2

Integration Method of Ant Colony Algorithm and Rough Set Theory for Simultaneous Real Value Attribute Discretization and Attribute Reduction

Yijun He[1], Dezhao Chen[1] and Weixiang Zhao[2]
[1]*Department of Chemical Engineering, Zhejiang University*
[2]*Department of Mechanical and Aeronautical Engineering, University of California, Davis*
[1]*People's Republic of China,* [2]*U.S.A*

1. Introduction

Discretization of real value attributes (features) is an important pre-processing task in data mining, particularly for classification problems, and it has received significant attentions in machine learning community (Chmielewski & Grzymala-Busse, 1994; Dougherty et al., 1995; Nguyen & Skowron, 1995; Nguyen, 1998; Liu et al., 2002). Various studies have shown that discretization methods have the potential to reduce the amount of data while retaining or even improving predictive accuracy. Moreover, as reported in a study (Dougherty et al., 1995), discretization makes learning faster. However, most of the typical discretization methods can be considered as univariate discretization methods, which may fail to capture the correlation of attributes and result in degradation of the performance of a classification model.

As reported (Liu et al., 2002), numerous discretization methods available in the literatures can be categorized in several dimensions: dynamic vs. static, local vs. global, splitting vs. merging, direct vs. incremental, and supervised vs. unsupervised. A hierarchical framework was also given to categorize the existing methods and pave the way for further development. A lot of work has been done, but still many issues remain unsolved, and new methods are needed (Liu et al. 2002).

Since there are lots of discretization methods available, how does one evaluate discretization effects of various methods? In this study, we will focus on simplicity based criteria while preserving consistency, where simplicity is evaluated by the number of cuts. The fewer the number of cuts obtained by a discretization method, the better the effect of that method. Hence, real value attributes discretization can be defined as a problem of searching a global minimal set of cuts on attribute domains while preserving consistency, which has been shown as NP-hard problems (Nguyen, 1998).

Rough set theory (Pawlak, 1982) has been considered as an effective mathematical tool for dealing with uncertain, imprecise and incomplete information and has been successfully applied in such fields as knowledge discovery, decision support, pattern classification, etc. However, rough set theory is just suitable to deal with discrete attributes, and it needs discretization as a pre-processing step for dealing with real value attributes. Moreover, attribute reduction is another key problem in rough set theory, and finding a minimal

Source: Swarm Intelligence: Focus on Ant and Particle Swarm Optimization, Book edited by: Felix T. S. Chan and Manoj Kumar Tiwari, ISBN 978-3-902613-09-7, pp. 532, December 2007, Itech Education and Publishing, Vienna, Austria

attribute reduction has also been shown as a NP-hard problem (Komorowski et al., 1998). Two main approaches to find a minimal attribute reduction can be categorized as discernibility functions-based and attribute dependency-based, respectively (Han et al., 2004). These algorithms, however, suffer from intensive computations of either discernibility functions or positive regions. An alternative way to find a minimal attribute reduction is to adopt meta-heuristic algorithms, such as genetic algorithm (Wróblewski, 1995), particle swarm optimization (Wang et al., 2007), and ant colony algorithm (Jensen & Shen, 2003). To our knowledge, there are rare studies about how discretization pre-processing influences attribute reduction and how one integrates these two steps into a unified framework. In this chapter, we will try to give a systematic view into this problem, and will introduce ant colony algorithm to solve it.

Ant colony algorithm (Colorni et al., 1991; Dorigo et al., 1996) is a kind of meta-heuristic algorithms and has been successfully applied to solve many combinatorial optimization problems, such as travelling salesman problem (Gambardella & Dorigo, 1995; Dorigo et al., 1996; Dorigo & Gambardella, 1997; Stützle & Hoos, 1997), sequential ordering problem (Gambardella & Dorigo, 2000), generalized assignment problem (Lourenço & Serra, 2002), scheduling problem (Stützle, 1998; Merkle et al., 2002; Merkle & Middendorf, 2003), network routing problem (Schoonderwoerd et al., 1996; Di Caro & Dorigo, 1998;), set covering problem (Hadji et al., 2000; Rahoual et al., 2002; Lessing et al., 2004), etc. Great achievements of ant colony algorithm have attracted lots of attentions from different disciplinary researchers, and its application fields have been expanded from combinatorial optimization to continuous optimization problems, single-objective problems to multi-objective problems, static problems to dynamic problems, etc.. In this chapter, we just focus on the discrete style of ant colony algorithm, and it is reconstructed to be adapted to simultaneously solve real value attribute discretization and attribute reduction.

This chapter is structured as follows. In section 2, preliminaries of rough set theory will be shortly described firstly, secondly the mathematical definition of real value attribute discretization and attribute reduction will be introduced, and then the relationship between discretization and reduction will be discussed and a unified framework will be proposed by introducing a weight parameter. The relationship between the unified framework and set covering problems will be analyzed in section 3. A detailed implementation of ant colony algorithm for simultaneously solving attribute discretization and reduction will be presented in section 4. The experimental results and discussion will be given in sections 5. Section 6 will make conclusions and provide future research directions.

2. Rough set theory, discretization and reducts

2.1 Preliminaries of rough set theory

In rough set theory, table, also called information system, is often used to organize sample data, where rows and columns of a table denote objects and attributes, respectively. If attributes in a table consist of conditional attributes and decision attribute, the information system will be called a decision table. The mathematical definition of an information system and a decision table can be shown as follows.

Information system and decision table (Komorowski et al., 1998) : an information system is a pair $\mathcal{A} = (U, A)$, where U is a non-empty finite set of objects called a universe and A is a non-empty finite set of attributes such that $a : U \to V_a$ for every $a \in A$. The set V_a is called

the value set of a. A decision table is an information system of the form $\mathcal{A} = (U, A \bigcup \{d\})$, where $d \notin A$ is the decision attribute. The elements of A are called conditional attributes. Let $U = \{x_1, \cdots, x_n\}$, $V_d = \{1,2,\cdots, r(d)\}$, and $A = \{a_1, \cdots, a_m\}$, where n, $r(d)$ and m are the numbers of samples, decision classes, and attributes, respectively. The decision attribute d determines a partition $\{X_1, \cdots, X_{r(d)}\}$ of the universe U, where $X_k = \{x \in U : d(x) = k\}$ for $k = 1, \cdots, r(d)$. The set X_k is called the k^{th} decision class of $\mathcal{A}$.

Intuitively, a decision table $\mathcal{A}$ is considered consistent if the following statement is satisfied. If $\forall\, x_1, x_2 \in U$ and $d(x_1) \neq d(x_2)$, then $\exists a \in A$, $a(x_1) \neq a(x_2)$.

In other words, for any two objects with different decision class, there at least exists one attribute to discern them. In this study, we assume that the decision table $\mathcal{A}$ is consistent, if not specifically denoted.

Before the discussion of attribute reduction, the notion of relative indiscernibility relation based on decision table $\mathcal{A}$ is briefly introduced as follows.

Relative indiscernibility relation: for any subset of attributes $B \subseteq A$, an equivalence relation called the relative B-indiscerniblity relation, denoted by $IND(B,d)$, is defined by $IND(B,d) = \{(x_i, x_j) \in U \times U : (d(x_i) = d(x_j)) \vee (Inf_B(x_i) = Inf_B(x_j))\}$.

where $Inf_B(x) = \{(a, a(x)) : a \in B \text{ for } x \in U\}$ is called B-information function. For any two objects x_i and x_j satisfying relation $IND(B,d)$ are either belonging to the same decision class, or having the same value for every attribute a in subset B. Hence, if any objects x_i and x_j fitting relation $IND(A,d)$ are belonging to the same class, the decision table $\mathcal{A}$ is considered consistent.

2.2 Reduct and Discretization

In this subsection, the notions of reduct and discretization will be firstly introduced, and then the relationship between them will be discussed based on the notion of a distinction table.

