Yao, Y.Y, Zhao, Y. and Wang, J., On reduct construction algorithms,
Rough Sets and Knowledge Technology, First International Conference, RSKT 2006, Proceedings,
LNAI 4062, pp. 297-304, 2006.

On Reduct Construction Algorithms

Yiyu Yao[1], Yan Zhao[1] and Jue Wang[2]

[1] Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada S4S 0A2
{yyao, yanzhao}@cs.uregina.ca
[2] Laboratory of Complex Systems and Intelligence Science, Institute of Automation
Chinese Academy of Sciences, Beijing, China 100080

Abstract. This paper critically analyzes reduct construction methods at two levels. At a high level, one can abstract commonalities from the existing algorithms, and classify them into three basic groups based on the underlying control structures. At a low level, by adopting different heuristics or fitness functions for attribute selection, one is able to derive most of the existing algorithms. The analysis brings new insights into the problem of reduct construction, and provides guidelines for the design of new algorithms.

Keywords: reduct construction algorithms, deletion strategy, addition-deletion strategy, addition strategy, attribute selection heuristics.

1 Introduction

The theory of rough sets has been applied in data analysis, data mining and knowledge discovery. A fundamental notion supporting such applications is the concept of reducts, which has been studied extensively by many authors [5, 7, 9, 10, 12, 14, 15]. A reduct is a subset of attributes that is jointly sufficient and individually necessary for preserving the same information under consideration as provided by the entire set of attributes. A review of the existing reduct construction algorithms shows that most of them tie together search strategies (i.e., control structures) and attribute selection heuristics. This leads to difficulties in analyzing, comparing, and classifying those algorithms, as well as the trend of introducing new algorithms constantly.

There are two basic search strategies. The addition strategy starts with the empty set and consecutively adds one attribute at a time until we obtain a reduct, or a superset of a reduct. The deletion strategy starts with the full set and consecutively delete one attribute at a time until we obtain a reduct. By considering the properties of reducts, the deletion strategy always results in a reduct [15]. On the other hand, algorithms based on a straightforward application of the addition strategy only produce a superset of a reduct [3, 4, 6, 8]. In order to resolve this problem, many authors considered a combined strategy by re-applying the deletion strategy on the superset of a reduct produced by the

addition strategy [1, 12, 14]. According to the above discussion, we have three control strategies used by reduct construction algorithms. They are the deletion strategy, the addition-deletion strategy, and the addition strategy.

With a clear separation of control structures and attribute selection heuristics, we can critically analyze reduct construction algorithms with respect to the high level control strategies, and the low level attribute selection heuristics, respectively. This allows us to conclude that the differences between the existing algorithms lie more on the attribute selection heuristics than on the control strategies.

2 Basic Concepts and Notations

We assume that data are represented by an information table, where a finite set of objects are described by a finite set of attributes.

Definition 1. *An information table S is the tuple:*

$$S = (U, At, \{V_a | a \in At\}, \{I_a | a \in At\}),$$

where U is a finite nonempty set of objects, At is a finite nonempty set of attributes, V_a is a nonempty set of values for an attribute $a \in At$, and $I_a : U \rightarrow V_a$ is an information function. For an object $x \in U$, an attribute $a \in At$, and a value $v \in V_a$, $I_a(x) = v$ means that the object x has the value v on attribute a.

A discernibility matrix stores attributes that differentiate any two objects of the universe [9].

Definition 2. *Let $|U|$ denote the cardinality of U. Given an information table S, its discernibility matrix, denoted by M, is a $|U| \times |U|$ matrix with $m_{x,y} \in M$ defined by:*

$$m_{x,y} = \{a \in At \mid I_a(x) \neq I_a(y), x, y \in U\}.$$

The physical meaning of $m_{x,y}$ is that objects x and y are distinguished by any of the attributes in $m_{x,y}$.

For any subset $A \subseteq At$, there is an associated equivalence relation $E_A \subseteq U \times U$, i.e.,

$$E_A = \{(x, y) \in U \times U \mid \forall a \in A \ [I_a(x) = I_a(y)]\},$$

which partitions U into disjoint subsets. Such a partition of the universe is denoted by U/E_A.

The partition U/E_{At} is the finest partition, and the partition $U/E_\emptyset$ is the coarsest partition. Given an arbitrary attribute set $A \subseteq At$, the partition U/E_A is not necessarily equivalent to the partition U/E_{At}. A set of attributes that individually necessary and jointly sufficient preserve the partition of U/E_{At} is called a reduct [7].

Definition 3. *Given an information table S, a subset $R \subseteq At$ is called a reduct of At, if R satisfies the two conditions:*

(i). $U/E_R = U/E_{At}$;
(ii). *for any* $a \in R$, $U/E_{(R-\{a\})} \neq U/E_{At}$.

