Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition Letters 29 (2008) 1351–1357
Pattern Recognition Letters
www.elsevier.com/locate/patrec

An efficient ant colony optimization approach to attribute reduction in rough set theory

Liangjun Ke *, Zuren Feng, Zhigang Ren

State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an 710049, China

Received 11 September 2006; received in revised form 6 December 2007
Available online 4 March 2008

Communicated by S.K. Pal

Abstract

Attribute reduction in rough set theory is an important feature selection method. Since attribute reduction is an NP-hard problem, it is necessary to investigate fast and effective approximate algorithms. In this paper, we introduce a new approach based on ant colony optimization (ACO) for attribute reduction. To verify the proposed algorithm, numerical experiments are carried out on thirteen small or medium-sized datasets and three gene expression datasets. The results demonstrate that this algorithm can provide competitive solutions efficiently.

Keywords: Ant colony optimization; Attribute reduction; Feature selection; Rough set theory

1. Introduction

With the proliferation of high-dimensional data, feature selection has become an indispensable task of a learning process. Feature selection studies how to select a subset of features from the original set of features while retaining a suitably high accuracy in representing the original features. By removing irrelevant and redundant features, feature selection helps to improve the quality and speed of learning algorithms and to enhance the comprehensibility of the constructed models (Theodoridis and Koutroumbas, 2006).

In real applications, data processing often involves vague or imprecise attributes or features. Since rough set theory (Pawlak, 1982, 1991) is a valid mathematic tool to handle imprecision, uncertainty and vagueness, attribute reduction in rough set theory has been recognized as an important feature selection method (Pawlak, 1996; Skowron and Pal, 2003; Swiniarski and Skowron, 2003; Pawlak and Skowron, 2007). It is well-known that attribute reduction is an NP-hard problem (Wong and Ziarko, 1985). Due to the fact that hill-climbing algorithms often fail to find optimal solutions (Jensen and Shen, 2004), many research efforts have shifted to metaheuristics, such as genetic algorithm (GA) (Wróblewski, 1995; Bazan et al., 2000; Jensen and Shen, 2004), simulated annealing (SA), ant colony optimization (ACO) (Jensen and Shen, 2004), tabu search (TS) (Hedar et al., 2006) and more recently particle swarm optimization (PSO) (Wang et al., 2007). These algorithms can often obtain high quality solutions. However, even for a medium-sized problem, thousands of seconds may be required to deal with it (Jensen and Shen, 2004; Wang et al., 2007).

In this paper, a novel ACO-based algorithm, called ant colony optimization for attribute reduction (ACOAR), is developed. The basic idea of ACO is as follows (Dorigo et al., 1996). Guided by pheromone trails and problem-dependent heuristic information, a colony of agents, called (artificial) *ants*, search in the solution space of a problem. Based on the ants' experience obtained previously, ACO tries to update pheromone trails so as to make the probability of constructing high quality solutions increase eventually (Dorigo and Gambardella, 1997). Its usefulness

* Corresponding author.
E-mail address: kelj163@163.com (L. Ke).

0167-8655/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2008.02.006

and versatility have amply been demonstrated by successful applications in a variety of hard combinatorial optimization problems (Dorigo and Di Caro, 1999; Dorigo et al., 1999; Dorigo and Stützle, 2004; Dorigo and Blum, 2005; Zlochin et al., 2004). It is noteworthy that an ACO-based algorithm (Jensen and Shen, 2003), which is called AntRSAR, has been proposed for attribute reduction. The experimental results have shown that AntRSAR is a promising approach. Nevertheless, AntRSAR is more time-consuming than other metaheuristics including GA and SA (Jensen and Shen, 2004). Although the standard ACO provides the basic underlying mechanism to deal with attribute reduction, it is useful to analyze the characteristics of attribute reduction. ACOAR extends the basic idea of max-min ant system (MMAS) (Stützle and Hoos, 2000) which is one of the most successful ACO variants (Dorigo and Blum, 2005). In addition, an efficient process is proposed to construct candidate solutions. The objective of this paper is to study ACOAR in terms of solution quality and computational effort.

The remainder is organized as follows. In Section 2, we present the basic concepts of attribute reduction in rough set theory. In Section 3, the new algorithm is given. Section 4 presents the experimental results. The conclusion and future work are included in Section 5.

2. Preliminaries

Let $I = (U, A)$ be an information system, where U, called universe, is a nonempty set of finite objects; A is a nonempty finite set of attributes such that a: $U \to V_a$ for every $a \in A$; V_a is the value set of a. In a decision system, $A = C \cup D$ where C is the set of condition attributes and D is the set of decision attributes.