An attribute $a \in B$ is relative dispensable in a subset of attributes $B \subseteq A$, if $IND(B - \{a\}, d) = IND(B,d)$, otherwise attribute a is relative indispensable. It means that attribute a can not influence the indiscernibility relation if it is dispensable.

A reduct of decision table $\mathcal{A}$ is a minimal set of attributes $B \subseteq A$ such that $IND(B,d) = IND(A,d)$ (Komorowski et al., 1998). In other words, a reduct is a minimal set of attributes from A that preserves the partitioning of the universe and hence the ability to perform classifications as the whole attribute set A does. Meanwhile, the minimal set of attributes $B \subseteq A$ means that every attribute $a \in B$ is indispensable with respect to B.

Now, a formal description of the real value discretization can be presented as follows (Komorowski et al., 1998).

We assume $V_a \in [l_a, r_a) \subset \Re$ to be a real interval for any $a \in A$ and a given decision table $\mathcal{A} = (U, A \bigcup \{d\})$ to be consistent. Any pair of (a,c) where $a \in A$ and $c \in \Re$ will be called a cut on V_a. For $a \in A$, any set of cuts: $\{(a, c_1^a), \cdots, (a, c_{k_a}^a)\}$ on V_a defines a partition $\mathbf{P}_a$ of V_a into subintervals expressed by $\mathbf{P}_a = \{[c_0^a, c_1^a), [c_1^a, c_2^a), \cdots, [c_{k_a}^a, c_{k_a+1}^a)\}$, where

$l_a = c_0^a < c_1^a < c_2^a < \cdots < c_{k_a}^a < c_{k_a+1}^a = r_a$, and $V_a = [c_0^a, c_1^a) \bigcup [c_1^a, c_2^a) \bigcup \cdots \bigcup [c_{k_a}^a, c_{k_a+1}^a)$. $C_a = \{c_1^a, \cdots, c_{k_a}^a\}$ is used to represent the set of cuts on V_a for convenience, which can uniquely define the partition $\mathbf{P}_a$. Hence any family $\mathbf{P} = \{\mathbf{P}_a : a \in A\}$ is called a partition on decision table $\mathcal{A}$ and can be represented by $\mathbf{P} = \bigcup_{a \in A} C_a$.

Moreover, according to the notion of the set of cuts, a new decision table $\mathcal{A}^{\mathbf{P}} = (U, A^{\mathbf{P}} \bigcup \{d\})$ can be defined, where $A^{\mathbf{P}} = \{a^{\mathbf{P}} : a^{\mathbf{P}}(x) = i \Leftrightarrow a(x) \in [c_i^a, c_{i+1}^a), x \in U, i = \{0, \cdots, k_a\}\}$.

Using meta-heuristic algorithms for real value attribute discretization, usually one needs to identify an initial set of cuts first and then screen these cuts. In this paper, each cut in an initial set of cuts is also called candidate cut. Assuming an arbitrary attribute $a \in A$ defines a sequence of its different attribute values ($x_1^a < \cdots < x_{n_a}^a$) and the length of this sequence is denoted by n_a ($n_a \le n$), the initial set of cuts on V_a can be represented by $C_a = \{\frac{x_1^a + x_2^a}{2}, \cdots, \frac{x_{n_a-1}^a + x_{n_a}^a}{2}\}$ which includes $n_a - 1$ elements, and the initial set of cuts on decision table $\mathcal{A}$ will be $\mathbf{P} = \bigcup_{a \in A} C_a$. Then, a new decision table $\mathcal{A}^{\mathbf{P}} = (U, A^{\mathbf{P}} \bigcup \{d\})$ can be established. It should be noticed that the new decision table with the initial set of cuts is consistent if the original decision table $\mathcal{A}$ is consistent.

In order to select cuts from the initial set of cuts, the effect of each cut on classification performance should be analyzed to construct a distinction table DT. Each pair of two objects with different decision classes correspond to a row in distinction table, so the total number of rows in a distinction table is $N = \sum_{p=1}^{r(d)-1} card(X_p) \sum_{q=p+1}^{r(d)} card(X_q)$, where $card(X_p)$ denotes the number of objects in decision class p. Each cut from the initial set of cuts corresponds to a column in distinction table, so the total number of columns is $M = \sum_{a \in A} (n_a - 1)$. Let the i^{th} row and j^{th} column of a distinction table represent one pair of two objects, such as (x_p, x_q) where $d(x_p) \neq d(x_q)$, and one cut of initial set of cuts, respectively. Hence, if the j^{th} column (or cut) can discern these two objects (x_p, x_q), the entry value of the distinction table corresponding to the i^{th} row and j^{th} column, denoted by dt_{ij}, is equal to 1; otherwise, dt_{ij} is equal to 0.

Obviously, a decision table is considered consistent if each row of the distinction table has at least one entry with the value of 1, which also means that there at least exists one discernable cut for any pair of objects from different decision classes.

According to the definition of distinction table, a real value attribute discretization problem can be equivalently represented as searching a minimal set of cuts $J \subseteq \{1, \cdots, M\}$ such that for any row $i \in \{1, \cdots, N\}$ of the decision table, there at least exists one cut $j \in J$ whose value of dt_{ij} is equal to 1. That is to say the minimal set of cuts J from the whole initial set of cuts can preserve the consistence of decision table. Correspondingly, an attribute reduction problem can also be represented as searching the set of cuts $J \subseteq \{1, \cdots, M\}$ corresponding to the minimal number of attributes such that for any row $i \in \{1, \cdots, N\}$ of the decision table, there at least exists one cut $j \in J$ whose value of dt_{ij} is equal to 1.

According to the above definitions, it is concluded that both the real value attribute discretization and reduction can be defined as finding a minimal set of cuts from the initial set of cuts, and the only difference between them is the computation of objective function. Hence, let $f_1(J)$ and $f_2(J)$ denote the number of cuts in J and the number of attributes corresponding to the set of cuts J, respectively. Consequently, a weight parameter w is introduced to balance these two objective functions, and thus both the real value attribute discretization and reduction can be further syncretised into a unified framework, which can be described as equation (1).

$$\begin{aligned} &\min_{J\subseteq\{1,\cdots,M\}} F(J) = \left[f_1(J)\right]^w + \left[f_2(J)\right]^{1-w} \\ &\text{s.t.} \sum_{j\in J} dt_{ij} \geq 1, \forall i \in \{1,\cdots,N\} \end{aligned} \tag{1}$$

where the i^{th} constraint, $\sum_{j\in J} dt_{ij} \geq 1$, represents that there at least exists a cut in J that can discern the pair of two objects in the i^{th} row. This insures the consistence of the obtained reduct decision table with regard to J.

In general, the weight parameter w is dependent on a practical problem and user's preference, so it is usually determined empirically. However, the above problem can also be described as a bi-objective optimization problem, which can be further solved by multi-objective optimization methods to provide a Pareto-optimal solution set. Thus the decision maker can decide to choose one Pareto-optimal solution based on his preference and justification. In this study, we just focus on a single objective optimization problem through introducing the weight parameter w. Moreover, the costs of attributes may be different, so the objective function in equation (1) should be modified while taking account of the costs of attributes. An attribute associated with lower cost will be favored. The proposed algorithm of this study is able to deal with different costs of attributes, but we just focus on the problem with the same costs of all attributes in the case studies of this study.

3. Set covering problem

The set covering problem (SCP) is an NP-hard problem combinatorial optimization problem that arises in a large variety of practical applications, such as resource allocation (Revelle, et al., 1970), airline crew scheduling (Housos & Elmoth, 1997), and so on. In this subsection, we study the relationship between SCP and the unified problem shown as equation (1), which can help us to design a more efficient ant colony algorithm profiting from the existing different heuristic and local search methods for solving SCP.

