Local Search for Attribute Reduction

Xiaojun Xie[1,2], Ryszard Janicki[2], Xiaolin Qin[1(✉)], Wei Zhao[2], and Guangmei Huang[3]

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
{xiexj,qinxcs}@nuaa.edu.cn

[2] Department of Computing and Software, McMaster University, Hamilton L8S 4K1, Canada
{janicki,zhaow9}@mcmaster.ca

[3] Faculty of Education, Guangxi Normal University, Guilin 541001, China
guangmeihuang@126.com

Abstract. Two new attribute reduction algorithms based on iterated local search and rough sets are proposed. Both algorithms start with a greedy construction of a relative reduct. Then attempts to remove some attributes to make the reduct smaller. Process of attributes selection is the main difference between the algorithms. It is random for the first one, and a sophisticated selection procedure is used for the second algorithm. Moreover a fixed number of iterations is assumed for the first algorithms whereas the second stops when a local optimum is reached. Various experiments using eight well-known data sets from UCI have been made and they show substantial superiority of our algorithms.

Keywords: Rough set · Attribute reduction · Local search · Positive region

1 Introduction

Feature selection, or attribute reduction, is a process of finding a minimal subset of attributes that still provides the same, or similar information as the set of all original attributes. Rough set theory has been very successful as a theoretical base used in filter-based feature selection algorithms in many fields, such as data mining, machine learning, pattern recognition and many others [1–7].

Attribute reduction methods can be divided into four categories: exact algorithms, approximation algorithms, general heuristic algorithms and metaheuristic algorithms.

Exact algorithms can find all reducts and an optimal reduct. The classical exact algorithm [8], consists in finding the discernibility matrix first, then deriving the discernibility function in its conjunctive normal form (CNF) from it, and at the end transforming CNF into DNF i.e. disjunctive normal form. Then, each prime implicant of the DNF corresponds to a reduct, and each minimal prime implicant of the DNF corresponds to an optimal reduct. Unfortunately, finding all reducts or an optimal reduct has been proven to be in general an NP-hard problem [8,9], which is a problem for big data sets with many attributes and objects.

Several efficient approximation algorithms have been proposed in recent years. Yang et al. [12] provided a new efficient method based on related family for computing all attribute reducts and relative attribute reducts. Tan et al. [10] proposed very time efficient matrix based approximation algorithm by introducing the concepts of minimal and maximal descriptions. Hacibeyoglu et al. [11] analyzed the main shortcoming of this algorithm, namely is its excessively high space complexity, and proposed a substantial improvement with the worst case space complexity of $\binom{N}{N/2}/2$, where N is the number of attributes.

For many big real-world applications, efficiency of approximation algorithms is still not enough. Frequently it is also not necessary to find all reducts, on contrary, quite often finding one reduct is enough, which leads to the idea of looking for heuristic algorithms.

The general heuristic algorithm normally starts with the core attribute set or an empty attribute set, then gradually adds an attribute with the maximal significance into the attribute reduct until the attribute reduct satisfies the stopping criterion. Different models have been used for stopping criteria, namely positive region [13], information entropy [14], knowledge granularity [15], and other models [16,17].

General heuristic algorithms usually fail to obtain an optimal reduct, so many meta-heuristic algorithms have been proposed such as genetic algorithms, tabu search, ant colony optimization, particle swarm optimization and artificial fish swarm algorithm, and so on. In [18], Xu et al. illustrated the shortcomings of the previous genetic algorithm-based methods and designed new fitness function, which resulted in more efficient genetic algorithm. Chen et al. [19] provided a novel rough set based method to feature selection using fish swarm algorithm. Inbarani et al. [20] proposed a supervised feature selection method based on quick reduct and improved harmony search. Luan et al. [21] developed a novel attribute reduction algorithm based on rough set and improved artificial fish swarm algorithm. Aziz and Hassanien [22] proposed an improved social spider algorithm for the minimal reduction problem. Xie et al. [23] designed a test-cost-sensitive rough set-based algorithm for the minimum weight vertex cover problem, which can also be used to solve attribute reduction problem in rough sets.

Nevertheless, for big data sets with huge number of attributes and objects, meta-heuristic algorithms are often still not sufficiently efficient. In recent years, *local search* has been shown to be an effective and promising approach to solve many NP-hard problems, such as, for example, the minimum vertex cover problem [24,25]. In this paper we will design, discuss and test two new algorithms for attribute reduction that is based on local search paradigm. The main ideas of these two algorithms can be described as follows (Fig. 1).

If a reduct has been obtained, then an upper bound of the target problem has also been found. Then, we decrease the upper bound by removing an attribute

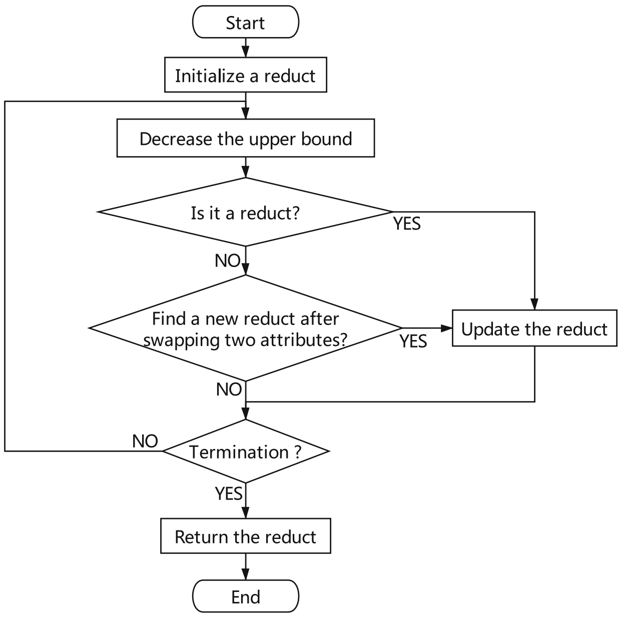

Fig. 1. Basic flowchart of our two algorithms. Procedures for termination and finding new reducts are different in each algorithm.

from the current reduct. The outcome may or may not be a reduct. If it is not a reduct, we swap attributes, one attribute from the current candidate reduct and the other that does not belong to the current candidate reduct. If the result is a reduct, it has smaller, i.e. better, upper bound. We continue this process as long as it is possible, the outcome is a relatively small reduct or even an optimal reduct.

