Attribute Core Computation Based on Divide and Conquer Method*

Feng Hu[1,2], Guoyin Wang[1,2], and Ying Xia[1,2]

[1] School of Information Science and Technology,
Southwest Jiaotong University,
Chengdu 600031, P.R. China
[2] Institute of Computer Science and Technology,
Chongqing University of Posts and Telecommunications,
Chongqing 400065, P.R. China
{hufeng, wanggy, xiaying}@cqupt.edu.cn

Abstract. The idea of divide and conquer method is used in developing algorithms of rough set theory. In this paper, according to the partitions of equivalence relations on attributes of decision tables, two novel algorithms for computing attribute core based on divide and conquer method are proposed. Firstly, a new algorithm for computing the positive region of a decision table is proposed, and its time complexity is $O(|U| \times |C|)$, where, $|U|$ is the size of the set of objects and C is the size of the set of attributes. Secondly, a new algorithm for computing the attribute core of a decision table is developed, and its time complexity is $O(|U| \times |C|^2)$. Both these two algorithms are linear with $|U|$. Simulation experiment results show that the algorithm of computing attribute core is not only efficient, but also adapt to huge data sets.

Keywords: Rough set, divide and conquer, positive region, attribute core.

1 Introduction

Rough set (RS) is a valid mathematical theory to deal with imprecise, uncertain, and vague information [1]. It has been applied in many fields such as machine learning, data mining, intelligent data analyzing and control algorithm acquiring successfully since it was proposed by Pawlak in 1982 [2].

In divide and conquer method, a problem which is hard to be solved directly is divided into many sub-problems and conquered respectively. The structures of the sub-problems are similar to the one of the original problem except their sizes are smaller. The divide and conquer method divide a problem into simpler

* This paper is partially supported by National Natural Science Foundation of China under Grants No.60373111 and No.60573068, Program for New Century Excellent Talents in University (NCET), Natural Science Foundation of Chongqing under Grant No.2005BA2003, Science & Technology Research Program of Chongqing Education Commission under Grant No.KJ060517.

M. Kryszkiewicz et al. (Eds.): RSEISP 2007, LNAI 4585, pp. 310–319, 2007.
© Springer-Verlag Berlin Heidelberg 2007

sub-problems iteratively in this way and the sizes of the sub-problems will be reduced to be easy enough to be processed directly [3,4].

In the study of rough set theory, the computation of positive region and attribute core are two basic operations, and their efficiencies will affect the efficiencies of further attribute reduction and value reduction. Many attribute reduction algorithms have already been proposed [5-16]. However, few of them can deal with huge data sets well. Combining the idea of divide and conquer method and partition of equivalence relation in a decision table, a huge data set can be divided into many small ones that can be processed directly easily. In addition, the complexity of the original problem could be reduced. According to the above analysis, a new algorithm for computing positive region based on divide and conquer method is proposed, and its time complexity is $O(|U| \times |C|)$. Furthermore, a new algorithm for computing attribute core based on divide and conquer is also proposed, and its time complexity is $O(|U| \times |C|^2)$.

The rest of this paper is organized as follows. In section 2, some basic notions about rough set theory are introduced. A new algorithm for computing positive region is proposed in section 3. In section 4, a novel algorithm for computing attribute core based on divide and conquer is proposed. In section 5, some experiment results are discussed. We draw some conclusions in section 6.

2 Basic Notions of Rough Set Theory

For the convenience of illustration, some basic notions of rough set theory are introduced here at first.

Def. 1 (decision table [2]) A decision table is defined as $S =< U, A, V, f >$, where U is a non-empty finite set of objects, called universe, R is a non-empty finite set of attributes, $A = C \cup D$, where C is the set of condition attributes and D is the set of decision attributes, $D \neq \emptyset$. $V = \bigcup_{p \in R} V_p$, and V_p is the domain of the attribute p. $f : U \times A \rightarrow V$ is a total function such that $f(x_i, A) \in V_p$ for every $p \in A, x_i \in U$.

Def. 2 (indiscernibility relation [2]) Given a decision table $S =< U, A = C \cup D, V, f >$, each subset $B \subseteq C$ of attribute determines an indiscernibility relation *IND(B)* as follows: $IND(B) = \{(x, y)|(x, y) \in U \times U, \forall b \in B(b(x) = b(y))\}$.