Based on a discernibility matrix M, a reduct can be redefined as follows [9], which is equivalent to Definition 3.

Definition 4. *Given a discernibility matrix* M, *a subset* $R \subseteq At$ *is called a reduct of* At, *if* R *satisfies the two conditions:*

(i). *for all* $m \in M$, $m \cap R \neq \emptyset$;
(ii). *for any* $a \in R$, *there exists at least one* $m \in M$ *such that* $m \cap (R - \{a\}) = \emptyset$.

In many cases, we consider decision-relative reducts instead of (absolute) reducts in a decision table. A *decision table* is an information table, with $At = C \cup D$, where C stands for a set of condition attributes that describe the features of objects, and D is a set of decision attributes.

Definition 5. *Given a consistent decision table* $S = \{U, At = C \cup D, \{V_a\}, \{I_a\}\}$, *a subset* $R \subseteq C$ *is called a relative-reduct of* C *with respect to* D, *if* R *satisfies the two conditions:*

(i). $U/E_R \preceq U/E_D$;
(ii). *for any* $a \in R$, $\neg(U/E_{(R-\{a\})} \preceq U/E_D)$,

where $\preceq$ *stands for the refinement relation between partitions.*

Based on a decision table, we can easily construct a discernibility matrix that only keeps track of the differences between any two objects that have different decision values. We can use this redefined discernibility matrix to compute the decision-relative reduct based on the same two conditions in Definition 4.

In this paper, we focus on computing the absolute reducts. The decision-relative reducts can be computed in a similar manner.

Given an information table, there may exist many reducts. The intersection of all reducts is called the core. The union of the singleton matrix elements composes the core of the attribute set [9].

An attribute set $R' \subseteq At$ is called a super-reduct of a reduct R, if $R' \supseteq R$; an attribute set $R' \subset At$ is called a partial reduct of a reduct R, if $R' \subseteq R$. Given a reduct, there exist many super-reducts and many partial reducts.

3 Three Reduct Construction Strategies

3.1 Reduct construction by deletion

By a deletion method, we take At as a super-reduct, which is the largest super-reduct. Deletion methods can be described generally as in Figure 1. Many algorithms are proposed based on this simple control strategy [3, 15].

Input: An information table.
Output: A reduct R.
(1) $R = At, CD = At$.
(2) While $CD \neq \emptyset$:
(2.1) Compute fitness of all the attributes in CD using a fitness function δ;
(2.2) Select an attribute $a \in CD$ according to its fitness, let $CD = CD - \{a\}$;
(2.3) If $R - \{a\}$ is a super-reduct, let $R = R - \{a\}$.
(3) Output R.

Fig. 1. Deletion method for computing a reduct

A deletion method starts with the trivial super-reduct, i.e., the entire attribute set. It has to check all the attributes in At for deletion. It is not efficient in the cases when a reduct is short, and many attributes are eliminated from the super-reduct after checking.

The order of attributes for deletion is essential for reduct construction. Different fitness functions determine different orders of attributes, and result in different reducts. The attribute selection heuristic is given by a fitness function:

$$\delta : At \longrightarrow \Re, \tag{1}$$

where $\Re$ is the set of real numbers. The meaning of the function δ is determined by many semantic considerations. For example, it may be interpreted in terms of the cost of testing, the easiness of understanding, or the actionability of an attribute, the information gain it produces, etc.

Many algorithms use entropy-based heuristics, such as information gain and mutual information [2, 6, 11, 13]. For example, the attribute entropy is given by:

$$\delta(a) = H(a) = -\sum_{x \in V_a} p(x) \log p(x). \tag{2}$$

Some algorithms use frequency-based heuristics with respect to the discernibility matrix, such as the ones reported in [7, 10, 12]. For example, we have:

$$\delta(a) = |\{m \in M \mid a \in m\}|. \tag{3}$$

This is, we attempt to delete first an attribute that differentiates a small number of objects.

3.2 Reduct construction by addition-deletion

By the addition-deletion strategy, we start the construction from an empty set or the core, and consequently add attributes until a super-reduct is obtained. The constructed super-reduct contains a reduct, but itself is not necessary a reduct unless it is shown that all the attributes in it are necessary. We need to delete the unnecessary attributes in the super-reduct till a reduct is found [14, 15]. The addition-deletion methods can be described generally as in Figure 2.

Input: An information table.
Output: A reduct R.
Addition:
(1) $R = \emptyset, CA = At$.
(2) While R is not a super-reduct and $CA \neq \emptyset$:
(2.1) Compute fitness of all the attributes in CA using a fitness function σ;
(2.2) Select an attribute $a \in CA$ according to its fitness, let $CA = CA - \{a\}$;
(2.3) Let $R = R \cup \{a\}$.
Deletion:
(3) $CD = R$.
(4) While $CD \neq \emptyset$:
(4.1) Compute fitness of all the attributes in CD using a fitness function δ;
(4.2) Select an attribute $a \in CD$ according to its fitness, let $CD = CD - \{a\}$;
(4.3) If $R - \{a\}$ is a super-reduct, let $R = R - \{a\}$.
(5) Output R.