Finding a new relative reduct after swapping two attributes is the key process in each iteration and the difference between our two algorithms. To make this efficient, the second algorithm uses the reverse incremental verification to check if a swapping results in a reduct. The second algorithm also uses a set of *removed attributes* to adjust the iteration process, which additionally improves the efficiency of our algorithm. Moreover the second algorithm stops when a local optimum is found while the first one performs given in advance number of iterations.

The rest of the paper is organized as follow. In Sect. 2, basic concepts about rough sets are introduced. Section 3 exposes the local search-based algorithms for attribute reduction. Experimental results on UCI data sets are presented in Sect. 4. Some conclusions and further researches are drawn in Sect. 5.

2 Preliminaries

This section recalls some basic concepts, definitions and notation used in this paper.

For any equivalence relation $R \subseteq U \times U$, where U is a set, $[x]_R$ denotes the equivalence class containing $x \in U$, i.e. $[x]_R = \{y \mid (x, y) \in R\}$, and U/R denotes the partition of U defined by R, i.e. $U/R = \{[x]_R \mid x \in U\}$.

A *decision table* is the 5-tuple: $S = (U, C, D, V, f)$, where U is a finite nonempty set of objects, called *universe*, C is a set of *conditional attributes*, D is a set of *decision attributes*, V is *domain of attributes* $C \cup D$ and $f : U \times (C \cup D) \to V$ is an *information function.*

Table 1 is a simple example of a decision table, where $U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$, $C = \{a_1, a_2, a_3, a_4, a_5, a_6\}$, and $D = \{Flu\}$ (or $D = \{a_7\}$).

Table 1. An example of a decision table.

	a_1	a_2	a_3	a_4	a_5	a_6	a_7
Patient	Headache	Temperature	Lymphocyte	Leukocyte	Eosinophil	Heartbeat	Flu
x_1	Yes	High	High	High	High	Normal	Yes
x_2	Yes	High	Normal	High	High	Abnormal	Yes
x_3	Yes	High	High	High	Normal	Abnormal	Yes
x_4	No	High	Normal	Normal	Normal	Normal	No
x_5	Yes	Normal	Normal	Low	High	Abnormal	No
x_6	Yes	Normal	Low	High	Normal	Abnormal	No
x_7	Yes	Low	Low	High	Normal	Normal	Yes

Let $S = (U, C, D, V, f)$ be a decision table. For each nonempty $B \subseteq C$ or $B = D$ we define a ***indiscernibility relation induced by*** B, denoted $\mathsf{ind}(B)$, as:

$$\mathsf{ind}(B) = \{(x, y) \mid x, y \in U \wedge \forall a \in B, f(x, a) = f(y, a)\}.$$

The relation $\mathsf{ind}(B)$ is clearly an equivalence relation on U.

When $B = D$, $\mathsf{ind}(B)$ is called ***classification relation induced by*** D and denoted by $\mathcal{D}$. In this case a partition $U/\mathcal{D}$ is called *classification defined by the decision attributes* D.

For every nonempty $B \subseteq C$, and every $X \subseteq U$, we define $B_-(X)$, the B-*lower approximation* of X, as $B_-(X) = \{x \in U \mid [x]_{\mathsf{ind}(B)} \subseteq X\}$. For every for every nonempty $B \subseteq C$ and every $U' \subseteq U$, we define the ***positive region*** (or ***lower approximation***) ***of*** D ***over*** U' ***with respect to*** B as:

$$\mathsf{POS}_B^{U'}(D) = \bigcup_{X \in U'/\mathcal{D}} B_-(X).$$

When $U' = U$, the most popular case, we will just write $\mathsf{POS}_B(D)$. Note that we always have: $\mathsf{POS}_B^{U'}(D) \subseteq U'$.

Definition 1. *Let $S = (U, C, D, V, f)$ be a decision table and let $B \subseteq C$.*

1. *A set B is called a* ***relative attribute reduct*** *if and only if $\mathsf{POS}_B(D) = \mathsf{POS}_C(D)$, and*
2. *a set B is called an* ***attribute reduct*** *if and only if it is a relative reduct and for each $B' \subsetneq B$, we have $\mathsf{POS}_{B'}(D) \neq \mathsf{POS}_B(D)$,*
3. *a set B is called an* ***optimal attribute reduct*** *if and only if it is a reduct and for any other reduct B', we have $|B| \leq |B'|$.* $\diamond$

In other words, reducts are minimal relative reducts and optimal reduct is a reduct with smallest cardinality.

3 Local Search for Attribute Reduction

This section describes in detail our local search method for solving the attribute reduction problem.

3.1 A Plain Local Search Algorithm for Attribute Reduction

Our method stems from the following simple result.

Suppose $S = (U, C, D, V, f)$ is a decision table, $Red \subseteq C$ is a relative reduct, we randomly select $a \in Red$. If $Red \setminus \{a\}$ is also a relative reduct, then we update $Red \setminus \{a\}$ as new Red and jump into the next iteration. If $Red \setminus \{a\}$ is *not* a relative reduct, we randomly choose $u \in Red \setminus \{a\}$ and $v \in C \setminus Red$ and verify if $Red_{auv} = (Red \setminus \{a, u\}) \cup \{v\}$ is a relative reduct. If it is, we update Red_{auv} as new Red, and go to the next iteration. Since $|Red_{auv}| = |Red| - 1$, Red_{auv} is better relative reduct than Red. If Red_{auv} is not a relative reduct, Red is not changed and we continue with the next iteration. The algorithm stops when it iterates T times, where T is a parameter given in advance.