Def. 3 (lower-approximation, upper-approximation and border region [2]) Given a decision table $S =< U, C \cup D, V, f >$, for any subset $X \subseteq U$and the indiscernibility relation $IND(B)$, the B lower-approximation, upper-approximation and border region of X are defined as: $B_{\_}(X) = \bigcup_{Y_i \in U/IND(B) \wedge Y_i \subseteq X} Y_i$, $B^{-}(X) = \bigcup_{Y_i \in U/IND(B) \wedge Y_i \cap X \neq \Phi} Y_i$, $BN(X) = B^{-}(X) - B_{-}(X)$.

Def. 4 (positive region [2]) Given a decision table $S =< U, A, V, f >$. $P \subseteq A$ and $Q \subseteq A$, the P positive region of Q is defined as: $Pos_P(Q) = \bigcup_{X \in U/Q} P_{\_}(X)$.

Def. 5 (relative core [2]) Given a decision table $S =< U, A, V, f >$, $P \subseteq A$, $Q \subseteq A$, and $r \in P$. r is unnecessary in P with reference to Q if and only if

$Pos_P(Q) = Pos_{P-\{r\}}(Q)$, otherwise r is unnecessary in P with reference to Q. The core of P with reference to Q is defined as: $CORE_Q(P) = \{r|r \in P$, r is necessary in P with reference to $Q\}$. Attribute r is necessary in P with reference to Q can be written as r is relative necessary, too.

Def. 6 [2] Given a decision table $S =< U, A, V, f >$, $P \subseteq A$, $Q \subseteq A$. $\forall r \in P$, if r is necessary in P with reference to Q, we call P is independent with reference to Q.

Def. 7 (relative reduction [2]) Given a decision table $S =< U, A, V, f >$, $P \subseteq A$, $Q \subseteq A$. $Red \subset P$, if Red is independent with reference to Q and $Pos_{Red}(Q) = Pos_P(Q)$, we call Red is a reduction of P with reference to Q.

In this paper, $f(x, c)(x \in U \wedge c \in C)$ is noted as $c(x)$, and $f(x, d)(x \in U \wedge D = \{d\})$ is noted as $d(x)$.

3 Algorithm for Computing Positive Region Based on Divide and Conquer Method

In [17], a method for computing positive region is proposed by partitioning the universe of a decision table. In this paper, by reducing condition attributes and partitioning the universe of a decision table, the original decision table could be divided into many new decision tables with different attribute spaces. The method is as follows.

Theorem 1. Given a decision table $S =< U, A = C \cup D, V, f >$. $\forall c(c \in C)$, $U/\{c\}$ is a partition of S, that is, S is divided into $k(k = |IND(U/\{c\})|)$ sub-decision tables S_1, S_2,..., S_k, where, $S_k =< U_k, (C-\{c\}) \cup D, V_k, f_k >$, satisfying $_{\forall x \in U_i \forall y \in U_i} c(x) = c(y)(1 \le i \le k)$ and $_{\forall x \in U_i \forall z \in U_j} c(x) \ne c(z)(1 \le i < j \le k)$. Let $R = C - \{c\}$, $Pos^i_R(D)(1 \le i \le k)$ be the positive region of a sub decision table S_i, $Pos_C(D)$ be the positive region of the original decision table S. Then, $Pos_C(D) = \bigcup_{1 \le i \le k} Pos^i_R(D)$.

Proof: firstly, prove $Pos_C(D) \subseteq \bigcup_{1 \le i \le k} Pos^i_R(D)$.

$\forall x \in Pos_C(D)$, suppose x is assigned to sub-decision table S_k, that is, $x \in U_k$. Now, we need to prove $x \in Pos^i_R(D)$. Prove to the reverse.

Suppose $x \notin Pos^i_R(D)$, then $\exists y \in U_j(_{\forall a \in C-\{c\}}(a(x) = a(y)) \wedge (d(x) \ne d(y)))$. Since $c(x) = c(y)$, so $(_{\forall a \in C}(a(x) = a(y)) \wedge (d(x) \ne d(y)))$, that is, $x \notin Pos_C(D)$, which is conflict with the premise $x \in Pos_C(D)$. Therefore, $x \in Pos^i_R(D)$, then $Pos_C(D) \subseteq \bigcup_{1 \le i \le k} Pos^i_R(D)$. That's to say, $Pos_C(D) \subseteq \bigcup_{1 \le i \le k} Pos^i_R(D)$.