Fig. 2. Addition-deletion method for computing a reduct

The addition-deletion strategy has been proposed and studied, since the deletion strategy is not efficient, and the over-simplified addition methods normally find a super-reduct, but not a reduct. A lack of consideration of the latter problem has produced many incomplete reduct construction algorithms, such as the ones reported in [3, 4, 6, 8]. An addition-deletion algorithm based on the discernibility matrix has been proposed by Wang and Wang [12], which can construct a super-reduct, and reduce it to a reduct efficiently.

For the addition-deletion strategies, the orders of attributes for addition and deletion are both essential for reduct construction. By using the fitness function σ, we add the fit attributes to the empty set or the core to form a super-reduct; by using the fitness function δ, we delete the fit attributes from the super-reduct to form a reduct. σ and δ can be two different heuristics, or the same heuristic. If one can order the attributes according to a fitness function δ from the most fit attribute to the least fit attribute, then this order can be used for adding them one by one until the sufficient condition is met, and the reversed order can be used for deleting the unnecessary attributes. By this means, one heuristic determines two orders, and a reduct composed of more fit attributes is obtained.

3.3 Reduct construction by addition

The goal of an addition method is to construct a reduct from an empty set or the core, and consequently add attributes until it becomes a reduct. The essential difference between the addition method and the addition-deletion method is that, the addition method takes in one attribute if the constructed set is a partial reduct. On the other hand, the addition-deletion method continuously adds attributes until a super-reduct is produced. In this case, superfluous attributes could be added, and the deletion process is required to eliminate them. The addition methods can be described generally as in Figure 3.

Input: An information table.
Output: A reduct R.
(1) $R = \emptyset,\ CA = At$.
(2) While R is not a reduct and $CA \neq \emptyset$:
(2.1) Compute fitness of all the attributes in CA using a fitness function σ;
(2.2) Select an attribute $a \in CA$ according to its fitness;
(2.3) If $R \cup \{a\}$ is a partial reduct, let $R = R \cup \{a\}$, and $CA = CA - \{a\}$.
(3) Output R.

Fig. 3. Addition method for computing a reduct

The process to check if a constructed attribute set is a partial reduct is not a trivial step. Zhao and Wang have proposed an algorithm to carry out this task [14]. Before we introduce this algorithm, we need to introduce two basic operations defined on a discernibility matrix:

Absorb is an absorption operation on the discernibility matrix. One can absorb a matrix element $m \in M$ if there exists another matrix element $m' \in M$ such that $m' \subseteq m$. It means that if two objects can be distinguished by any attribute in the matrix element m, then they can also be distinguished by any attribute in a subset of m. We do not need to track the supersets, but only the subsets, the absorbers. The operation is defined as:

$$Absorb(M):\ \text{For any } m, m' \in M,\ \text{if } m' \subseteq m,\ \text{then } M = M - \{m\}.$$

Group is a grouping operation on elements of a discernibility matrix. A set of matrix elements can be grouped together with respect to an attribute by collecting all the individual matrix elements containing the attribute. Since each matrix element is associated with two objects, the grouping reflects the fact that a set of objects associated with the grouped matrix elements can be distinguished by this common attribute. We only need to track this common attribute for simplicity. For an attribute $a \in At$, the grouping is defined as:

$$Group(a) = \{m \in M \mid a \in m\}.$$

An addition algorithm for computing reducts based on a discernibility matrix is described in Figure 4.

The fitness function σ can be the one discussed in Sections 3.2. We need to discuss more about the fitness function dm. To ensure that the chosen attribute a is in a partial reduct, we need to choose one element $m \in Group(a)$, and delete m from all the matrix element in M, and update M accordingly. This deletion, here for simplicity, is called "delete the tail of a", ensures that a has to be a reduct attribute, otherwise, at least one pair of objects cannot be distinguished. We should note that the fitness function dm of the proposed addition algorithm is different from the fitness function δ of the general deletion algorithm we discussed in Section 3.1. That is because δ evaluates the fitness of one single attribute at a time, dm evaluates the fitness of a matrix element

Input: A discernibility matrix M.
Output: A reduct.
$R = \emptyset, CA = At$.
Do while $M \neq \emptyset$:
(1) $M = Absorb(M)$.
(2) Compute fitness of all the attributes in CA using a fitness function σ;
(3) Select an attribute $a \in CA$ with $Group(a) \neq \emptyset$ according to its fitness, let $R = R \cup \{a\}, CA = CA - \{a\}$;
(4) Compute fitness of all the matrix elements in $Group(a)$ according to a fitness function dm;
(5) Select a matrix element $m_i \in Group(a)$ based on its fitness, update M by two steps:
(5.1) Delete $Group(a)$ from M: $M = M - Group(a)$,
(5.2) Update matrix elements: $M = \{m - m_i \mid m \in M\}$.
Output the reduct R.