Given a zero-one matrix $\boldsymbol{S}$ with m rows and n columns, let $cost_j$ be the cost of column j, where $j = 1,\cdots,n$. The i^{th} row is said to be covered by column j if the entry s_{ij} is equal to 1. The problem of set covering is to find a subset of columns with a minimal cost to cover all rows, and can be formulated as equation (2), where $\boldsymbol{x} = [x_1,\cdots,x_n]^{\text{T}}$ denotes a solution of SCP and x_j denotes the j^{th} element of the solution $\boldsymbol{x}$ indicating whether column j belongs to the solution based on the value of x_j ("1", belonging while "0", not belonging).

$$\begin{aligned} &\min f(\boldsymbol{x}) = \sum_{j=1}^{n} cost_j \cdot x_j \\ &\text{s.t.} \sum_{j=1}^{n} s_{ij} x_j \geq 1, \quad i = 1, \cdots, m \\ &\quad x_j \in \{0,1\}, \qquad j = 1, \cdots, n \end{aligned} \tag{2}$$

If the distinction table DT is regarded as matrix $\boldsymbol{S}$, the unified problem of simultaneous real value attribute discretization and reduction shown as equation (1) can be handled as SCP, whose main goal is also to find a subset of columns via a minimal objective function to cover all rows. The only minor difference between them is the definition of the objective function. Hence, the existing heuristic methods for solving SCP should be modified to be suited to solve the unified problem shown as equation (1).

Now, we will reformulate the unified problem in equation (1) based on the description form of SCP, as shown in equation (3). In this study, we assume the candidate cuts belonging to the same attribute are arranged together.

$$\begin{aligned} &\min f(\boldsymbol{x}) = \left[f_1(\boldsymbol{x})\right]^w + \left[f_2(\boldsymbol{x})\right]^{1-w} \\ &\quad f_1(\boldsymbol{x}) = \sum_{j=1}^{n} x_j \\ &\quad f_2(\boldsymbol{x}) = \sum_{l=1}^{L} cost_l \cdot y_l \\ &\text{s.t. } y_l = \begin{cases} 1, \text{ if } \sum_{p=1}^{n_l} x_{Index+p} \geq 1, Index = \begin{cases} 0, \ l = 1 \\ \sum_{q=1}^{l-1} n_q, l > 1 \end{cases} \\ 0, \text{ otherwise} \end{cases} \\ &\quad l = 1, \cdots, L \\ &\quad \sum_{j=1}^{n} s_{ij} x_j \geq 1, \quad i = 1, \cdots, m \\ &\quad x_j \in \{0,1\}, \qquad j = 1, \cdots, n \end{aligned} \tag{3}$$

where L is the number of attributes; n_l is the number of candidate cuts on the l^{th} attribute domain; $n = \sum_{l=1}^{L} n_l$ is the total number of candidate cuts on the whole attributes domain, also called the number of columns; y_l is the indicator whether the l^{th} attribute is selected with regard to solution $\boldsymbol{x}$; m is the number of rows. In the case studies, we assume costs of all attributes to be the same.

Up to now, the relationship between SCP and the unified framework is analyzed, and the similarities and differences between them are discussed. In the next section, we will propose a novel ant colony algorithm for solving the problem shown in equation (3).

4. Ant colony algorithm

For simultaneously solving real value attribute discretization and reduction shown as equation (3), solution construction and pheromone update are two key issues for an ant colony algorithm. In the solution construction step, heuristic information should be reasonably designed to improve the optimization efficiency. Moreover, we will also introduce a local search strategy to improve the search speed of algorithm.

4.1 Fundamental notions

In this subsection, we will introduce some notions which will be used for the description of the algorithm.

Search domain, feasible domain and feasible solution: a search domain consists of 2^n solutions, where n denotes the total number of candidate cuts on the whole attributes domain; those solutions in search domain meeting the constraints of equation (3) are denoted as feasible solutions, and all feasible solutions form a feasible domain.

Cover: if s_{ij} is equal to 1, we can say the i^{th} row is covered by the j^{th} column. Let $\kappa_j = \sum_{i=1}^{m} s_{ij}$ denote the number of rows covered by the j^{th} column. In general, the larger the value of κ_j, the more important the j^{th} column.

Node, taboo node set and feasible node set: every element of set $\{1,\cdots,n\}$ is called a node, let $node_j$ denote the j^{th} node, where $j=1,\cdots,n$. Let $tabu_k$ denote the set of nodes that has been visisted by the k^{th} ant and be called taboo node set which is set equal to null initially. Let $allow_k$ denote the set of nodes that remain to be visisted by the k^{th} ant and be called feasible node set which is set equal to an all columns set $\{1,\cdots,n\}$ initially. In nature, the solution construction is a process of iteratively selecting unvisited node from $allow_k$ until a feasible solution is constructed. From another point of view, if $tabu_k$ covers all rows, the solution construction ends and $tabu_k$ is a feasible solution.

Covered rows set and uncovered rows set: Let $CRS = \{i \mid \exists j \in tabu_k, s_{ij} = 1\}$ and $UCRS = \{1,\cdots,m\} \setminus CRS$ denote the set of covered rows and that of uncovered rows by $tabu_k$ respectively. During the process of solution construction, the number of covered rows will increase, and the number of uncovered rows will decrease. The solution construction will continue until the set of uncovered rows $UCRS$ is equal to null set.

Pheromone: let $\tau_j(t)$ denote the quantity of pheromone defined in the j^{th} column at time t. At time 0, all columns' pheromones τ_0 are set equal to 0.5.

Heuristic information: the heuristic information adopted in this study differs from that in other combinatorial optimization problems. Generally, in other combinatorial optimization problems, such as travelling salesman problem, the heuristic information is calculated before the solution construction step, while in this study, it is dynamically calculated during solution construction. Let η_j denote the heuristic information in the j^{th} column, where $j \in allow_k$. In this study, the calculation of heuristic information not only depends on cut infomation (which is usually adopted by the existing heuristic methods in literatures), but

also depends on extra attribute information. The concrete implementation for each solution construction step by ant k is shown below.

1. For each cut or column, i.e. $j \in allow_k$, determine which attribute it belongs to, for example, assume the j^{th} column belong to the l^{th} attribute.
2. Obtain the value of CN_l, which denotes the number of selected candidate cuts of the l^{th} attribute during solution contruction.
3. Calculate extra attribute information v_l, defined by equation (4).

$$v_l = \begin{cases} c / cost_l, \text{ if } 0 < CN_l \le CN_{\max} \\ 1 / cost_l, \text{ otherwise} \end{cases} \tag{4}$$

where c (c>1) is a prescribed parameter. It means that if none candidate cuts of l^{th} attribute are selected or the number of selected cuts of l^{th} attribute is larger than $CN_{\max}$, the value of v_l is set equal to $1/cost_l$, otherwise, it is set equal to $c/cost_l$. On the one hand, we tend to choose a candidate cut of an attribute whose several candidate cuts have been selected during solution construction, because this may help decrease the number of selected attributes; on the other hand, we would not like to choose too many candidate cuts of an attribute, because this is not conducive to understanding the decision model and could decreases the robustness of the decision model as well. Hence, a prescribed parameter, $CN_{\max}$, which denotes the maximum number of selected candidate cuts for each attribute, is introduced to decrease the probability of the above phenomenon. In this study, the values of c and $CN_{\max}$ are empiricallly set equal to 4 and 5, respectively.