The process always returns a relative reduct and the bigger value of T, the smaller, i.e. better, the solution is. Algorithm 1 represents the procedure described above. The algorithm starts with a construction some relative reduct Red (steps 1 and 2). The computation is greedy, the set Red is initially empty and then, in each iteration we choose an attribute $a \in C \setminus Red$ at random and add it to Red. The computation process stops when $\mathsf{POS}_{Red}(D) = \mathsf{POS}_C(D)$. The process always converges, the worst case is when $Red = C$, so no reduct exists. The worst case time complexity is $O(\sum_{i=1}^{|Red|} i|U|) = O(|Red|^2|U|) = O(|C|^2|U|)$. Steps 3–14 represent T iterations that result in a derivation of a reduct Red_T from a relative reduct Red. Clearly $|Red_T| \leq |Red|$. The worst case time complexity of the ith iteration is $O(|Red_i||U|)$, where Red_i is Red from ith iteration, so the worst case time complexity of lines 3–14 is $O((\sum_{i=1}^{T} |Red_i|)|U|) = O(T|Red||U|) = O(T|C||U|)$ as clearly $|Red_i| \leq |Red| \leq |C|$ for all $i = 1, \ldots, T$. For the entire Algorithm 1 we have $O(\max(T, |Red|)|Red||U|) = O(\max(T, |C|)|C||U|) = O(T|C||U|)$ as usually $T > |C|$.

Algorithm 1 always finds some reducts but not necessarily an optimal reduct. The quality of solution clearly depends on the size of T, but also on smart selection of pairs (u, v). Foundations of such selection process are presented in the next section. We would also like to get rid of this arbitrary limit T and just stop when a local minimum is found.

Algorithm 1. (LSAR) Local search algorithm for attribute reduction

```
Input: A decision table $S = (U, C, D, V, f)$, the maximum number of iterations $T$
Output: The attribute reduction $Red$.
1  $t = 0$, $Red = \emptyset$;
2  construct a relative reduct $Red$ using greedy algorithm;
3  while $t < T$ do
4      remove an attribute $a$ from $Red$ randomly;
5      if $\mathsf{POS}_{Red \setminus \{a\}}(D) = \mathsf{POS}_C(D)$ then
6          $Red = Red \setminus \{a\}$;
7      else
8          select randomly the deleting attribute $u \in Red \setminus \{a\}$ and the adding attribute $v \in C \setminus Red$ ;
9          if $\mathsf{POS}_{(Red \setminus \{a,u\}) \cup \{v\}}(D) = \mathsf{POS}_C(D)$ then
10             $Red = (Red \setminus \{a, u\}) \cup \{v\}$;
11         end
12     end
13     $t = t + 1$;
14 end
15 return $Red$;
```

3.2 Attribute Pair Selection Mechanism

In principle, the basic problem we have to deal with in Algorithm 1 can be formulated as follows. Suppose that $\mathsf{POS}_B(D) \neq \mathsf{POS}_C(D)$. How to select attributes u and v such that $\mathsf{POS}_{(B \setminus \{u\}) \cup \{v\}}(D) = \mathsf{POS}_C(D)$? We will use a reverse incremental verification approach to solve this problem and start with two useful lemmas.

Lemma 1. *Let $S = (U, C, D, V, f)$ be a decision table. For each $B \subseteq C$, we have:* $\mathsf{POS}_B(D) = \mathsf{POS}_B^{\mathsf{POS}_B(D)}(D)$.

Proof. Clearly $\mathsf{POS}_B^{\mathsf{POS}_B(D)}(D) \subseteq \mathsf{POS}_B(D)$. Let $x \in \mathsf{POS}_B(D)$ and $\mathsf{POS}_B(D)/\mathcal{D}$ be the partition of $\mathsf{POS}_B(D)$ defined by D. Then $x \in X_x \in \mathsf{POS}_B(D)/\mathcal{D}$. But by the definition: $\mathsf{POS}_B^{\mathsf{POS}_B(D)}(D) = \bigcup_{X \in \mathsf{POS}_B(D)/\mathcal{D}} B_-(X)$, hence $x \in \mathsf{POS}_B^{\mathsf{POS}_B(D)}(D)$. □

Before formulating our next result we need to introduce one more concept.

Let $S = (U, C, D, V, f)$ be a decision table. For each nonempty $B \subseteq C$ we define the ***inconsistent objects pairs***, denoted $\mathsf{iop}(B)$, as:

$$\mathsf{iop}(B) = \{(x, y) \mid x, y \in U \wedge (\forall a \in B. f(x, a) = f(y, a)) \wedge (\exists d \in D. f(x, d) \neq f(y, d))\}.$$

If (x, y) forms an inconsistent object pair, then the value of all conditional attribute are the same and the values of some decision attributes are different.

Lemma 2. *Let $S = (U, C, D, V, f)$ be a decision table and $B \subseteq C$. Then we have:*

1. *For each attribute $v \in C \setminus B$,*

$$\mathsf{POS}_{B\cup\{v\}}(D) = \mathsf{POS}_B(D) \cup \mathsf{POS}^{U'}_{\{v\}}(D),$$

   where $U' = \mathsf{POS}_{B\cup\{v\}}(D) \setminus \mathsf{POS}_B(D)$.
2. *For each attribute $u \in B$,*

$$\mathsf{POS}_{B\setminus\{u\}}(D) = \mathsf{POS}_B(D) \setminus \bigcup_{X\in\mathcal{X}_u} X,$$

   where $\mathcal{X}_u = \{X \mid X \in \mathsf{POS}_B(D)/\mathsf{ind}(B) \wedge (X \times U) \cap \mathsf{iop}(B \setminus \{u\}) \neq \emptyset\}$.