For an attribute set $P \subseteq A$, there is an associated indiscernibility relation IND(P):

$$\mathrm{IND}(P) = \{(x, y) \in U^2 | \forall a \in P,\ a(x) = a(y)\} \tag{1}$$

If $(x, y) \in \mathrm{IND}(P)$, then x and y are indiscernible by attributes from P. The family of all equivalence classes of IND(P), i.e., the partition determined by P, is denoted as U/P. An equivalence class of IND(P), i.e., the block of the partition U/P, containing x is denoted by $[x]_P$. The indiscernibility relation is the mathematical basis of rough set theory.

In rough set theory, the lower and upper approximations are two basic operations. Given an arbitrary set $X \subseteq U$, the P-lower approximation of X, denoted as $\underline{P}X$, is the set of all elements of U, which can be certainly classified as elements of X based on the attribute set P. The P-upper approximation of X, denoted as $\overline{P}X$, is the set of all elements of U, which can be possibly classified as elements of X based on the attribute set P. These two definitions can be expressed as:

$$\underline{P}X = \{x | [x]_P \subseteq X\} \tag{2}$$

$$\underline{P}X = \{x | [x]_P \cap X \neq \varnothing\} \tag{3}$$

Definition 1 (*Dependency degree*). Let P, $Q \subseteq A$, the dependency degree κ is defined by

$$\kappa = r_P(Q) = |\mathrm{POS}_P(Q)|/|U| \tag{4}$$

where $|Y|$ is the cardinality of set Y. $\mathrm{POS}_P(Q)$, called positive region, is defined by

$$\mathrm{POS}_P(Q) = \bigcup_{X \in U/Q} \underline{P}X \tag{5}$$

The positive region contains all objects in U that can be uniquely classified to blocks of the partition U/Q by means of the knowledge in attributes P.

The quantity κ can be used to measure the degree of dependency between Q and P. If $\kappa = 1$, Q depends totally on P which means that all attribute values from Q are uniquely determined by values of attributes from P. If $0 < \kappa < 1$, Q depends on P in a degree κ. If $\kappa = 0$, then Q does not depend on P.

Attribute reduction in rough set theory provides a filter-based tool to extract knowledge from a domain in a concise way. It can preserve the information content while reducing the amount of attributes involved. Based on dependency degree, a reduct can be defined by the following definition.

Definition 2 (*Reduct*). Let R be a subset of C, then R is said to be a reduct if

$$r_R(D) = r_C(D) \wedge \forall R' \subset R,\ r_{R'}(D) < r_C(D) \tag{6}$$

Specially, a reduct with minimal cardinality is called *minimal reduct*. The goal of attribute reduction is to find a minimal reduct. Its objective function is

$$\min_{R \in \mathbb{R}} |R| \tag{7}$$

where $\mathbb{R}$ is the set which consists of all reducts of C.

3. Ant colony optimization for attribute reduction

ACOAR, the proposed algorithm for attribute reduction, follows the standard ACO algorithmic scheme for static combinatorial optimization problems. However, it has many new features which will be explained in this section. The algorithm performs as follows: at each cycle, every ant constructs a solution and then pheromone trails are updated. The algorithm stops iterating when a termination condition is met. Generally, the termination condition may be a maximum number of cycles or a given time limit. In the following, we first discuss the definition of pheromone trails and heuristic information. Then we describe how to construct a candidate solution. Finally, the pheromone updating step is presented. The pseudo-code of ACOAR is given in the Appendix.

3.1. Pheromone trails and heuristic information

Attribute reduction can be described by a complete graph $G = (V, E)$ with V being the set of vertices, which

represent condition attributes, and E being the set of edges fully connecting the vertices. Attribute reduction then is modeled as the problem of finding a path such that the set of selected attributes is a reduct and the cardinality of this set is minimized. Each edge is assigned a pheromone trail and heuristic information. The heuristic information we used is on the basis of dependency degree. In practice, it is necessary to limit the value of heuristic information, otherwise the algorithm may be too greedy. Formally, for $\forall a, b \in C$, the heuristic information is given as follows:

$$\eta(a,b) = |\mathrm{POS}_{\{a,b\}}(D)|/|U| \tag{8}$$

$$\text{If } \eta(a,b) < \varepsilon, \text{ then } \eta(a,b) = \varepsilon \tag{9}$$

where ε ($0 < \varepsilon < 1$) is a small positive parameter.