4. Calculate cut information κ_j defined by equation (5), which denotes the number of rows covered by the j^{th} column from the set of uncovered rows $UCRS$.

$$\kappa_j = \sum_{i \in UCRS} s_{ij}, \ j \in allow_k \tag{5}$$

5. Calculate the heuristic information η_j defined by equation (6), using cut information and extra attribute information.

$$\eta_j = \kappa_j \cdot v_l, \ j \in allow_k \tag{6}$$

6. Nomalize the heuristic information via equation (7).

$$\eta_j = \frac{\eta_j}{\sum_{q \in allow_k} \eta_q}, \ j \in allow_k \tag{7}$$

Selection probability: in the solution construction step, let $p_k(j,t)$ denote the selection probability of column j by the k^{th} ant at time t, and be expressed by equation (8).

$$p_k(j,t) = \frac{\tau_j(t) * [\eta_j]^\beta}{\sum_{q \in allow_k} \tau_q(t) * [\eta_q]^\beta}, \ j \in allow_k \tag{8}$$

where β is a heuristic factor, which determines the relative importance of pheromone versus heuristic information. In this study, it is set equal to 2.

4.2 Solution construction

Unlike the solution construction of travelling salesman problems, where each ant must visit each city node, in this study, each ant will end the solution construction when all rows are covered by the selected column nodes.

The detailed implementation of solution construction by ant k is shown as follows.

Step1 set $tabu_k = \Phi$, $allow_k = \{1, \cdots, n\}$, $CRS = \Phi$, $UCRS = \{1, \cdots, m\}$, and $CN_l = 0$ for all $l \in \{1, \cdots, L\}$;

Step2 calculate the heuristic information η_j based on section 4.1, where $j \in allow_k$;

Step3 generate a uniform random number r in [0, 1]. If the value of r is less than q_0, the next node u is selected by equation (9); otherwise, the next node u is selected based on the probability shown as equation (8). It should be noticed that q_0 can be considered as an exploitation probability factor and is used to control how strongly an ant exploit deterministically the combined past search experience and heuristic information. By adjusting parameter q_0, we can balance the trade-off between exploitation and biased exploration.

$$u = \arg\max_{j \in allow_k} \{\tau_j(t) * [\eta_j]^\beta\} \tag{9}$$

Step4 implement local pheromone update operation for column u by using equation (10).

$$\tau_u(t) = (1 - \xi)\tau_u(t) + \xi\tau_0 \tag{10}$$

where ξ, $0 < \xi < 1$, is a parameter called local pheromone update strength factor. The effect of local pheromone update is to make the desirability of columns change dynamically: each time when an ant choose a column, this column becomes slightly less desirable for the other ants. Hence, this can prevent ants from converging to a common solution and help increase exploration.

Step5 determine which attribute the selected node u belongs to If this node belongs to the l^{th} attribute, set $CN_l = CN_l + 1$, $tabu_k = tabu_k \bigcup \{u\}$, $allow_k = allow_k \setminus \{u\}$, $CRS = CRS \bigcup \{i \mid s_{ij} = 1, i \in UCRS, j = u\}$, and $UCRS = UCRS \setminus \{i \mid s_{ij} = 1, i \in UCRS, j = u\}$;

Step6 if $UCRS$ is not a null set, go back to **Step2**; otherwise, solution construction is finished, and return a feasible solution $tabu_k$.

4.3 Redundant columns remove

After a feasible solution $tabu_k$ is constructed, all redundant columns will be removed from the solution. Column j is considered redundant if $tabu_k \setminus \{j\}$ is also a feasible solution. Let

$\sigma_j = \min_{i \in Cover(j)} (\omega_i - 1)$ represent an associate variable to judge whether column j is redundant, where $j \in tabu_k$, $Cover(j) = \{i \mid s_{ij} = 1\}$ denotes the subset of rows covered by column j, and ω_i is the number of selected columns covering row i. It should be noticed that column j is redundant if and only if $\sigma_j > 0$.

If there is only one redundant column, it is easy to remove this column from the solution. However, if there are several redundant columns, we will remove the redundant column step by step until there is none redundant column. The detailed implementation is shown as follows.

Step1 calculate the value of σ_j, where $j \in tabu_k$, and determine the set of redundant columns, $RCS = \{j \mid \sigma_j > 0\}$. If RCS is equal to null set, stop; otherwise, go to **Setp2**;

Step2 for each column $j \in RCS$, determine which attribute it belongs to. And also let RAS denote the subset of attributes at least one of whose selected candidate cuts is redundant;

Step3 calculate the value of CN_l defined in subsection 4.1, where $l \in RAS$, and sort them by ascending. If the minimal value is equal to 1, go to **Step4**; otherwise, go to **Step5;**

Step4 find the subset of attributes $RAS_{\min}$ with minimal value CN_l from RAS. If there exist several attributes with the same largest cost in $RAS_{\min}$, randomly remove one column which belongs to one of attributes in $RAS_{\min}$; otherwise, remove the column corresponding to the largest attribute cost. Set $tabu_k = tabu_k \setminus \{j\}$, and go back to **Step1**;

Step5 find the subset of attributes $RAS_{\max}$ with maximal value CN_l from RAS. If there exist several attributes with the same largest cost in $RAS_{\max}$, randomly remove one column which belongs to one of attributes in $RAS_{\max}$; otherwise, remove the column with largest corresponding attribute cost. Set $tabu_k = tabu_k \setminus \{j\}$, and go back to **Step1**.

Based on the above discussion about the redundant column removal, it should be noticed that the goal of **Step4** is to decrease the number of attributes in the final solution while taking account of the costs of attributes; moreover, the goal of **Step5** is conducive to creating a final decision model with smaller number of cuts on each attribute domain.

4.4 Local search

To improve the solution quality and global convergence speed, a simple local search procedure is applied after solution construction. The introduced local search consists of two phases, removing columns and adding columns.

In the phase of removing columns, a number of columns determined by a user-defined parameter λ, $0 < \lambda < 1$, are removed from the constructed feasible solution. However it may result in the infeasibility of a new partial solution because some rows will not be covered. The parameter λ is used to control how many columns will be removed from the solution $tabu_k$, and the new partial solution $tabu_k^{new}$ consists of $ceil(|tabu_k| \times (1-\lambda))$ columns, where $|tabu_k|$ denotes the number of columns in set $tabu_k$, $ceil$ denotes a function of round an element to the nearest integer.

In the phase of adding columns, firstly, a size reduction problem is constructed, which is based on the uncovered rows and the columns that could cover these rows, and then this size reduction problem is solved by a greedy algorithm based on the heuristic information

discussed in subsection 4.1. The column with the largest heuristic information will be selected step by step until the all rows are covered by the selected columns. The main steps of the column adding procedure are the same as the solution construction described in subsection 4.2, and the only difference between them is the step of column selection. Therefore we just briefly introduce the implementation below. Let $LLmt$ denotes the local search iteration number, which is set equal to 5 in this study.

Step1 set $t = 1$;

Step2 randomly remove $ceil(|\ tabu_k\ | \times \lambda)$ columns from solution $tabu_k$, and construct a size reduction problem. Let $tabu_k^{new}$ denotes the new partial infeasible solution;

Step3 solve this size reduction problem by a greedy algorithm based on the dynamically calculated heuristic information. Let $tabu_k^{'}$ denotes the solution of this reduced size problem, and set $tabu_k^{'} = tabu_k^{'} \bigcup tabu_k^{new}$;

Step4 remove the redundant columns from $tabu_k^{'}$;

Step5 if the objective function value of $tabu_k^{'}$ is less than that of $tabu_k$, set $tabu_k = tabu_k^{'}$;

Step6 set $t = t + 1$, and if t is less than $LLmt$, go back to **Step2**; otherwise, stop the local search procedure and return $tabu_k$.

4.5 Pheromone update

When all ants finish the solution construction procedure and the local search procedure, the operation of global pheromone update will be implemented, which includes pheromone release and pheromone evaporate. In nature, pheromone release reflects the positive feedback mechanism in ant colony algorithm; while pheromone evaporate operation follows ant biology background and helps avoid excessive pheromone drowned heuristic information to some extent.