Proof. (sketch) (1) First note that $U' \cap \mathsf{POS}_B(D) = \emptyset$ and $\mathsf{POS}^{U'}_{\{v\}}(D) \subseteq U'$, so $\mathsf{POS}_{B\cup\{v\}}(D) = \mathsf{POS}_B(D) \cup \mathsf{POS}^{U'}_{\{v\}}(D) \iff U' = \mathsf{POS}^{U'}_{\{v\}}(D)$. Suppose that $x \in U' \setminus \mathsf{POS}^{U'}_{\{v\}}(D)$, i.e. $x \in \mathsf{POS}_{B\cup\{v\}}(D)$, $x \notin \mathsf{POS}_B(D)$ and $x \notin \mathsf{POS}^{U'}_{\{v\}}(D)$, which clearly implies $[x]_{\mathsf{ind}(B\cup\{v\})} \subseteq \mathsf{POS}_{B\cup\{v\}}(D)$, $[x]_{\mathsf{ind}(B)} \cap \mathsf{POS}_B(D) = \emptyset$ and $[x]_{\mathsf{ind}(\{v\})} \cap \mathsf{POS}^{U'}_{\{v\}}(D) = \emptyset$. However, since $v \notin B$, we also have $\mathsf{ind}(B \cup \{v\}) = \mathsf{ind}(B) \cap \mathsf{ind}(\{v\})$, which means $[x]_{\mathsf{ind}(B\cup\{v\})} \subseteq [x]_{\mathsf{ind}(B)} \cap [x]_{\mathsf{ind}(\{v\})}$, a contradiction.
(2) Since $B\setminus\{u\} \subsetneq B$ then $\mathsf{POS}_{B\setminus\{u\}}(D) \subseteq \mathsf{POS}_B(D)$. Consider $X \in \mathcal{X}_u$. Since $X \in \mathsf{POS}_B(D)/\mathsf{ind}(B)$ then $X \subseteq \mathsf{POS}_B(D)$ and since $(X \times U) \cap \mathsf{iop}(B \setminus \{u\}) \neq \emptyset$ then $X \cap \mathsf{POS}_{B\setminus\{u\}}(D) = \emptyset$. Hence $\mathsf{POS}_{B\setminus\{u\}}(D) \subseteq \mathsf{POS}_B(D) \setminus \bigcup_{X\in\mathcal{X}_u} X$. Let $x \in \mathsf{POS}_B(D) \setminus \bigcup_{X\in\mathcal{X}_u} X$. Hence $x \in \mathsf{POS}_B(D)$ and there is $y \in U$ such that $(x, y) \notin \mathsf{iop}(B \setminus \{u\})$, i.e. $x \in \mathsf{POS}_{B\setminus\{u\}}(D)$. □

Lemma 2 shows the results of adding and deleting attributes to and from a positive region $\mathsf{POS}_B(D)$. We will use them to provide a pair selection mechanism described in Algorithm 1. More precise rules are given by the next result.

Proposition 1. *Let $S = (U, C, D, V, f)$ be a decision table, $B \subsetneq C$, $u \in B$ and $v \in C \setminus B$ such that*

- $\mathsf{POS}_B(D) \neq \mathsf{POS}_C(D)$,
- $\mathsf{POS}_{(B\setminus\{u\})\cup\{v\}}(D) = \mathsf{POS}_C(D)$ *and*
- $\mathsf{POS}_C(B)/\mathsf{ind}(B \cup \{v\}) = \{X_1, \ldots, X_n\}$.

Then the following properties hold.

1. $\mathsf{POS}^{U'}_{\{v\}}(D) = U'$*, where* $U' = \mathsf{POS}_C(D) \setminus \mathsf{POS}_B(D)$*.*
2. $\mathsf{POS}^{\widehat{U}}_{(B\setminus\{u\})\cup\{v\}}(D) = \widehat{U}$*, for every* $\widehat{U} = \{x_1, \ldots, x_n\} \subseteq U$ *such that* $\widehat{U} \cap X_i = \{x_i\}$ *for* $i = 1, \ldots, n$*.*

Proof. (sketch) (1) Since $\mathsf{POS}_{(B\setminus\{u\})\cup\{v\}}(D) = \mathsf{POS}_C(D)$, then directly from the definition of positive region we have: $\mathsf{POS}_{B\cup\{v\}}(D) = \mathsf{POS}_C(D)$. By Lemma 2(1) we have: $\mathsf{POS}_C(D) = \mathsf{POS}_{B\cup\{v\}}(D) = \mathsf{POS}_B(D) \cup \mathsf{POS}^{\mathsf{POS}_C(D)\setminus\mathsf{POS}_B(D)}_{\{v\}}(D)$, i.e. $\mathsf{POS}^{\mathsf{POS}_C(D)\setminus\mathsf{POS}_B(D)}_{\{v\}}(D) = \mathsf{POS}_C(D) \setminus \mathsf{POS}_B(D)$.
(2) From Lemma 1 it follows $\mathsf{POS}_{(B\setminus\{u\})\cup\{v\}}(D) = \mathsf{POS}^{\mathsf{POS}_{(B\setminus\{u\})\cup\{v\}}(D)}_{(B\setminus\{u\})\cup\{v\}}(D)$ and $\mathsf{POS}_{B\cup\{v\}}(D) = \mathsf{POS}^{\mathsf{POS}_{B\cup\{v\}}(D)}_{B\cup\{v\}}(D)$. However $\mathsf{POS}_{(B\setminus\{u\})\cup\{v\}}(D) = \mathsf{POS}_{B\cup\{v\}}(D) = \mathsf{POS}_C(D)$, so $\mathsf{POS}^{\mathsf{POS}_C(D)}_{(B\setminus\{u\})\cup\{v\}}(D) = \mathsf{POS}^{\mathsf{POS}_C(D)}_{B\cup\{v\}}(D)$. On the other hand, since $(B \setminus \{u\}) \cup \{v\} = (B \cup \{v\}) \setminus \{u\}$, by Lemma 2(2) we have $\mathsf{POS}^{\mathsf{POS}_C(D)}_{(B\setminus\{u\})\cup\{v\}}(D) = \mathsf{POS}^{\mathsf{POS}_C(D)}_{B\cup\{v\}}(D) \setminus \bigcup_{X\in\mathcal{X}^v_u} X$, where $\mathcal{X}^v_u = \{X \mid X \in \mathsf{POS}_C(D)/\mathsf{ind}(B\cup\{v\}) \wedge (X\times\mathsf{POS}_C(D)) \cap \mathsf{iop}((B\cup\{v\})\setminus\{u\}) \neq \emptyset\}$. This means that $\bigcup_{X\in\mathcal{X}^v_u} X = \emptyset$, i.e. $\mathcal{X}^v_u = \emptyset$, or, equivalently, $X \in \mathsf{POS}_C(D)/\mathsf{ind}(B \cup \{v\})$ implies $(X\times\mathsf{POS}_C(D)) \cap \mathsf{iop}((B\cup\{v\})\setminus\{u\}) = \emptyset$. But this also means that $X \in \mathsf{POS}_C(D)/\mathsf{ind}(B \cup \{v\}) = \{X_1, \ldots, X_n\}$ implies $X \subseteq \mathsf{POS}^{\mathsf{POS}_C(D)}_{(B\setminus\{u\})\cup\{v\}}(D)$. For each $i = 1, \ldots, n$, let x_i be an arbitrary element of X_i and set $\widehat{U} = \{x_1, \ldots, x_n\}$. If $i \neq j$, then we now have $(x_i, x_j) \in \mathsf{ind}((B \setminus \{u\}) \cup \{v\})$ and $f(x_i, d) = f(x_j, d)$ for each $d \in D$. But this means that we have $\mathsf{POS}^{\widehat{U}}_{(B\setminus\{u\})\cup\{v\}}(D) = \widehat{U}$. □