3.2. Constructing a solution

While constructing a solution, each ant starts from a randomly selected attribute, then it probabilistically selects next attribute from those unselected attributes. The probability is given by (Dorigo et al., 1996)

$$p(c_{k+1} = v|\tau, c_k = u) = \begin{cases} \frac{\tau(u,v)^{\alpha}\cdot\eta(u,v)^{\beta}}{\sum_{w\in C_u}\tau(u,w)^{\alpha}\cdot\eta(u,w)^{\beta}}, & \text{if } v \in C_u \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where c_k denotes the attribute selected at the kth construction step, C_u is the set of condition attributes that have not yet been selected, $\tau(u,w)$and $\eta(u,w)$ are the pheromone value and heuristic information of edge (u,w), respectively. α and β are two parameters which control the relative importance of pheromone trails and heuristic information. If α is far larger than β, then ants will make decisions mainly based on pheromone trails, and if β is far larger than α, ants will select those edges with higher heuristic information in a greedy manner. Based on preliminary experimental results, with larger values of β, it often takes ants longer time to reach a high quality solution, whereas ants can find better solutions when their search behavior is mainly influenced by pheromone trails.

The construction process stops when one of the following conditions is met:

(C_1) The cardinality of the solution is larger than the smallest cardinality obtained so far.
(C_2) $r_R(D) = r_C(D)$, where R is a solution constructed by an ant.

The first condition means that no better solution will be constructed, thus it is unnecessary to continue the construction process. While the second implies that a better solution has been constructed, thus the construction process can be terminated.

Fig. 1 illustrates a typical construction process, where C is $\{a_1, a_2, a_3, a_4, a_5, a_6\}$. At the first step, attribute a_1 is chosen in a random manner, then attributes a_3 and a_6 are selected. The last selected attribute is a_4.

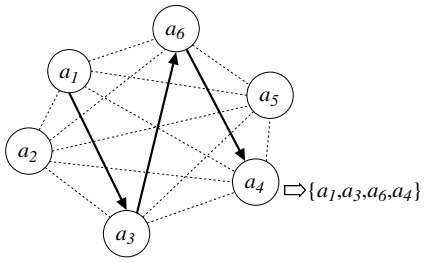

Fig. 1. An illustrative process of constructing a solution, where the condition attribute set C is $\{a_1, a_2, a_3, a_4, a_5, a_6\}$ and the constructed solution is $\{a_1, a_3, a_6, a_4\}$.

During constructing a solution, it is necessary to calculate dependency degree after adding a new attribute. Since this calculating process is time-consuming (Nguyen and Nguyen, 1996), a more efficient calculating method will accelerate ACOAR. In order to calculate this quantity, the first step is to determine new equivalence classes, then to determine whether all objects in each equivalence class have the same decision values (Nguyen and Nguyen, 1996). In practice, for a given equivalence class, if its objects have the same decision values, there is no need to partition it again after a new attribute is added. Therefore, the dependency degree can be calculated faster.

3.3. Pheromone update

After each ant has constructed a solution, pheromone trails are updated mainly according to MMAS. In ACOAR, only the ant which has constructed the best-so-far solution is selected to update pheromone trails. Suppose the best-so-far solution is $s_{gb} = \{a_{i_1}, \ldots, a_{i_j}\}$. Conventionally, only edges along the path visited by the ant, i.e., the edges connecting every two successive attributes of s_{gb} are updated. Formally, pheromone trails are updated by the following rule:

Pheromone_Update_Rule_1 :

for $k = 1$ *to* $j - 1$ *do*

$l = k + 1$

$\tau(a_{i_k}, a_{i_l}) = \rho\tau(a_{i_k}, a_{i_l}) + q/L_{gb}$

$\tau(a_{i_l}, a_{i_k}) = \tau(a_{i_k}, a_{i_l})$

where q is a parameter, $L_{gb} = |s_{gb}|$. Note that AntRSAR updates the edges connecting every two successive attributes of the chosen solution, denoted as $\tilde{s}$, and besides, the updated values of pheromone trails depend on cardinality and goodness, i.e., $r_{\tilde{s}}(D)$. However, since $r_{s_{gb}}(D)$ is a constant ($r_{s_{gb}}(D) = r_c(D)$), the goodness of s_{gb} can be neglected in ACOAR.

As mentioned in Section 3.1, each solution of attribute reduction is an attribute subset. Although the solution construction process is ordered, the elements in the set are unordered. For example, in Fig. 1, though the solution is

constructed by the order from a_1 to a_3, then to a_6 and a_4, every permutation of $\{a_1, a_3, a_6, a_4\}$ is also a solution. Therefore, it is reasonable to reinforce the pheromone trails of the edges connecting every two different vertices. In contrast to *Pheromone_Update_Rule_1*, this rule can make the probability of searching around the best-so-far solution higher. Moreover, this rule reinforces more edges, thus it is more possible to explore new solutions.