In this study, in addition to the local pheromone update operation discussed in **Step4** of the solution construction in subsection 4.2, two types of global pheromone update operations (iteration-best based and global-best based) are also adopted, defined by equations (11) and (12), respectively.

$$\tau_j(t+1) = (1-\rho)\tau_j(t) + \rho\Delta\tau_j^{ib},$$

$$\Delta\tau_j^{ib} = \begin{cases} \left[f(S^{ib})\right]^{-1}, \text{if } j \in S^{ib} \\ 0, \quad \text{otherwise} \end{cases} \tag{11}$$

$$\tau_j(t+1) = (1-\rho)\tau_j(t) + \rho\Delta\tau_j^{gb}$$

$$\Delta\tau_j^{gb} = \begin{cases} \left[f(S^{gb})\right]^{-1}, \text{if } j \in S^{gb} \\ 0, \quad \text{otherwise} \end{cases} \tag{12}$$

where ρ, $0 < \rho < 1$, is the pheromone evaporate factor; S^{ib} denotes the best solution in current iteration; S^{gb} denotes the global best found solution since the start of the algorithm;

$f(S^{ib})$ and $f(S^{gb})$ are the objective function vale of S^{ib} and S^{gb}, respectively; $\Delta\tau_j^{ib}$ and $\Delta\tau_j^{gb}$ are the quantities of pheromone released to column j based on these two pheromone update operations, respectively.

The global-best based pheromone update operation could make the search be concentrated around a current solution and limit the exploration of possible better ones, hence it is prone to trap into local optima; while the iteration-best based pheromone update operation could alleviate the danger of trapping into local optima but it may adversely reduce the convergence speed. Therefore, a mixed strategy, proposed by Stützle & Hoos (2000), is adopted in this study. Moreoever, in this study, another regulating strategy for these two types of pheromone update operations is also introduced, and the probability of the implementation of iteration-best based pheromone update operation is based on a bell-function defined by equation (13). In each iteration, if a generated uniform random number in the interval [0, 1] is less then the value of p_{ib}, the iteration-best based pheromone update operation will be executed; otherwise, the global-best based pheromone update operation will be executed.

$$p_{ib} = \frac{1}{1+\left|\frac{t-c}{a}\right|^{2b}} \tag{13}$$

where a, b and c are three parameters, specified by user. In this study, a, b and c are set equal to $Lmt/3$ (Lmt denotes the maximal iteration number), 2.5 and 0, respectively. When Lmt is set equal to 100, the curve of number of iteration vs. p_{ib} is shown as Fig.1. In the early stage, the value of probability p_{ib} is relatively large, so the algorithm can benefit from stronger exploration of the search space, which can help escape from local optima and avoid the stagnation phenomenon. In the later stage, the value of probability p_{ib} is relatively small, so the algorithm can benefit from stronger exploitation of all best solution candidates, which can be conducive to improving the convergence speed.

4.6 Framework of ant colony algorithm

In this subsection, we will briefly present the whole framework of ant colony algorithm for simultaneous real value attribute discretization and reduction as follows.

1. Determine the candidate cuts and construct distinction table based on samples.
2. Parameters setting, such as weight parameter w, exploitation probability factor q_0, local pheromone update strength factor ξ, pheromone evaporate factor ρ, local search magnitude factor λ, number of ants $ANum$, and maximum iteration number Lmt.
3. Pheromone initialization.
4. Set $t = 1$.
5. For each ant, implement solution construction operation.
6. For each feasible solution, remove all redundant columns.
7. For the set of best solutions in this iteration, implement local search operation.
8. Determine the iteration-best solution, and update the global-best solution.
9. Implement pheromone update operation.

10. Set $t = t + 1$, if t is less than Lmt, go back to step 5; otherwise, terminate and return the best solution.

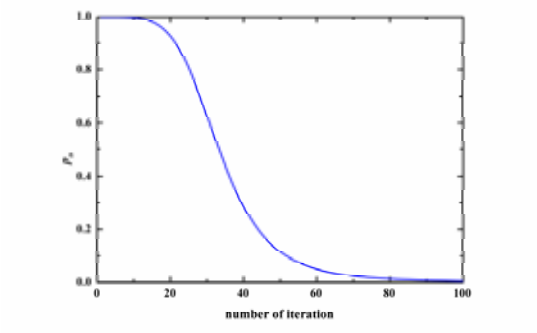


Figure 1. The curve of number of iteration vs. p_{ib} (Lmt = 100)

5. Experimental results and discussion

In this section, firstly, four datasets used for this study are briefly described; secondly, the effect of parameters on performance are studied and some suggestions for setting these parameters are presented; finally, the comparisons of our method with other three rough set theory based heuristic methods are provided.

5.1 Description of datasets

In this study, four datasets are used to validate the effectiveness of the proposed method, one dataset is nature spearmint essence (NSE), while the other three datasets are Glass, Wine and Iris obtained from the UCI (University of California, Irvine) machine learning repository available by the link: http://www.ics.uci.edu/~mlearn/MLRepository.html. The main characteristics of these four data sets are summarized in Table 1. The last column shows the number of initial cuts for these four datasets.

Dataset	Cases	Categorical attributes	Continuous Attributes	Classes	Number of Initial cuts
NSE	55	0	56	3	770
Glass	214	0	9	6	930
Wine	178	0	13	3	1263
Iris	150	0	4	3	119

Table 1. Four datasets used for this study

5.2 Parameters setting in ant colony algorithm

In this subsection, we will use the NSE dataset for studying the effect of parameters on performance, and some suggestions for setting these parameters will also be presented. In this study, we focus on the four controllable parameters which affect the performance of ant colony algorithm, exploitation probability factor q_0 shown in subsection 4.2, local pheromone update strength factor ξ shown in equation (10), pheromone evaporate factor ρ shown in equations (11) and (12), and local search magnitude factor λ shown in subsection 4.4. The number of ants $ANum$ is also an important parameter in ant colony algorithm. Dependent on the complexcity of a problem, this parameter must be sufficiently large to explore all potential solutions. In general, the larger the value of $ANum$, the better the final solution quality obtained by ant colony algorithm, and also the longer the execution time. Hence, the selection of number of ants should balance the trade-off between the solution quality and execution time. According to empirical studies, the value of $ANum$ is suggested to set equal to between [5, 20].

In order to study the effect of the parameters on the performance on the NSE dataset, a base value for each parameter is arbitrarily set as: $w = 0.5$, $q_0 = 0.5$, $\xi = 0.1$, $\rho = 0.05$, $\lambda = 0.2$, $ANum = 10$, $Lmt = 100$. When tuning one parameter, the other parameters are kept as their base values. Moreover, to reduce the influence of incidental variation, the tests for each parameter are all executed 10 times independently, and both the average value of objective function and the average iteration number of the first finding of the best solution are used for evaluating the effect of each parameter on performance. However, due to the existence of random factors, the tests can only reflect to some extent the impact of the various parameters on the final results.

5.2.1 Exploitation probability factor

Exploitation probability factor q_0 is used to control how strongly an ant deterministically exploits the combined past search experience and heuristic information. The larger the value of q_0, the larger the exploitation probability. Fig.2 and Fig.3 show the effect of exploitation probability factor q_0 on the average value of the objective function and the average iteration number, respectively. Fig.2 shows that the best algorithm performance when q_0 is equal to 0.4 or 0.7. Fig.3 shows that with the increase of the value of q_0, the average iteration number decreases. The possible reason could be with the large value of q_0, ant colony algorithm focuses on exploiting the space around the best found solution, so the convergence speed may be improved. However, a large value of q_0 may also lead to stagnation and the decrease of the probability of finding global optimum. Hence, the selection of "optimal" exploitation probability factor should balance the trade-off between exploration and exploitation performance. According to this study, the value of q_0 is suggested to set equal to between [0.4, 0.8], but it could vary in other cases.