Proposition 1 suggests the following useful definition. Let $S = (U, C, D, V, f)$ be a decision table, $B \subsetneq C$ and $U' = \mathsf{POS}_C(D) \setminus \mathsf{POS}_B(D)$. We define $C^*_B \subseteq C$, a ***set of attributes filtered by*** B as:

$$C^*_B = \{v \mid v \in C \setminus B \wedge \mathsf{POS}^{U'}_{\{v\}}(D) = U'\}.$$

We will now show a sample application of the results stated above.

Example 1. Take the decision table Table 1, where $U = \{x_1, x_2, \ldots, x_7\}$, $C = \{a_1, a_2, \ldots, a_6\}$, and $D = \{Flu\}$. Consider $B = \{a_1, a_4\}$. In this case $\mathsf{POS}_{\{a_1,a_4\}}(D) = \{x_4, x_5\}$ and $\mathsf{POS}_C(D) = U$. We want to find such $u \in B = \{a_1, a_4\}$ and $v \in C \setminus B = \{a_2, a_3, a_5, a_6\}$ that $\mathsf{POS}_{(\{a_1,a_4\}\setminus\{u\})\cup\{v\}}(D) = \mathsf{POS}_C(D)$. We have to perform the following steps.

1. First we compute U' as defined in Proposition 1(1). In this case $U' = \mathsf{POS}_C(D) \setminus \mathsf{POS}_{\{a_1,a_4\}}(D) = \{x_1, x_2, x_3, x_6, x_7\}$.
2. For each $v \in \{a_2, a_3, a_5, a_6\}$, we compute $\mathsf{POS}^{U'}_{\{v\}}(D)$ and for this case we have: $\mathsf{POS}^{U'}_{\{a_2\}}(D) = \{x_1, x_2, x_3, x_6, x_7\}$, $\mathsf{POS}^{U'}_{\{a_3\}}(D) = \{x_1, x_2, x_3\}$, $\mathsf{POS}^{U'}_{\{a_5\}}(D) = \{x_1, x_2\}$ and $\mathsf{POS}^{U'}_{\{a_6\}}(D) = \{x_1, x_7\}$.

3. We now can calculate $C^*_{\{a_1,a_4\}}$. Only $\mathsf{POS}^{U'}_{\{a_2\}}(D) = U'$, so $C^*_{\{a_1,a_4\}} = \{a_2\}$, i.e. we set $v = a_2$.
4. We calculate $\mathsf{POS}_{B\cup\{v\}}(D) = \mathsf{POS}_{\{a_1,a_4\}\cup\{a_2\}}(D) = \mathsf{POS}_{\{a_1,a_2,a_4\}}(D) = U$.
5. We calculate that $\mathsf{POS}_{\{a_1,a_2,a_4\}}(D)/\mathsf{ind}(\{a_1, a_2, a_4\}) = \{\{x_1, x_2, x_3\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\}\}$, and construct $\widehat{U}$ as $\widehat{U} = \{x_1, x_4, x_5, x_6, x_7\}$.
6. We will now use Proposition 1(2) to find proper u. Since we have $\mathsf{POS}^{\widehat{U}}_{\{a_1,a_2\}}(D) = \widehat{U}$ and $\mathsf{POS}^{\widehat{U}}_{\{a_2,a_4\}}(D) = \widehat{U}$, we set either $u = a_1$ or $u = a_4$.
7. Finally we set $(u, v) = (a_1, a_2)$ or $(u, v) = (a_4, a_2)$. ◇

3.3 A Local Search Algorithm with the Attribute Pair Selection Mechanism for Attribute Reduction

In step 4 of Algorithm 1, some element a is randomly removed from *Red*. Next we try to find appropriate u and v, but we may not succeed. In such a case a should not be used in next iteration. To implement this we use a set of *removed attributes* denoted by *RemoveSet* in Algorithm 2. Moreover at some point we will reach some local optimum so no more iteration is needed as we have just got our result. Local optimum means that we cannot remove any attribute a from the current reduct *Red*, all elements of *Red* have been tried but none has worked so they all have been put into *RemoveSet*, i.e. a local optimum is reached when *Red* = *RemoveRed*. Therefore we have designed the following four adjustment rules.

Adjustment rule 1: In each iteration, the randomly deleted attribute a must not belong to *RemoveSet*.

Adjustment rule 2: If a pair of attributes (u, v) cannot be found in the current iteration, the randomly deleted attribute a is added to the set *RemoveSet*.

Adjustment rule 3: *RemoveSet* is initialized to empty set. If a pair of attributes (u, v) is found, the search of current reduct is stopped, *RemoveSet* is reset to empty set again and the new iteration begins.

Adjustment rule 4: If the current attribute reduct *Red* equals *RemoveSet*, the algorithm stops and returns *Red*. Since *RemoveSet* $\subseteq$ *Red*, we can replace equality *Red* = *RemoveSet* with computationally simpler $|Red| = |RemoveSet|$.

Algorithm 2 applies all the above four rules and techniques described in Sect. 3.2. As opposed to Algorithm 1, it does not have an arbitrary limit of iterations T.