In ACOAR, the pheromone trails are updated by the following rule:

Pheromone_Update_Rule_2 :

for $k = 1$ *to* $j - 1$ *do*

for $l = k + 1$ *to* j *do*

$$\tau(a_{i_k}, a_{i_l}) = \rho\tau(a_{i_k}, a_{i_l}) + q/L_{gb}$$

$$\tau(a_{i_l}, a_{i_k}) = \tau(a_{i_k}, a_{i_l})$$

While for other edges, the pheromone trails are evaporated by

$$\tau(a_u, a_v) = \rho\tau(a_u, a_v) \tag{11}$$

$$\tau(a_v, a_u) = \tau(a_u, a_v) \tag{12}$$

where a_u, $a_v \in C$. It is known that search stagnation may occur if the relative differences of pheromone trails are too extreme (Stützle and Hoos, 2000). To avoid this situation, the pheromone trails are limited between the upper and lower limits as follows:

$$\text{if } \tau(a_u, a_v) > \tau_{\max}, \text{ then } \tau(a_u, a_v) = \tau_{\max} \tag{13}$$

$$\text{if } \tau(a_u, a_v) < \tau_{\min}, \text{ then } \tau(a_u, a_v) = \tau_{\min} \tag{14}$$

where $\tau_{\max}$ and $\tau_{\min}$ are the upper and lower trail limits, respectively.

4. Experimental study

We conclude this paper by presenting numerical results using ACOAR on various datasets, with different numbers of attributes and objects. We implemented ACOAR in Java and run it on a 1.7 GHz CPU PC with 1 GB of RAM. In the experiments, the parameters, except when indicated differently, were set to the following values: $\alpha = 1, \beta = 0.1$, and the initial pheromone value was set to 0.5 with a small random perturbation added, $q = 0.1$, $\varepsilon = 0.01$, $\rho = 0.9$, $\tau_{\max} = 1$, $\tau_{\min} = 0.001$, the number of ants was 10 and the maximum number of cycles was 100. These parameters were determined based on a small number of preliminary runs.

4.1. Experimental results for small or medium-sized datasets

Thirteen datasets were taken into account. Three of them, namely, *m-of-n*, *exactly*, and *exactly2*, are from (Raman and Ioerger, 2002), and the others are from UCI data (Blake and Merz, 1998). As suggested by Jensen and Shen (2004), each dataset was tested 20 times.

First, we compared ACOAR with two ACO-based algorithms, i.e., AntRSAR (Jensen and Shen, 2004) and ACOAR-1 which uses *Pheromone_Update_Rule_1* to update pheromone trails. Note that AntRSAR was terminated after 250 iterations and the number of ants was set to $|C|$. The results are given in Table 1. The 'attributes' column indicates the number of attributes in each dataset. For each algorithm, the smallest cardinality obtained at every run is given, and in parentheses is the number of runs that this cardinality is achieved.

According to the experimental results, it can be seen that ACOAR outperforms ACOAR-1 and AntRSAR on all datasets, whereas ACOAR-1 works better than AntRSAR in 8 out of 13 datasets. Since the runtime of each algorithm is mainly determined by calculating dependency degree, the number of solution constructions is suitable as a main quantity to evaluate the speed of these algorithms. Note that ACOAR constructs fewer solutions than AntRSAR. Thus, ACOAR can rapidly converge to the best solutions. This statement was also confirmed by the results in Table 2, where the average iterations and average runtimes to

Table 1
Comparison of AntRSAR, ACOAR-1 and ACOAR

Dataset	Attributes	AntRSAR	ACOAR-1	ACOAR
M-of-n	13	6	6	6
Exactly	13	6	6	6
Exactly2	13	10	10	10
Heart	13	$6^{(18)}7^{(2)}$	6	6
Vote	16	8	8	8
Credit	20	$8^{(12)}9^{(4)}10^{(4)}$	$8^{(10)}9^{(8)}10^{(2)}$	$8^{(16)}9^{(4)}$
Mushroom	22	4	4	4
Led	24	$5^{(12)}6^{(4)}7^{(3)}$	$5^{(18)}6^{(2)}$	5
Letters	25	8	$8^{(17)}9^{(3)}$	8
Derm	34	$6^{(17)}7^{(3)}$	$6^{(12)}7^{(8)}$	6
Derm2	34	$8^{(3)}9^{(17)}$	$8^{(2)}9^{(9)}10^{(9)}$	$8^{(4)}9^{(16)}$
Wq	38	$12^{(2)}13^{(7)}14^{(11)}$	$13^{(3)}14^{(9)}15^{(8)}$	$12^{(4)}13^{(12)}14^{(4)}$
Lung	56	4	4	4

For each algorithm, the smallest cardinality obtained at every run is given. In parentheses is the number of runs that this cardinality is achieved.

