candidate_Queue = <{C_1,C_3,C_5}, {C_3,C_4,C_5}, {C_1,C_3}, {C_2,C_3},
{C_3,C_4}>,
and C' = {C_2,C_3,C_5}.
Calculate $\gamma_{C'}(D)$ = 0.8.
We have $\gamma_{C'}(D)$ = k. Then Reduct_Set= {{C_1,C_2,C_3}, {C_2,C_3,C_4}, {C_1,C_3,C_4},
{C_2,C_3,C_5}},
Candidate_Queue = <{C_1,C_3,C_5}, {C_3,C_4,C_5}, {C_1,C_3 }, {C_2,C_3}, {C_3,C_4}, {C_3,C_5}>,
and NonReduct_Set = {{C_1,C_2,C_4,C_5}}.

End Loop

Iteration 11: Candidate_Queue is not empty

Loop

Get an element of Candidate_Queue and assign it to C'.
Then we have Candidate_Queue = <{C_3,C_4,C_5}, {C_1,C_3}, {C_2,C_3},{C_3,C_4}, {C_3,C_5}>,
and C' = {C_1,C_3,C_5}.
Calculate $\gamma_{C'}(D)$ = 0.4.
We have $\gamma_{C'}(D)$ < k. Then
Reduct_Set= {{C_1,C_2,C_3}, {C_2,C_3,C_4}, {C_1,C_3,C_4}, {C_2,C_3,C_5}},
Candidate_Queue = <{C_3,C_4,C_5}, {C_2,C_3},{C_3,C_4}>,
and NonReduct_Set = {{C_1,C_2,C_4,C_5}, {C_1,C_3,C_5}}.

End Loop

Iteration 12: Candidate_Queue is not empty.

Loop

Get an element of Candidate_Queue and assign it to C'.
Then we have Candidate_Queue = <{C_2,C_3},{C_3,C_4}>
and C' = {C_3,C_4,C_5}.
Calculate $\gamma_{C'}(D)$ = 0.6.
We have $\gamma_{C'}(D)$ < k. Then
Reduct_Set= {{C_1,C_2,C_3},{C_2,C_3,C_4}, {C_1,C_3,C_4}, {C_2,C_3,C_5}},
Candidate_Queue = < {C_2,C_3}>,
and NonReduct_Set = {{C_1,C_2,C_4,C_5}, {C_1,C_3,C_5}, { C3, C4, C5}}.

End Loop

Iteration 13: Candidate_Queue is not empty.

Loop

Get an element of Candidate_Queue and assign it to C'.
Then we have Candidate_Queue = < >
and C' = {C_2,C_3}.
Calculate $\gamma_{C'}(D)$ = 0.8.
We have $\gamma_{C'}(D)$ = k. Then, Reduct_Set = {{C_1,C_3,C_4}, {C_2,C_3}},

```
    Candidate_Queue = < >,
    and NonReduct_Set = {{C1,C2,C4,C5}, {C1,C3,C5}, { C3, C4, C5}}.
End Loop
Step 4:
    Return Reduct_Set = {{C1,C3,C4}, {C2,C3}}.
```

The algorithm obtains a result in 13 iterations. That is to say, only 13 subsets, of $2^5 - 1 = 31$ subsets, are explored to get all reducts. Therefore, it is a feasible solution to the problem of finding all reducts of an information system.