The analysis of its time complexity is similar to that for Algorithm 1. Algorithm 2 also starts with construction of a relative reduct using the same greedy procedure, so the worst case time complexity of this step (i.e. step 2) is $O(|Red|^2|U|) = O(|C|^2|U|)$.

Algorithm 2. (LSAR-APS) Local search algorithm with the attribute pair selection mechanism for attribute reduction

```
Input: A decision table $S = (U, C, D, V, f)$.
Output: The attribute reduction $Red$.
1  $t = 0$, $Red = \emptyset$ and $RemoveSet = \emptyset$;
2  construct a relative reduct $Red$ using greedy algorithm;    /* the same as in
     Algorithm 1 */
3  while $|Red| \neq |RemoveSet|$                    /* Adjustment rule 4 */ do
4      remove at random an attribute $a$ from $Red \setminus RemoveSet$;   /* Adjustment
         rule 1 */
5      if $\mathsf{POS}_{Red}(D) = \mathsf{POS}_{Red\setminus\{a\}}(D)$ then
6          $Red = Red \setminus \{a\}$;
7      else
8          calculate $C^*_{Red}$;
9          $flag = 0$;    /* the tag $flag$ is used to mark whether or not the
             attribute pair $(u, v)$ can be found */
10         for each $v \in C^*_{Red}$ and each $u \in Red \setminus \{a\}$ when $flag = 0$ do
11             compute $\mathsf{POS}_C(D)/\mathsf{ind}((Red \setminus \{a\}) \cup \{v\}) = \{X_1, \ldots, X_n\}$;
12             construct a set $\widehat{U} = \{x_1, \ldots, x_n\}$, where $x_i \in X_i$;
13             if $\mathsf{POS}^{\widehat{U}}_{(Red\setminus\{a,u\})\cup\{v\}}(D) = \widehat{U}$ then
14                 $Red = (Red \setminus \{u\}) \cup \{v\}$;
15                 $flag = 1$; /* $flag = 1$ means finding an attribute pair and
                     it causes exit from the loop, as by Adjustment rule 3
                     */
16             end
17         end
18         if $flag = 0$ then
19             $RemoveSet = RemoveSet \cup \{a\}$;        /* Adjustment rule 2 */
20         else
21             $RemoveSet = \emptyset$;                   /* Adjustment rule 3 */
22         end
23     end
24 end
25 return $Red$;
```

For the time essential steps inside the loop **while do** (step 3) we have the following worst case time complexities. Let Red_i represents the relative reduct used in the i^{th} iteration. Step 5 is $O(|Red_i||U|) = O(|C||U|)$. Time complexity of step 8, i.e. finding $C^*_{Red_i}$, is $O(|C \setminus Red_i||\mathsf{POS}_C(D) \setminus \mathsf{POS}_{Red_i}(D)|) = O(|C||U|)$. Steps 11–12 construct $\widehat{U}$ and their time complexity is $O(|Red_i||\mathsf{POS}_C(D)|) = O(|C||U|)$, while steps 13–16 verify if a pair (u, v) fixes Red_i, and they are $O(|Red_i||\mathsf{POS}_{Red_i}(D)) = O(|C||U|)$ as well. The remaining steps inside **while do** have complexity $O(1)$. Hence the entire worst case time complexity of the i^{th} iteration is $O(|C||U|)$, or more precisely $O(|Red_i||U|)$.

As far as the worst case time complexity is concerned, the i^{th} iteration of Algorithm 1 and the i^{th} iteration of Algorithm 2, have the same upper approximation $O(|Red_i||U|) = O(|C||U|)$. However, because $|\mathsf{POS}_C(D) \setminus \mathsf{POS}_{Red_i \setminus \{a\}}(D)| \ll |U$, $|\widehat{U} \leq |\mathsf{POS}_C(D)| \leq |U|$ and, usually, $|C^*_{Red_i}| \ll |Red_i|$, an average case time complexity of Algorithm 2 is usually much smaller than $O(|Red_i||U|)$ for the i^{th} iteration.

The loop **while do** executes $O(|Red|) = O(|C|)$ times, so the overall worst case time complexity of Algorithm 2 is $O(|C|^2|U|)$. In reality, Algorithm 2 (LSAR-APS) is usually much faster than Algorithm 1 (LSAR), however there might be some exceptions (for example see Table 4, data set CNAE-9).

4 Experiments

In this section, we will present the results of experiments conducted to evaluate the performance of Algorithms 1 and 2, also named as LSAR and LSAR-APS, on eight well-known UCI data sets [26]. The characteristics of these data sets are given in Table 2. We compare our two algorithms with the positive region-based heuristic algorithm POSR [13], the backward search strategy-based quick heuristic algorithm GARA-BS [16], and the immune quantum-behaved particle swarm attribute reduction algorithm IQPOSR [23]. All the experiments have been ran on a personal computer with Inter(R) Core(TM) i5-7300HQ CPU, 2.50 GHz and 16 GB memory. The programming language is Matlab R2016a.

Table 2. Description of data sets.

Data sets	Names	No. of objects	No. of attributes	No. of classes
S1	Soybean (small)	47	35	4
S2	Zoo	101	16	7
S3	Dermatology	366	33	6
S4	Mushroom	8124	22	2
S5	Letter	20000	16	26
S6	CNAE-9	1080	856	9
S7	Musk (Ver.2)	6598	166	2
S8	Connect-4	67557	42	3

4.1 Reduct Size and Computation Time

We evaluate the feasibility and effectiveness of our two algorithms according to two aspects: the reduct size and the computation time. The algorithms POSR, GARA-BS and LSAR-APS have no parameters. For IQPOSR, the parameters use the settings on small-scale problem instances in [23], and the specific settings are as follows: the particle size $M = 50$, the total number of iterations $T = 200$, the particle protection period $K = 10$, the accuracy error $\varepsilon_0 = 0.01$, and the test cost of each attribute $c(a) = 1$. LSAR is a single candidate solution-based stochastic local search algorithm, and it requires more iterations than population-based iterated algorithms. Hence the maximum iterations of LSAR is 10 times that of IQPOSR, i.e., $T = 2000$. However the time complexity of LSAR is much less than that of IQPOSR. Each algorithm runs 10 times on each data set, and we record the best reduct and the average computation time of the 10 runs. The experiment results shown in Tables 3 and 4.