Table 2
Average iterations and average runtimes to obtain the minimal reducts by ACOAR (over 20 trials)

Dataset	Attributes	Average iteration[a]	Average runtime (s)
M-of-n	13	22	0.31
Exactly	13	23	0.38
Exactly2	13	16	0.34
Heart	13	30	0.21
Vote	16	32	0.23
Credit	20	72	1.78
Mushroom	22	34	2.01
Led	24	37	1.37
Letters	25	48	0.21
Derm	34	57	0.80
Derm2	34	478	5.72
Wq	38	953	20.26
Lung	56	31	0.17

[a] The average iteration is rounded off.

Table 3
Comparison of ACOAR, GenRSAR, SimRSAR and TSAR

Dataset	Attributes	ACOAR	GenRSAR	SimRSAR	TSAR
M-of-n	13	6	$6^{(6)}7^{(12)}$	6	6
Exactly	13	6	$6^{(10)}7^{(10)}$	6	6
Exactly2	13	10	$10^{(9)}11^{(11)}$	10	10
Heart	13	6	$6^{(18)}7^{(2)}$	$6^{(29)}7^{(1)}$	6
Vote	16	8	$8^{(2)}9^{(18)}$	$8^{(15)}9^{(15)}$	8
Credit	20	$8^{(16)}9^{(4)}$	$10^{(6)}11^{(14)}$	$8^{(18)}9^{(1)}11^{(1)}$	$8^{(13)}9^{(5)}10^{(2)}$
Mushroom	22	4	$5^{(1)}6^{(5)}7^{(14)}$	4	$4^{(17)}5^{(3)}$
Led	24	5	$6^{(1)}7^{(3)}8^{(16)}$	5	5
Letters	25	8	$8^{(8)}9^{(12)}$	8	$8^{(17)}9^{(3)}$
Derm	34	6	$10^{(6)}11^{(14)}$	$6^{(12)}7^{(8)}$	$6^{(14)}7^{(6)}$
Derm2	34	$8^{(4)}9^{(16)}$	$10^{(4)}11^{(16)}$	$8^{(3)}9^{(7)}$	$8^{(2)}9^{(14)}10^{(4)}$
Wq	38	$12^{(4)}13^{(12)}14^{(4)}$	16	$13^{(16)}14^{(4)}$	$12^{(1)}13^{(13)}14^{(6)}$
Lung	56	4	$6^{(8)}7^{(12)}$	$4^{(7)}5^{(12)}6^{(1)}$	$4^{(6)}5^{(13)}6^{(1)}$

For each algorithm, the smallest cardinality obtained at every run is given. In parentheses is the number of runs that this cardinality is achieved.

obtain the minimal reducts are given. For each dataset except *Derm2* and *Wq*, all the minimal reducts can be found in less than 72 iterations. Even for *Derm2* and *Wq*, ACOAR can find the minimal reducts within 21s.

We then compared ACOAR with other metaheuristics: SimRSAR (SA-based), GenRSAR (GA-based) (Jensen and Shen, 2004) and TSAR (TS-based) (Hedar et al., 2006). The results are listed in Table 3. It can be seen that ACOAR provides the best results, the performance of GenRSAR is the worst, and TSAR is slightly better than SimRSAR. With regard to the number of solutions generated, the rough ordering of techniques is: ACOAR < TSAR < SimRSAR < GenRSAR.

As the experimental results show, ACOAR is one of the best performing approaches for attribute reduction. Basically, ACOAR is a constructive method (a survey on metaheuristics is available in Colorni et al., 1996). At each construction step, each ant prefers choosing more desirable attributes. As a result, it is more likely to construct high quality solutions. Moreover, it uses a simple mechanism to balance exploration and exploitation. It updates the pheromone trails of the edges connecting every two different attributes of the best-so-far solution so as to explore in the neighborhood of this solution, and avoids premature convergence by limiting pheromone values. On the other hand, ACOAR is able to construct a candidate solution rapidly. It uses an improved procedure (Nguyen and Nguyen, 1996) to calculate dependency degree, and calculates dependency degree as few as possible due to conditions C_1 and C_2.