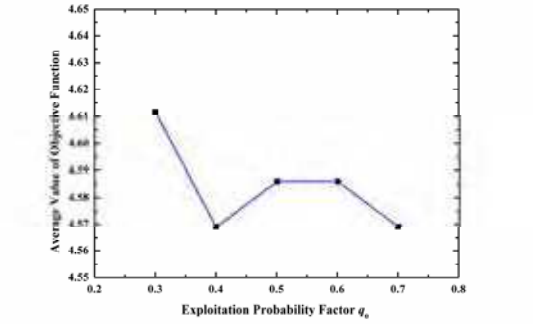


Figure 2. Effect of exploitation probability factor q_0 on average value of objective function for the NSE dataset

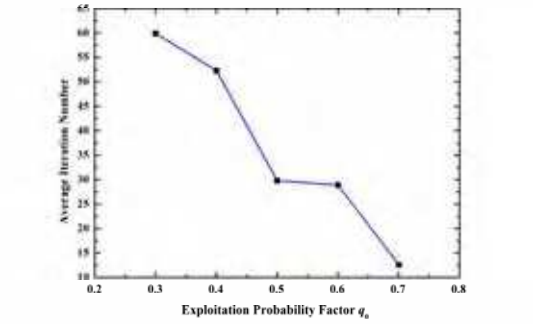


Figure 3. Effect of exploitation probability factor q_0 on average iteration number for the NSE dataset

5.2.2 Pheromone evaporate factor

In ant colony algorithm, cooperation among ants is based on pheromone, and global pheromone update operation can help ants maintain a well coordinated pheromone mediated cooperation. Fig.4 and Fig.5 show the effect of pheromone evaporate factor ρ on the average value of objective function and the average iteration number, respectively.

Fig.4 shows that the average quality of final solution is the best when ρ is equal to 0.05. On the one hand, according to equations (11) and (12), the larger the value of ρ, the more the amount of pheromone released in the best found solutions, which will attract more ants in the next iteration but probably also lead to stagnation. On the other hand, as shown in Fig.5, a large value of ρ can enhance the amount of different pheromone between the relative better solutions and relative worse solutions, so the convergence speed can be improved. However, the final solution could be a local optimum. Extremely, if the value of ρ is set equal to 0, the cooperation among ants will disappear, which would lead to a bad performance. Balancing the trade-off between the solution quality and execution time, the value of ρ is suggested to set between [0.05, 0.2].

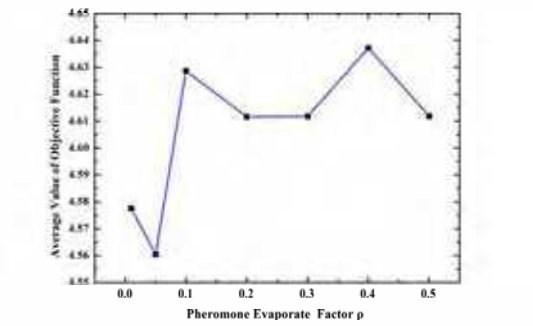


Figure 4. Effect of pheromone evaporate factor ρ on average value of objective function for the NSE dataset

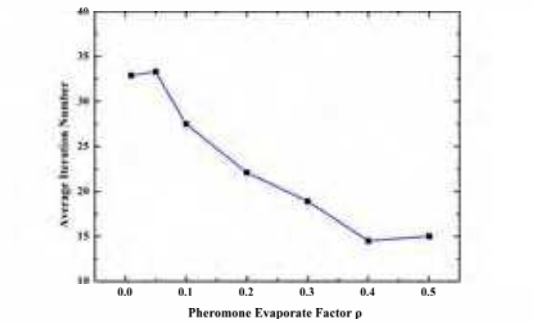


Figure 5. Effect of pheromone evaporate factor ρ on average iteration number for the NSE dataset

5.2.3 Local pheromone update strength factor

Similar to the function of pheromone evaporate factor, local pheromone update strength factor ξ also helps ants maintain a well coordinated pheromone mediated cooperation. Local pheromone update operation helps ants choose those columns that have never been explored previously, which can prevent the ants from converging to a common premature solution. Fig.6 and Fig.7 show the effect of the local pheromone update strength factor ξ on the average value of the objective function and the average iteration number, respectively. Fig.6 shows that the best average quality of final solution is obtained at ξ being 0.01 or 0.1, and Fig.7 shows the least average iteration number when ξ is equal to 0.05. In general, the local pheromone update operation just changes the desirability of columns for following ants, which can be conducive to increasing exploration. The larger the value of ξ, the more likely the probability for the system pheromone to go back to the initial level. Although it can enhance the exploration performance, it also weakens the pheromone level released by previous ants and leads to a lower convergence speed. Hence, we should reasonably select the value of ξ to balance the trade-off between exploration and convergence speed. The value of ξ is suggested to set between [0.05, 0.2] for this study.

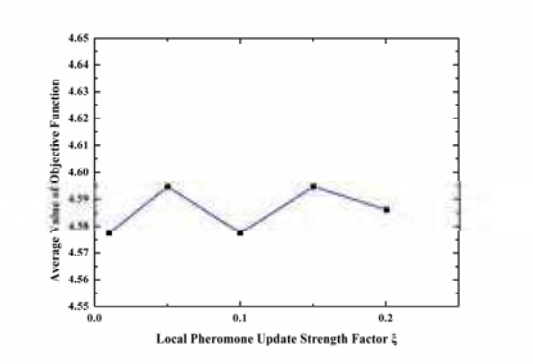


Figure 6. Effect of local pheromone update strength factor ξ on average value of objective function for the NSE dataset

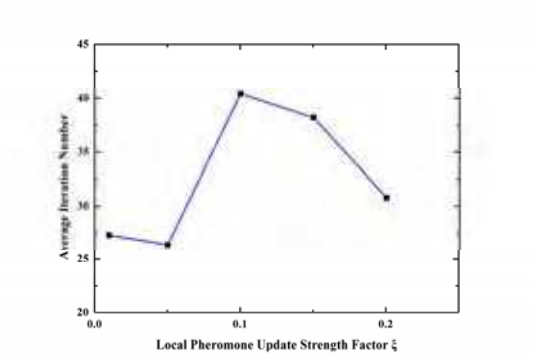


Figure 7. Effect of local pheromone update strength factor ξ on average iteration number for the NSE dataset

5.2.4 Local search magnitude factor

Local search operation usually can improve both the final solution quality and the speed of finding the best solution. Local search magnitude factor λ plays an important role in determining neighbourhood magnitude of a current solution. The larger the value of λ, the larger the neighbourhood magnitude. However, due to a limited iteration number of local search, a large value of λ could decrease the probability of finding a better solution in the neighbourhood area of the current solution. Fig.8 and Fig.9 show the effect of local search magnitude factor λ on the average value of the objective function and the average iteration number, respectively. Fig.8 shows that the best average quality of final solutions is obtained at λ being 0.3. As shown in Fig.9, with the increase of the value of λ, the average iteration number decreases. Balancing the trade-off the final solution quality and execution time, the value of λ is suggested set between [0.2, 0.4].