Table 3. Comparison of reduct size on eight data sets

Data set	Reduct size				
	POSR	GARA-BS	IQPSOR	LSAR	LSAR-APS
Soybean (small)	2	2	2	2	2
Zoo	5	5	5	5	5
Dermatology	10	9	9	8	8
Mushroom	4	4	4	4	4
Letter	11	12	11	11	11
CNAE-9	81	75	84	80	71
Musk (Ver.2)	4	4	4	4	4
Connect-4	34	34	35	34	34

Table 3 shows that the reduct sizes obtained by LSAR and LSAR-APS are the same on all data sets, except for the data set CNAE-9. From all five algorithms, LSAR-APS is the best one in terms of the reduct size, especially for the data set CNAE-9. The reduct size of these five algorithms are the same on data sets Soybean (small), Zoo, Mushroom, and Musk (Ver.2). POSR obtains the worst reduct size on data sets Dermatology, and the reduct size of GARA-BS is the worst one on data set Letter. On data sets CNAE-9 and Connect-4, IQPSOR performs the worst in terms of the reduct size.

From Table 4 we have that GARA-BS is the fastest algorithm on data sets Soybean (small) and Zoo. On data sets Dermatology, Mushroom, Letter, Musk (Ver.2), and Connect-4, the algorithm LSAR-APS performs the best in terms of the computational time. On data set CNAE-9, the computational time of LSAR is the best one. *This is one of these rare cases when LSAR performed better than LSAR-APS.* The algorithm POSR is very complex, so its computational time grows dramatically as the data set increases. IQPSOR is a population-based meta-heuristic algorithm, and its computational times are stable. Among three previous algorithms, GARA-BS obtains the smallest computational time, but its computational time is still far greater than that of LSAR-APS.

In summary, especially when large data sets are concerned, our algorithm LSAR-APS can achieve a better reduct in a much shorter time. For example,

Table 4. Comparison of computational time on eight data sets

Data set	Computational time/s				
	POSR	GARA-BS	IQPSOR	LSAR	LSAR-APS
Soybean (small)	0.138	0.011	2.357	1.130	0.017
Zoo	0.184	0.014	3.387	1.339	0.033
Dermatology	3.186	0.132	12.751	2.817	0.105
Mushroom	19.773	1.022	324.006	16.930	0.617
Letter	245.631	3.813	737.317	72.026	2.239
CNAE-9	2064.218	220.204	718.453	23.013	74.643
Musk (Ver.2)	365.983	10.319	449.029	17.112	1.695
Connect-4	12417.113	175.689	2665.508	614.399	56.953

the algorithm LSAR-APS only takes an average of 74.643 s to find a reduct with a smallest size 71, and this is definitely the best results among these five algorithms. To the best of our knowledge, the reduct size 71 on data set CNAE-9 is also the best solution obtained so far.

4.2 Classification Accuracy Analysis

The classification accuracy was conducted on the selected attribute reducts found by all five algorithms with classifier 3NN (k-Nearest Neighbor algorithm and $k = 3$), which is a popular classifier for testing the attribute reduction algorithms. All of the classification accuracies are obtained with 10-fold cross validation. In 10-fold cross validation, a given data set is randomly divided into 10 nearly equally sized subsets, of these 10 subsets, 9 subsets are used as training set, a single subset is retained as testing set to assess the classification accuracy. The average performance results in terms of the classification accuracy are summarized in Table 5, where the column "Raw" depicts the classification accuracies with the original data and the boldface highlights the highest accuracy among these five algorithms.

Table 5. Classification accuracy on different data sets.

Data set	Classification accuracy/%					
	Raw	POSR	GARA-BS	IQPSOR	LSAR	LSAR-APS
S1	100.00±0.00	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**
S2	93.18 ±7.93	90.09±8.17	89.51 ±10.54	89.09±8.77	90.18±10.36	**91.00±11.01**
S3	96.72±2.50	**92.64±3.62**	73.75± 8.56	90.16 ±5.51	76.26± 7.66	76.52± 5.24
S4	100.00±0.00	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**
S5	95.63±0.41	**94.61±0.31**	94.23±0.55	93.68±0.30	93.38±0.51	94.36±0.50
S6	85.83±2.73	85.74±3.30	85.83±2.66	85.93±3.17	85.28±4.37	**86.11±2.99**
S7	96.79±0.58	90.85±1.29	91.60±0.59	**92.83±0.89**	91.54±0.99	91.71±1.05
S8	66.60±1.23	67.31±1.53	67.30±0.72	67.03±0.86	67.21±0.98	**67.68±0.86**

Table 5 shows that the algorithm LSAR-APS achieves the best classification performance as its number of the highest classification accuracy is five times out of eight data sets. For POSR this number is four times among eight data sets, QIPSOR matched the best classification accuracies for 3 out of 8 cases while and LSAR and GARA-BS only obtain the best classification performance on data sets S1 and S4. Hence, LSAR-APS can achieve better or comparable classification accuracy in comparison with other four algorithms.

5 Conclusion

In this paper, we studied local search approach for attribute reduction problem in rough set theory that has a wide range of applications. We introduced a local search framework for this problem and proposed two advanced strategies to improve the iteration process of the local search-based algorithm, i.e., attribute pair selection mechanism and adjustment rules. The results of the experiment on the broadly used data set indicated that our proposed algorithm LSAR-ASP significantly outperforms other state-of-the-art algorithms.

We are surprised to find that the reduct found by LSAR-APS on data set CNAE-9 is actually an optimal reduct (see Appendix A). In this sense, this work provides a new idea for solving the optimal reduct of large data sets. In the future work, we will test our proposed algorithm on high-dimensional large data sets and propose some additional improved strategies to enhance the efficiency of the local search-based attribute reduction algorithm.