4.2. Experimental results for high-dimensional datasets

We tested ACOAR on three gene expression datasets: the colon cancer dataset, the lymphoma dataset and the leukemia dataset. The first consists of 62 samples and 2000 attributes (genes). The goal is to discriminate between cancerous and normal tissues in a colon cancer problem (Alon et al., 1999). The dataset was randomly split into a training set of 50 samples and a test set of 12. The random split was performed 1000 times. The second contains 96 samples and 4026 attributes. The task is to separate cancerous and normal tissues in a large B-Cell lymphoma problem (Alizadeh et al., 2000). We randomly split the dataset into a training set of 60 samples and a test set of 36. The random split was performed 1000 times. The last contains 72 samples and 7129 attributes. The goal is to distinguish between two classes of leukemia ALL and AML (Golub et al., 1999; Guyon et al., 2002; Weston et al., 2003; Xing et al., 2001). The first 38 samples are used as training set and the remaining 34 as test set. Note that these datasets were split according to the common way in the literature (Gentile, 2003). So far several algorithms, such as approximate large margin algorithm (Gentile, 2003), neural networks (Huang, 2004) and multi-objective genetic algorithm (Banerjee et al., 2007), have been developed to deal with gene expression datasets.

When ACOAR is applied to a high-dimensional problem, the ants use a candidate list of length 100 which contains the best unselected attributes ordered according to decreasing dependency degrees. At each construction step, one attribute in this list is selected and then the list is updated. Candidate list is also popularly used in other applications (see e.g., Stützle and Hoos, 2000). In the experiments, each attribute was discretized into three intervals by the equal-width method. LEM2 algorithm (Stefanowski, 1998) was applied to induce decision rules and voting strategy was used for rule negotiation. The number of ants was 100. Each split was tested once except that the leukemia dataset was test 100 times. The computational results were compared with the results of approximate large margin algorithm for feature selection (ALMA-FS) (Gentile, 2003), since this algorithm has excellent performance and can automatically determine the final number of genes.

Table 4 shows a summary of the results obtained for these datasets. In contrast to the total numbers of attributes, the average cardinalities of the reducts obtained by ACOAR are very small. For example, the leukemia dataset has 7129 attributes while the average number of selected attributes is only 5. With regard to the computation time, the leukemia dataset is the largest dataset and requires the longest time for ACOAR (200s on average). Nevertheless, the classification accuracy of our results is inferior to ALMA-FS.

Table 4
Experimental results of ALMA-FS and ACOAR

Dataset	Attributes	ALMA-FS		ACOAR			
		Avg. num.[a]	Error (%)	Avg. num.	Error (%)	Rules[b]	Time (s)
Colon cancer	2000	23	12.7	8	40.5	13	44.71
Lymphoma	4026	31	8.1	7	35.7	12	73.92
Leukemia	7129	27	3.0	5	44.9	9	200.46

[a] The average number (avg. num.) of the selected attributes is rounded off.
[b] The number of rules is rounded off.

5. Conclusion and future work

In this paper, we propose an ACO-based algorithm, called ACOAR, to deal with attribute reduction in rough set theory. This algorithm has the following features: (a) it updates the pheromone trails of the edges connecting every two different attributes of the best-so-far solution; (b) pheromone values are limited between the upper and lower trail limits; (c) it uses a rapid procedure to construct candidate solutions. Due to its pheromone update rule and solution construction process, ACOAR has the ability to find solutions with very small cardinality rapidly.

Nevertheless, the accuracy results we obtained are less satisfactory than those excellent ones reported in the literature. This prompts us to extend this approach for finding other kinds of reducts such as approximate entropy reducts (Slezak and Wroblewski, 2003). Additionally, research on multi-objective optimization approaches may be a very promising direction (Banerjee et al., 2007). Moreover, since the classical rough set theory is unable to effectively deal with real-valued attributes, it is interesting to research into the use of fuzzy-rough set theory and ACO for attribute reduction (Jensen and Shen, 2002, 2004). As for large datasets, further research should be carried out to cut down memory consumption and alleviate computing load.

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments and suggestions. This work is supported by the National Natural Science Foundation of China, No. 60475023 and the Ph.D. Programs Foundation of Ministry of Education of China, No. 20050698032.

Appendix

Algorithm 1 (ACOAR).

INPUT: An information system $I=(C\cup D,U)$ and parameters.
OUTPUT: minimal reduct R^* and its cardinality L_{gb}.
1. initiate the pheromone trials τ and calculate heuristic information η and $r_C(D)$; $L_{gb}=|C|$.
2. while (the terminated condition is not reached) do
   $R_{\text{save}}:=\varnothing$; $L:=L_{gb}$
   for (each ant) do
      construct a solution by Algorithm 2, then a reduct R and its order l are obtained;
      if ($l<L$) then $R_{\text{save}}=R$ and $L=l$;
   update pheromone trails by Algorithm 4;
3. output R^* and L_{gb}.