References

1. Pawlak, Z. *Rough Sets: Theoretical Aspects of Reasoning about Data*; Kluwer: Dordrecht, The Netherlands, 1991.
2. Hu, X.H.; Cercone, N. Learning in relational databases: A rough set approach. *Int. J. Comput. Intell.* **1995**, *11*, 323–338. [CrossRef]
3. Miao, D.Q.; Hu, G.R. A heuristic algorithm for reduction of knowledge. *J. Comput. Res. Dev.* **1999**, *36*, 681–684.
4. Qian, Y.H.; Liang, J.Y.; Pedrycz, W.; Dang, C.Y. Positive approximation: An accelarator for attribute reduction in rough set thoery. *Artifical Intell.* **2010**, *174*, 595–618. [CrossRef]
5. Dai, J.H.; Hu, H.; Zheng, G.J.; Hu, Q.H.; Han, H.F.; Shi, H. Attribute reduction in interval-valued information systems based on information entropies. *Front. Inf. Technol. Electron. Eng.* **2016**, *17*, 919–928. [CrossRef]
6. Benouini, R.; Batioua, I.; Ezghari, S.; Zenkouar, K.; Zahi, A. Fast feature selection algorithm for neighborhood rough set model based on Bucket and Trie structures. *Granul. Comput.* **2019**, 1–9. [CrossRef]
7. Chebrolu, S.; Sanjeevi, S.G. Attribute reduction on real-valued data in rough set theory using hybrid artificial bee colony: Extended FTSBPSD algorithm. *Soft Comput.* **2017**, *21*, 7543–7569. [CrossRef]
8. Chebrolu, S.; Sanjeevi, S.G. Attribute reduction in decision-theoretic rough set models using genetic algorithm. In Proceedings of the International Conference on Swarm, Evolutionary, and Memetic Computing (LNCS 7076), Visakhapatnam, India, 19–21 December 2011; pp. 307–314.
9. Chebrolu, S.; Sanjeevi, S.G. Attribute reduction on continuous data in rough set theory using ant colony optimization metaheuristic. In Proceedings of the Third International Symposiumon Women in Computing and Informatics, Kochi, India, 10–13 August 2015; pp. 17–24.
10. Chebrolu, S.; Sanjeevi, S.G. Attribute reduction in decision-theoretic rough set model using particle swarm optimization with the threshold parameters determined using LMS training rule. *Procedia Comput. Sci.* **2015**, *57*, 527–536. [CrossRef]
11. Chen, Y.M.; Miao, D.Q.; Wang, R.Z. A rough set approach to feature selection based on ant colony optimization. *Pattern Recognit. Lett.* **2010**, *31*, 226–233. [CrossRef]
12. Min, F.; Zhang, Z.H.; Dong, J. Ant colony optimization with partial-complete searching for attribute reduction. *J. Comput. Sci.* **2018**, *25*, 170–182. [CrossRef]
13. Jia, X.Y.; Liao, W.H.; Tang, Z.M.; Shang, L. Minimum cost attribute reduction in decision-theoretic rough set models. *Inf. Sci.* **2013**, *219*, 151–167. [CrossRef]
14. Cheng, Y.; Zheng, Z.R.; Wang, J.; Yang, L.; Wan, S.H. Attribute reduction based on genetic algorithm for the coevolution of meteorological data in the industrial internet of things. *Wirel. Commun. Mob. Comput.* **2019**, *2019*, 3525347. [CrossRef]
15. Zhang, N.; Gao, X.Y.; Yu, T.Y. Heuristic Approaches to Attribute Reduction for Generalized Decision Preservation. *Appl. Sci.* **2019**, *9*, 2841. [CrossRef]
16. Starzyk, J.; Nelson, D.E.; Sturtz, K. Reduct generation in information systems. *Bull. Int. Rough Set Soc.* **1998**, *3*, 19–22.
17. Chen, Y.; Miao, D.; Wang, R.; Wu, K. A rough set approach to feature selection based on power set tree. *Knowl. Based Syst.* **2011**, *24*, 275–281. [CrossRef]

18. Rezvan, M.T.; Hamadani, A.Z.; Hejazi, S.R. An exact feature selection algorithm based on rough set theory. *Complexity* **2015**, *20*, 50–62. [CrossRef]
19. Li, H.; Zhou, X.; Zhao, J.; Liu, D. Attribute Reduction in Decision-Theoretic Rough Set Model: A Further Investigation. In Proceedings of the Rough Sets and Knowledge Technology—6th International Conference, RSKT 2011, Banff, AB, Canada, 9–12 October 2011; pp. 466–475.