Fig. 4. An addition algorithm by using a discernibility matrix

m, which is a set of attributes. Typically, dm is the summation or the average fitness of all the included attributes.

The selection of a matrix element for deletion can be described by a mapping:

$$dm : \{m \mid m \in Group(a)\} \longrightarrow \Re. \tag{4}$$

The meaning of the mapping function dm is determined by many semantic considerations as well.

A frequency-based heuristic can be defined as follows. The higher the value, the more matrix elements are to be updated, and most possibly, after absorption, a smaller matrix can be obtained. That is,

$$dm(m_i) = |\{m \in M \mid m \cap m_i \neq \emptyset\}|. \tag{5}$$

We can also define the fitness function dm as the information entropy, i.e., the joint entropy of all the attributes in the attribute set $m_i - \{a\}$. For example, if $m_i - \{a\} = \{b, c\}$, then

$$\begin{aligned} dm(m_i) &= H(m_i - \{a\}) \\ &= H(\{b, c\}) \\ &= -\sum_{x \in V_b} \sum_{y \in V_c} p(b, c) \log p(b, c). \end{aligned} \tag{6}$$

4 Conclusion

This paper provides a critical study of the existing reduct construction algorithms based on a two-level view: a high level view of control strategy and a low level view of attribute selection heuristic. Three basic groups are discussed. They are the deletion strategy, the addition-deletion strategy, and the addition

strategy. Several attribute selection heuristics are examined. The analysis not only produces valuable insights into the problem, but also provides guidelines for the design of new reduct construction algorithms.

References

1. Bazan, J.G., Nguyen, H.S., Nguyen, S.H., Synak, P., Wroblewski, J.: Rough set algorithms in classification problem. In: Polkowski, L., Tsumoto, S., Lin, T.Y. (Ed.), *Rough Set Methods and Applications* (2000) 49-88.
2. Beaubouef, T., Petry, F.E., Arora, G.: Information-theoretic measures of uncertainty for rough sets and rough relational databases. *Information Sciences* **109** (1998) 185-195.
3. Hu, X., Cercone, N.: Learning in relational databases: a rough set approach. *Computation Intelligence: An International Journal* **11** (1995) 323-338.
4. Jenson, R., Shen, Q.: A rough set-aided system for sorting WWW bookmarks. In: Zhong, N. et al. (Eds.), *Web Intelligence: Research and Development* (2001) 95-105.
5. Mi, J.S., Wu, W.Z., Zhang, W.X.: Approaches to knowledge reduction based on variable precision rough set model. *Information Sciences* **159** (2004) 255-272.
6. Miao, D., Wang, J.: An information representation of the concepts and operations in rough set theory. *Journal of Software* **10** (1999) 113-116.
7. Pawlak, Z.: *Rough Sets: Theoretical Aspects of Reasoning About Data*, Kluwer, Boston (1991).
8. Shen, Q., Chouchoulas, A.: A modular approach to generating fuzzy rules with reduced attributes for the monitoring of complex systems. *Engineering Applications of Artificial Intelligence* **13** (2000) 263-278.
9. Skowron, A., Rauszer, C.: The discernibility matrices and functions in information systems. In: Slowiński, R. (Ed.), *Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory.* Dordrecht, Kluwer (1992).
10. Slezak, D., Various approaches to reasoning with frequency based decision reducts: a survey. In: Polkowski, L., Tsumoto, S., Lin, T.Y. (Eds.), *Rough set methods and applications.* Physica-verlag, Heidelberg (2000) 235-285.
11. Wang, G., Yu, H., Yang, D.: Decision table reduction based on conditional information entropy. *Chinese Journal of Computers* **25** (2002) 759-766.
12. Wang, J., Wang, J.: Reduction algorithms based on discernibility matrix: the ordered attributes method. *Journal of Computer Science and Technology* **16** (2001) 489-504.
13. Yu, H., Yang, D., Wu, Z., Li, H.: Rough set based attribute reduction algorithm. *Computer Engineering and Applications* **17** (2001) 22-47.
14. Zhao, K., Wang, J.: A reduction algorithm meeting users' requirements. *Journal of Computer Science and Technology* **17** (2002) 578-593.
15. Ziarko, W.: Rough set approaches for discovering rules and attribute dependencies. In: Klösgen, W., Żytkow, J.M. (Eds.), *Handbook of Data Mining and Knowledge Discovery.* Oxford (2002) 328-339.