Algorithm 2 (Constructing a solution).

INPUT: L_{gb}.
OUTPUT: a reduct R and its cardinality l.
1. select the first attribute $b_1\in C$ randomly, $R:=\{b_1\}$, $l:=1$;
2. if $r_R(D)=r_C(D)$ return R and l;
3. else repeat
      select next attribute $b_{\text{next}}\in C$ by formula (10), $l:=l+1$, $R:=R\cup\{b_{\text{next}}\}$;
      calculate $r_R(D)$ by Algorithm 3;
   until ($r_R(D)=r_C(D)$) or ($l=L_{gb}$);
4. output R and l.

Algorithm 3 (*Calculating* $r_R(D)$).

INPUT: equivalence classes $\{X_1,X_2,\ldots,X_k\}$; a vector V, if all objects in X_i have the same decision values, $V(i)=1$,else $V(i)=0$; the selected attribute a.
OUTPUT: $r_R(D)$, equivalence classes Y and vector W.
1. sum := 0; $Y:=\emptyset$; index := 0;
2. for $i=1$ to k do
      if ($V(i)=1$) then sum := sum + $|X_i|$, index := index + 1, $Y:=Y\cup X_i$, W(index) := 1;
      else
         determine the equivalence classes of X_i by a, suppose that they are $X_i^1,\ldots,X_i^t$
         for $j=1$ to t do
            index := index + 1; $Y=Y\cup X_i^j$;
            if (all objects in X_i^j have the same decision values)
               then W(index) := 1, sum = sum + $|X_i^j|$;
            else W(index) := 0;
3. output $r_R(D)=\text{sum}/|U|$, Y and W.

Algorithm 4 (Pheromone update).

INPUT: L, L_{gb}, R^*, R_{save}, τ.
OUTPUT: τ, L_{gb}.
1. if ($L < L_{gb}$) then $R^* := R_{\text{save}}$, $L_{gb} := L$.
2. for (every a, $b \in R^*$) do
   $\tau(a,b) := \tau(a,b) + q/\rho L_{gb}$;
3. for (every u, $v \in C$) do
   $\tau(u,v) := \rho\tau(u,v)$;
   if ($\tau(u,v) > \tau_{\max}$), then $\tau(u,v) := \tau_{\max}$;
   if ($\tau(u,v) < \tau_{\min}$), then $\tau(u,v) := \tau_{\min}$;
4. output t and L_{gb}.

References

Alizadeh, A. et al., 2000. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. Nature 403, 503–511.

Alon, U. et al., 1999. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon cancer tissues probed by oligonucleotide arrays. Proc. Natl. Acad. Sci. USA 96, 6745–6750.

Banerjee, M., Mitra, S., Banka, H., 2007. Evolutionary rough feature selection in gene expression data. IEEE Trans. Syst. Man Cybern. C 37, 622–632.

Bazan, J., Nguyen, H.S., Nguyen, S.H., Synak, P., Wroblewski, J., 2000. Rough set algorithms in classification problem. In: Polkowski, L., Tsumoto, S., Lin, T.Y. (Eds.), Rough Set Methods and Applications. Physica-Verlag, Heidelberg, New York, pp. 49–88.

Blake, C.L., Merz, C.J., 1998. UCI Repository of Machine Learning Databases, University of California at Irvine. <http://www.ics.uci.edu/~mlearn/>.

Colorni, A. et al., 1996. Heuristics from nature for hard combinatorial optimization problems. Internat. Trans. Oper. Res. 3 (1), 1–21.

Dorigo, M., Blum, C., 2005. Ant colony optimization theory: A survey. Theor. Comput. Sci. 344, 243–278.

Dorigo, M., Di Caro, G., 1999. The ant colony optimization meta-heuristic. In: Corne, D., Dorigo, M., Glover, F. (Eds.), New Ideas in Optimization. McGraw-Hill, London, UK, pp. 11–32.

Dorigo, M., Gambardella, L.M., 1997. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Trans. Evolut. Comput. 1, 53–66.

Dorigo, M., Stützle, T., 2004. Ant Colony Optimization. MIT Press, Cambridge, MA.

Dorigo, M., Maniezzo, V., Colorni, A., 1996. Ant system: Optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. B 26, 29–41.