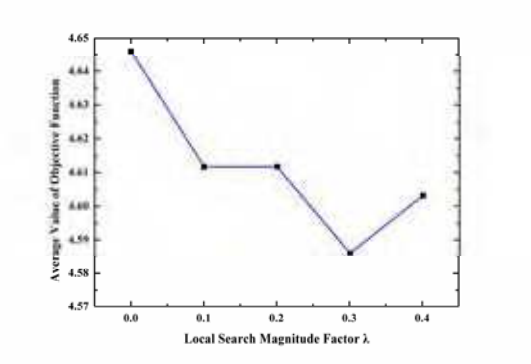


Figure 8. Effect of local search magnitude factor λ on average value of objective function for the NSE dataset

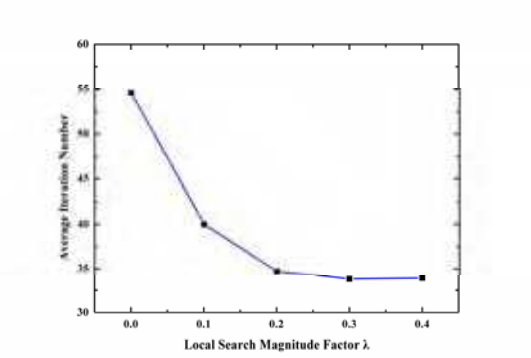


Figure 9. Effect of local search magnitude factor λ on average iteration number for the NSE dataset

5.3 Performance comparison

In this subsection, the other three rough set theory based heuristic methods are used for performance comparison with ant colony algorithm. These three methods include greedy method (Nguyen & Skowron, 1995), modified greedy method, and attribute importance based method (Hou et al., 2000). All of these three methods are deterministic methods, so their obtained results are unique for a given problem. The parameters setting for ant colony algorithm is shown as follows: $q_0 = 0.5$, $\xi = 0.1$, $\rho = 0.05$, $\lambda = 0.2$, $ANum = 10$, and $Lmt = 100$. To reduce the influence of incidental variation, the tests for each dataset are all executed 10 times independently. For performance comparison, on the one hand, we will provide the attribute numbers $AttrNum$ and cut numbers $CutNum$ obtained by these four methods; on the other hand, by introducing the weight parameter, we will give an exponential weighted value EWV, where $EWV = CutNum^{w} + AttrNum^{1-w}$, for a comprehsneive examination of the results obtained by these four methods. Due to the

stochastic character of ant colony algorithm, the three performance comparison metrics are the average results of the 10 executations.

Table 2 shows the performance comparison of four methods for the NSE, Glass, Wine and Iris datasets, where the weight parameter w is set equal to 0.5 for computing exponential weighted value. It can be seen from Table 2 that (1) the attribute numbers obtained by ant colony algorithm are remarkably less than those obtained by greedy method and modified greedy method for all the four datasets, (2) although ant colony algorithm yields a little more cuts than the greedy method and modified greedy method for the NSE and Glass datasets, the cut numbers generated by ant colony algorithm are less than or equal to those obtained by these two methods for the Wine and Iris datasets, (3) the cut numbers obtained by ant colony algorithm are remarkably less than those obtained by attribute importance based method for the NSE, Glass and Wine datasets, and (4) although ant colony algorithm requires a little more attributes than the attribute importance based method for the Wine dataset, the attribute numbers obtained by ant colony algorithm are less than or equal to those obtained by that method for the NSE, Glass, and Iris datasets. These superiorities of ant colony algorithm could be supported by the fact that both greedy method and modified greedy method more focus on finding the minimal number of cuts while attribute importance based method more focus on finding the minimal number of attributes. Different from them, ant colony algorithm can simultaneously consider both objectives, minimal numbers of cuts and attributes.

It can also be seen from Table 2 that the exponential weighted value EWV obtained by ant colony algorithm is better than those obtained by the other three methods for the NSE, Glass and Wine datasets; and for the Iris dataset, the exponential weighted value EWV obtained by ant colony algorithm and attribute importance based method both rank top 1. Moreover, in our experiments, the standard deviations of these three metrics in 10 runs are very close to 0 for all the four datasets, which illustrates the stability of the ant colony algorithm.

Metrics		Methods			
		Attribute Importance Based Method	Greedy Method	Modified Greedy Method	Ant Colony Algorithm
NSE	*AttrNum*	3	7	6	3
	CutNum	19	7	7	8
	EWV	6.091	5.095	5.292	4.560
Glass	*AttrNum*	5	8	8	4.4
	CutNum	41	14	14	15.6
	EWV	8.639	6.570	6.570	6.043
Wine	*AttrNum*	2	6	5	3
	CutNum	21	6	6	6
	EWV	5.997	4.899	4.686	4.182
Iris	*AttrNum*	3	4	4	3
	CutNum	6	10	10	6
	EWV	4.182	5.162	5.162	4.182

Table 2. Performance comparison with four methods for four datasets ($w = 0.5$)

Fig.10 shows the exponential weighted values of the four methods for the NSE dataset with different weight parameters. Through changing the relative importance between the cut number and the attribute number by employing the weight parameter w, ant colony algorithm can easily find the optimal cut and attribute numbers, whose corresponding exponential weighted value is better than those obtained by the other three methods. Hence, from a balanced choice of optimal attributes and cuts, ant colony algorithm outperforms the other three methods.

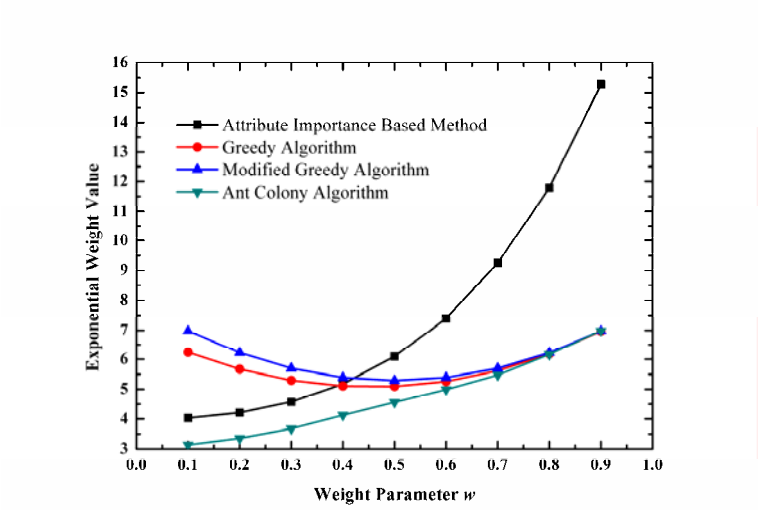


Figure 10. Performance comparison of four methods for the NSE dataset with different weight parameter w

6. Conclusions

In this study, ant colony algorithm is proposed for simultaneous real value attributes discretization and reduction. Based on the concept of distinction table in rough set theory, the relationship between discretization and reduction is discussed, and these two different problems can be integrated into a unified framework. Moreover, the relationship between this unified framework and set covering problem is analyzed. The detailed strategy for ant colony algorithm to solve this problem is proposed and applied to the four datasets. The obtained results demonstrate the effectiveness of the proposed method, showing the better performance than those of the other three rough set theory based heuristic methods.

However, this is a preliminary study, because we only considered the pre-processing step for pattern classification, and how this pre-processing step influences the final classification prediction performance has not been studied. Hence in the future research direction, we will focus on improving the performance of ant colony algorithm including global convergence performance and convergence speed and apply these proposed methods to problems with mass data. Meanwhile, we will combine the present method with any classification methods, such as rough set theory, decision tree, support vector machine, etc., for practical pattern classification problems.

7. Acknowledgements

This project was supported by the National Natural Science Foundation (20276063) of China.

8. References

Chmielewski, M.R. & Grzymala-Busse, J.W. (1994). Global discretization of attributes as preprocessing for machine learning. *Proceedings of the III International Workshop on Rough Set and Soft Computing*, pp. 294-301, San Jose, CA, November 1994

Colorni, A.; Dorigo, M. & Maniezzo V. (1991). Distributed optimization by ant colonies. *Proceeding of the First European Conference on Artificial Life*, pp. 134-142, Paris, France, 1991

Di Caro G. & Dorigo M. (1998). AntNet: distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, Vol. 9, 317-365.

Dougherty J.; Kohavi R. & Sahami M. (1995). Supervised and unsupervised discretization of continuous features. *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 194-202, Tahoe, CA, July 1995, Morgan Kaufmann

Dorigo, M. & Gambardella, L.M. (1997). Ant colony system: a cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation*, Vol.1, No.1, 53-66.

Dorigo, M.; Maniezzo, V. & Colorni A. (1996). Ant system: optimization by a colony of cooperation agents. *IEEE Transaction on Systems, Man, and Cybernetics-Part B*, Vol. 26, No. 1, 29-41.