Acknowledgment. The authors gratefully acknowledge three anonymous referees for their helpful comments. The research was supported by The National Natural Science Foundation of China (grant nos. 61373015, 61728204), China Scholarship Council (grant no. 201806830058), State Key Laboratory for smart grid protection and operation control Foundation, Science and Technology Funds from National State Grid Ltd(The Research on Key Technologies of Distributed Parallel Database Storage and Processing based on Big Data), and NSERC of Canada (Discovery grant no. 6466-15).

Appendix A

Here we report the optimal solution found by LSAR-APS on data set CNAE-9. The optimal reduct is: 7 20 63 68 73 75 77 105 118 119 133 150 151 183 191 194 199 201 202 207 211 246 247 258 272 276 328 333 334 338 345 350 359 360 373 382 390 403 415 417 421 423 424 443 476 483 499 518 519 539 546 555 581 607 608 614 615 618 619 631 648 650 673 684 705 726 731 815 823 824 832.

References

1. Pawlak, Z.: Rough sets. Int. J. Comput. Inf. Sci. **11**(5), 341–356 (1982)
2. Swiniarski, R.W., Skowron, A.: Rough set methods in feature selection and recognition. Pattern Recogn. Lett. **24**(6), 833–849 (2003)
3. Lingras, P.J., Yao, Y.Y.: Data mining using extensions of the rough set model. J. Am. Soc. Inf. Sci. **49**(5), 415–422 (1998)
4. Herawan, T., Deris, M.M., Abawajy, J.H.: A rough set approach for selecting clustering attribute. Knowl. Based Syst. **23**(3), 220–231 (2010)
5. Janicki, R., Lenarčič, A.: Optimal approximations with rough sets and similarities in measure spaces. Int. J. Approximate Reasoning **71**, 1–14 (2016)
6. Janicki, R.: Approximations of arbitrary relations by partial orders. Int. J. Approximate Reasoning **98**, 177–195 (2018)
7. Xie, X., Qin, X.: Dynamic feature selection algorithm based on minimum vertex cover of hypergraph. In: Phung, D., Tseng, V.S., Webb, G.I., Ho, B., Ganji, M., Rashidi, L. (eds.) PAKDD 2018. LNCS (LNAI), vol. 10939, pp. 40–51. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93040-4_4
8. Skowron, A., Rauszer, C.: The discernibility matrices and functions in information systems. In: Słowiński, R. (ed.) Intelligent Decision Support. Handbook of Applications and Advances of the Rough Sets Theory, Dordrecht, Kluwer (1992)
9. Nguyen, H.S.: Approximate boolean reasoning approach to rough sets and data mining. In: Ślęzak, D., Yao, J.T., Peters, J.F., Ziarko, W., Hu, X. (eds.) RSFDGrC 2005. LNCS (LNAI), vol. 3642, pp. 12–22. Springer, Heidelberg (2005). https://doi.org/10.1007/11548706_2
10. Tan, A., Li, J., Lin, Y., Lin, G.: Matrix-based set approximations and reductions in covering decision information systems. Int. J. Approximate Reasoning **59**, 68–80 (2015)
11. Hacibeyoglu, M., Salman, M.S., Selek, M., Kahramanli, S.: The logic transformations for reducing the complexity of the discernibility function-based attribute reduction problem. Knowl. Inf. Syst. **46**(3), 599–628 (2016)
12. Yang, T., Li, Q., Zhou, B.: Related family: a new method for attribute reduction of covering information systems. Inf. Sci. **228**, 175–191 (2013)
13. Xu, Z., Liu, Z., Yang, B.: A quick attribute reduction algorithm with complexity of $\max(O(|C||U|), O(|C|^2|U/C|))$. Chin. J. Comput. **29**(3), 391–399 (2006)
14. Jiang, F., Sha-sha, W., Du, J.W., Yue-Fei, S.: Attribute reduction based on approximation decision entropy. Control Decis. **30**(1), 65–70 (2015)
15. Deng, T., Yang, C., Hu, Q.: Feature selection in decision systems based on conditional knowledge granularity. Int. J. Comput. Intell. Syst. **4**(4), 655–671 (2011)
16. Ge, H., Li, L., Xu, Y., Yang, C.: Quick general reduction algorithms for inconsistent decision tables. Int. J. Approximate Reasoning **82**, 56–80 (2017)
17. Xie, X., Qin, X.: A novel incremental attribute reduction approach for dynamic incomplete decision systems. Int. J. Approximate Reasoning **93**, 443–462 (2018)
18. Xu, Z., Gu, D., Yang, B.: Attribute reduction algorithm based on genetic algorithm. In: Proceedings of International Conference on Intelligent Computation Technology and Automation, Zhangjiajie, China, pp. 169–172 (2009)
19. Chen, Y., Zhu, Q., Xu, H.: Finding rough set reducts with fish swarm algorithm. Knowl. Based Syst. **81**, 22–29 (2015)
20. Inbarani, H.H., Bagyamathi, M., Azar, A.T.: A novel hybrid feature selection method based on rough set and improved harmony search. Neural Comput. Appl. **26**(8), 1859–1880 (2015)

21. Luan, X.Y., Li, Z.P., Liu, T.Z.: A novel attribute reduction algorithm based on rough set and improved artificial fish swarm algorithm. Neurocomputing **174**, 522–529 (2016)
22. Abd El Aziz, M., Hassanien, A.E.: An improved social spider optimization algorithm based on rough sets for solving minimum number attribute reduction problem. Neural Comput. Appl. **30**(8), 2441–2452 (2018)
23. Xie, X., Qin, X., Yu, C., Xu, X.: Test-cost-sensitive rough set based approach for minimum weight vertex cover problem. Appl. Soft Comput. **64**, 423–435 (2018)
24. Cai, S., Su, K., Sattar, A.: Local search with edge weighting and configuration checking heuristics for minimum vertex cover. Artif. Intell. **175**(9), 1672–1696 (2011)
25. Cai, S., Hou, W., Lin, J., Li, Y.: Improving local search for minimum weight vertex cover by dynamic strategies. In: Proceedings of International Joint Conferences on Artificial Intelligence, Stockholm, Sweden, pp. 1412–1418 (2018)
26. UCI machine learning repository. http://www.ics.uci.edu/mlearn/MLRepository.html