Dorigo, M., Di Caro, G., Gambardella, L.M., 1999. Ant algorithms for distributed discrete optimization. Artif. Life 5, 137–172.

Gentile, C., 2003. Fast feature selection from microarray expression data via multiplicative large margin algorithms. In: Proc. 16th Conf. on Neural Information Processing Systems (NIPS 2003).

Golub, T. et al., 1999. Molecular classification of cancer: Class discovery and class prediction by gene expression. Science 286, 531–537.

Guyon, I., Weston, J., Barnhill, S., Vapnik, V., 2002. Gene selection for cancer classification using support vector machines. Mach. Learn. 46 (1–3), 389–422.

Hedar, A., Wang, J., Fukushima, M., 2006. Tabu search for attribute reduction in rough set theory. Technical Report 2006-008, Department of Applied Mathematics and Physics, Kyoto University.

Huang, C.J., 2004. Class prediction of cancer using probabilistic neural networks and relative correlation metric. Appl. Artif. Intell. 18, 117–128.

Jensen, R., Shen, Q., 2002. Fuzzy-rough sets for descriptive dimensionality reduction. In: Proc. of 2002 IEEE Internat. Conf. on Fuzzy Systems, vol. 1, pp. 29–34.

Jensen, R., Shen, Q., 2003. Finding Rough set reducts with ant colony optimization. In: Proc. of 2003 UK Workshop Computational Intelligence, pp. 15–22.

Jensen, R., Shen, Q., 2004. Semantics-preserving dimensionality reduction: rough and fuzzy-rough-based approaches. IEEE Trans. Knowledge Data Eng. 16, 1457–1471.

Nguyen, S.H., Nguyen, H.S., 1996. Some efficient algorithms for rough set methods. In: Proc. Conf. of Information Processing and Management of Uncertainty in Knowledge-Based Systems, Granada, Spain, pp. 1451–1456.

Pawlak, Z., 1982. Rough sets. Internat. J. Comput. Inform. Sci. 11 (5), 341–356.

Pawlak, Z., 1991. Rough Sets: Theoretical Aspects of Reasoning About Data. Kluwer, Boston.

Pawlak, Z., 1996. Rough sets and data analysis. In: Proc. on Asian fuzzy systems Symp., pp. 1–6.

Pawlak, Z., Skowron, A., 2007. Rudiments of rough sets. Inform. Sci. 177, 3–27.

Raman, B., Ioerger, T.R., 2002. Instance-based filter for feature selection. J. Mach. Learn. Res. 1, 1–23.

Skowron, A., Pal, S.K., 2003. Rough sets, pattern recognition, and data mining. Pattern Recognition Lett. 24, 829–933.

Slezak, D., Wroblewski, J., 2003. Order based genetic algorithms for the search of approximate entropy reducts. In: Wang, G.Y., et al. (Eds.), RSFDGrC. LNAI, vol. 2639. Chongqing, China, pp. 308–311.

Stefanowski, J., 1998. On rough set based approaches to induction of decision rules. In: Skowron, A., Polkowski, L. (Eds.), . In: Rough Sets in Knowledge Discovery, vol. 1. Physica-Verlag, Heidelberg, pp. 500–529.

Stützle, T., Hoos, H.H., 2000. Max–min ant system. Future Generat. Comput. Syst. 16, 889–914.

Swiniarski, R.W., Skowron, A., 2003. Rough set methods in feature selection and recognition. Pattern Recognition Lett. 24, 833–849.

Theodoridis, S., Koutroumbas, K., 2006. Pattern Recognition. Academic Press.

Wang, X., Yang, J., Teng, X., Xia, W., Jensen, R., 2007. Feature selection based on rough sets and particle swarm optimization. Pattern Recognition Lett. 28, 459–471.

Weston, J., Elisseeff, A., Scholkopf, B., Tipping, M., 2003. Use of the zero-norm with linear models and kernel methods. J. Mach. Learn. Res. 3, 1439–1461.

Wong, S.K.M., Ziarko, W., 1985. On optional decision rules in decision tables. Bull. Polish Acad. Sci. 33, 693–696.

Wróblewski, J., 1995. Finding minimal reducts using genetic algorithms. In: Proc. 2nd Annual Joint Conf. on Information Sciences, Wrightsville Beach, NC, September 28–October 1, pp. 186–189.

Xing, E., Jordan, M., Karp, R., 2001. Feature selection for high-dimensional genomic microarray data. In: Proc. 18th ICML.

Zlochin, M., Birattari, M., Meuleau, N., Dorigo, M., 2004. Model-based search for combinatorial optimization: A critical survey. Ann. Oper. Res. 131, 373–395.