Gambardella, L.M. & Dorigo, M. (1995). Ant-Q: a reinforcement learning approach to the travelling salesman problem. *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 252-260, Tahoe, CA, July 1995, Morgan Kaufmann

Gambardella, L.M. & Dorigo, M. (2000). Ant colony system hybridized with a new local search for sequential ordering problem. *INFORMS Journal on Computing*, Vol. 12, No. 3, 237-255.

Hadji, R.; Rahoual, M.; Talbi, E. & Bachelet, V. (2000). Ant colonies for the set covering problem. *Proceedings of ANTS' 2000 -- From Ant Colonies to Artificial Ants: Second International Workshop on Ant Algorithms*, pp. 63-66, Brussels, Belgium, September 2000, Springer-Verlag

Han, J.C.; Hu, X.H & LIN T.Y. (2004). Feature Subset Selection Based on Relative Dependency between Attributes. *Proceedings of the Fourth International Conference on Rough Sets and Current Trends in Computing*, pp. 176-185, Uppsala, Sweden, June 2004, Springer-Verlag

Hou, L.J.; Wang G.Y. & Nie. N. (2000). The discretization problems of rough set theory. *Computer Science*, Vol.27, No.12, 89-94.

Housos, E. & Elmoth T. (1997). Automatic optimization of subproblems in scheduling airlines crews. *Interfaces*, Vol. 27, No. 5, 68-77.

Jensen, R. & Shen, Q. (2003). Finding Rough Set Reducts with Ant Colony Optimization. *Proceeding of 2003 UK Workshop on Computational Intelligence*, pp. 15-22, Bristol, UK, September 2003

Komorowski, J.; Pawlak, Z.; Polkowski, L. & Skowron, A. (1998). Rough sets: A Tutorial. In: *Rough-fuzzy hybridization: a new trend in decision making*, Pal S.K. & Skowron A, pp. 3-98, Springer Verlag

Lessing, L.; Dumitrescu, I. & Stützle, T. (2004). A Comparison between ACO Algorithms for the Set Covering Problem. *ANTS' 2004, Fourth Internatinal Workshop on Ant Algorithms and Swarm Intelligence*, pp. 1-12, Brussels, Belgium, September 2004, Springer Verlag

Liu, H.; Hussain, F.; Tan, C.L. & Dash, M. (2002). Discretization: an enabling technique. *Data Mining and Knowledge Discovery*, Vol. 6, No. 4, 393-423.

Lourenço, H.R. & Serra, D. (2002). Adaptive search heuristics for the generalized assignment problem. *Mathware and Soft Computing*, Vol. 9, No. 2-3, 209-234.

Merkle, D.; Middendorf, M. & Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. *IEEE Transaction on Evolutionary Computation*, Vol. 6, No. 4, 333-346.

Merkle, D. & Middendorf, M. (2003). Ant colony optimization with global pheromone evaluation for scheduling a single machine. *Applied Intelligence*, Vol. 18, No. 1, 105-111.

Nguyen, S.H. & Skowron, A. (1995). Quantization of real value attributes: rough set and boolean reasoning approach. Proceedings of the Second Joint Annual Conference on Information Sciences, pp. 34-37, Wrightswille Beach, NC, September 1995

Nguyen, H.S. (1998). Discretization problem for rough sets methods. *Proceedings of the First International Conference on Rough Sets and Current Trends in Computing*, pp. 545-552, Warsaw, Poland, June 1998, Springer-Verlag

Pawlak, Z. (1982). Rough sets. *International Journal of Information and Computer Science*, Vol.11, No.5, 341-356.

Rahoual, M.; Hadji, R. & Bachelet V. (2002). Parallel Ant System for the Set Covering Problem. *Proceedings of the Third International Workshop on Ant Algorithms*, pp. 262-267, Brussels, Belgium, September 2002, Springer-Verlag

Revelle, C.D.; Marks, D. & Liebman, J.C. (1970). An analysis of private and public sector facilities location models. *Management Science*, Vol. 16, 692-707.

Schoonderwoerd, R.; Holland, O.; Bruten, J. & Rothkrantz, L. (1996). Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, Vol. 5, No. 2, 169-207.

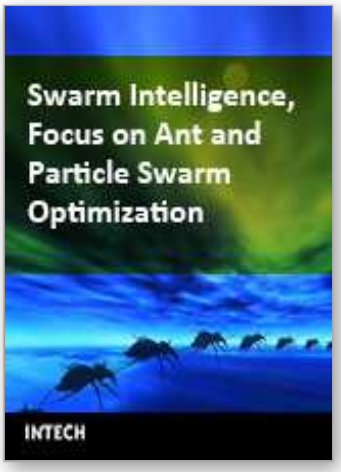
Stützle, T. & Hoos, H.H. (1997). The Max-Min ant system and local search for the travelling salesman problem. *Proceeding of the 1997 IEEE International Conference on Evolutionary Computation*, pp. 309-314, Piscataway, NJ, April 1997, IEEE Press

Stützle, T. (1998). An ant approach to the flow shop problem. *Proceeding of the Sixth European Congress on Intelligent Techniques & Soft Computing*, pp. 1560-1564, September 1998, Aachen, Germany, Verlag Mainz, Wissenschaftsverlag

Stützle, T. && Hoos, H.H. (2000). MAX-MIN ant system. *Future Generation Computer Systems*, Vol. 16, No. 8, 889-914.

Wang, X.Y.; Yang, J.; Teng, X.L.; Xia, W.J. & Jensen, R. (2007). Feature selection based on rough sets and particle swarm optimization. *Pattern Recognition Letters*. Vol. 28, No. 4, 459-471.

Wróblewski, J. (1995). Finding minimal reducts using genetic algorithms. *Proceeding of the Second Annual Join Conference on Information Sciences*, pp.186-189, September 1995, Wrightsville Beach, NC



Swarm Intelligence, Focus on Ant and Particle Swarm Optimization

Edited by FelixT.S.Chan and Manoj KumarTiwari

ISBN 978-3-902613-09-7
Hard cover, 532 pages
Publisher I-Tech Education and Publishing
Published online 01, December, 2007
Published in print edition December, 2007

In the era globalisation the emerging technologies are governing engineering industries to a multifaceted state. The escalating complexity has demanded researchers to find the possible ways of easing the solution of the problems. This has motivated the researchers to grasp ideas from the nature and implant it in the engineering sciences. This way of thinking led to emergence of many biologically inspired algorithms that have proven to be efficient in handling the computationally complex problems with competence such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), etc. Motivated by the capability of the biologically inspired algorithms the present book on "Swarm Intelligence: Focus on Ant and Particle Swarm Optimization" aims to present recent developments and applications concerning optimization with swarm intelligence techniques. The papers selected for this book comprise a cross-section of topics that reflect a variety of perspectives and disciplinary backgrounds. In addition to the introduction of new concepts of swarm intelligence, this book also presented some selected representative case studies covering power plant maintenance scheduling; geotechnical engineering; design and machining tolerances; layout problems; manufacturing process plan; job-shop scheduling; structural design; environmental dispatching problems; wireless communication; water distribution systems; multi-plant supply chain; fault diagnosis of airplane engines; and process scheduling. I believe these 27 chapters presented in this book adequately reflect these topics.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Yijun He, Dezhao Chen and Weixiang Zhao (2007). Integration Method of Ant Colony Algorithm and Rough Set Theory for Simultaneous Real Value Attribute Discretization and Attribute Reduction, Swarm Intelligence, Focus on Ant and Particle Swarm Optimization, FelixT.S.Chan and Manoj KumarTiwari (Ed.), ISBN: 978-3-902613-09-7, InTech, Available from: http://www.intechopen.com/books/swarm_intelligence_focus_on_ant_and_particle_swarm_optimization/integration_method_of_ant_colony_algorithm_and_rough_set_theory_for_simultaneous_real_value_attribut

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821