Then, prove $\bigcup_{1 \le i \le k} Pos^i_R(D) \subseteq Pos_C(D)$.

$\forall x \in Pos^i_{C-\{c\}}(D)(1 \le i \le k)$, $\forall y \in U$, if $y \notin U_i$, there is $c(x) \ne c(y)$. So, $x \in Pos_C(D)$. That's to say, $\bigcup_{1 \le i \le k} Pos^i_R(D) \subseteq Pos_C(D)$.

Therefore, $Pos_C(D) = \bigcup_{1 \le i \le k} Pos^i_R(D)$. Theorem 1 holds.

With Theorem 1, we could develop an algorithm for computing positive region based on divide and conquer.

Algorithm 1. Computing Positive Region Based on Divide and Conquer Method
Input: A decision table $S =< U, C \cup D, V, f >$
Output: Positive region $Pos_C(D)$
Step1: (*Initiative*) $Pos_C(D) = \phi$;
Step2: (*Compute positive region by invoking recursive function*)
  $Get\_Positive(U, 1)$;
Step3: (*Return*) return $Pos_C(D)$
Recursive Function $Get\_Positive$(Set $OSet$, int k)
  if ($k < 1$) or ($|OSet| < 1$) then return; end if
  if ($|OSet| = 1$) then
    $Pos_C(D) = Pos_C(D) \cup OSet$; return;
  end if
  if ($k > |C|$) then
    if $_{\forall x \in OSet \forall y \in OSet} d(x) = d(y)$ then $Pos_C(D) = Pos_C(D) \cup OSet$; end if
    return;
  end if
  Let $c = c_k$, $V^c = \phi$;
  for $i = 1$ to $|OSet|$ do
    $V^c = V^c \cup f(x_i, c)$;
  end for
  for $i = 1$ to $|V^c|$ do
    $OSet_j^c = \phi$;
  end for
  construct a mapping function $f^{'} : V^c \rightarrow N(N = 1, 2, ..., |V^c|)$, satisfying:
  $_{\forall x \in V^c \forall y \in V^c}$ $(f^{'}(x) = f^{'}(y)) \Leftrightarrow (x = y)$.
  for $i = 1$ to $|OSet|$ do
    let $j = f^{'}(f(x_i, c))$; $OSet_j^c = OSet_j^c \cup \{x_i\}$;
  end for
  for $j = 1$ to $|V^c|$ do
    recursive invoking: $Get\_Positive(OSet_j^c, k + 1)$
  end for
End Function

Let's analyze the time complexity and space complexity of Algorithm 1 now.

Suppose $n = |U|$, $m = |C|$, $p = max(|V_i|)(1 \leq i \leq |C|)$. Because calculating all values of k-th attribute in the set of objects $OSet$ can be performed in the time $O(n)$, the time complexity of Algorithm 1 could be approximated by the following recursive equation:

$$T(n,m) = \begin{cases} 1. & (n = 1) \\ n. & (m = 0) \\ 2n + p_1 + T(n_1, m-1) + T(n_2, m-1) + ... + T(n_k, m-1). \\ & (n_1 + n_2 + ... + n_k = n, n > 1, m > 0, p_1 \leq min(p, n)) \\ 0. & (else) \end{cases} \quad (1)$$

According to the iterative method and solution of recursive equation [3], we can find:

$$
\begin{aligned}
T(n,m) &\le (2n+n) + T(n_1, m-1) + T(n_2, m-1) + ... + T(n_k, m-1) \\
&\le 3n + T(n_1, m-1) + T(n_2, m-1) + ... + T(n_k, m-1) \\
&\le 3n + (3n_1 + T(n_1^1, m-2) + T(^1_2, m-2) + ... + T(^1_{t_1}, m-2)) \\
&\quad +(3n_2 + T(n_1^2, m-2) + T(^2_2, m-2) + ... + T(^2_{t_2}, m-2)) \\
&\quad +... \\
&\quad +(3n_k + T(n_1^k, m-2) + T(^k_2, m-2) + ... + T(^k_{t_k}, m-2)) \\
&\le 3n + 3n_1 + 3n_2 + ... + 3n_k + (T(n_1^1, m-2) + T(^1_2, m-2) + ... + T(^1_{t_1}, m-2)) \\
&\quad +(T(n_1^2, m-2) + T(^2_2, m-2) + ... + T(^2_{t_2}, m-2)) \\
&\quad +... \\
&\quad +(T(n_1^k, m-2) + T(^k_2, m-2) + ... + T(^k_{t_k}, m-2)) \\
&\le 3n + 3n + (T(n_1^1, m-2) + T(^1_2, m-2) + ... + T(^1_{t_1}, m-2)) \\
&\quad +(T(n_1^2, m-2) + T(^2_2, m-2) + ... + T(^2_{t_2}, m-2)) \\
&\quad +... \\
&\quad +(T(n_1^k, m-2) + T(^k_2, m-2) + ... + T(^k_{t_k}, m-2)) \\
&\le 3n + 3n + ... + 3n + n \\
&\le 3 \times m \times n + n
\end{aligned}
$$

That is, $T(n, m) = O(n \times m)$.

Suppose $n = |U|$, $m = |C|$, $p = max(|V_i|)(1 \le i \le |C|)$. Then, the space complexity of Algorithm 1 is: $O(n + p \times m)$.

4 Algorithm for Computing Attribute Core Based on Divide and Conquer Method

Lemma 1. Given a decision table $S =< U, A = C \cup D, V, f >$. $\forall c(c \in C)$, $U/\{c\}$ is a partition of S, that is, S is divided into $k(k = |IND(U/\{c\})|)$ sub-decision tables S_1, S_2,..., S_k. Where, $S_k =< U_k, (C-\{c\}) \cup D, V_k, f_k >$, satisfying $_{\forall x \in U_i \forall y \in U_i} c(x) = c(y)(1 \le i \le k)$ and $_{\forall x \in U_i \forall z \in U_j} c(x) \ne c(z)(1 \le i < j \le k)$. Suppose $Core_i(1 \le i \le k)$ be the attribute core of the sub-decision table S_i, and $Core$ be the attribute core of the decision table S. Then, $_{\forall a \in Core_i}\ a \in Core$.

Lemma 2. Given a decision table $S =< U, A = C \cup D, V, f >$. $\forall c(c \in C)$, which is unnecessary in C with reference to D, that is, $Pos_{C-\{c\}}(D) = Pos_C(D)$. $U/\{c\}$ is a partition of S, that is, S is divided into $k(k = |IND(U/\{c\})|)$ sub-decision tables S_1, S_2,..., S_k. Where, $S_k =< U_k, (C-\{c\}) \cup D, V_k, f_k >$, satisfying $_{\forall x \in U_i \forall y \in U_i} c(x) = c(y)(1 \le i \le k)$ and $_{\forall x \in U_i \forall z \in U_j} c(x) \ne c(z)(1 \le i < j \le k)$. Suppose $Core_i(1 \le i \le k)$ be the attribute core of the sub-decision table S_i, and $Core$ be the attribute core of the decision table S. Suppose $red_i(1 \le i \le k)$ be an attribute reduction of the sub-decision table S_i. Let $R = \bigcup_{1 \le i \le k} red_i$. Then, there are two conclusions:

(1) $Core = \bigcup_{1 \le i \le k} Core_i$. (2)In the decision table S, $Pos_R(D) = Pos_C(D)$.

Lemma 3. Given a decision table $S =< U, A = C \cup D, V, f >$. Let $Core(Core \neq \phi)$ be the attribute core of S. $\forall c(c \in Core)$, which is a core attribute (necessary attribute) of S, that is, $Pos_{C-\{c\}}(D) \neq Pos_C(D)$. $U/\{c\}$ is a partition of S, that is, S is divided into $k(k = |IND(U/\{c\})|)$ sub decision tables S_1, S_2,..., S_k. Where, $S_k =< U_k, (C - \{c\}) \cup D, V_k, f_k >$, satisfying $_{\forall x \in U_i \forall y \in U_i} c(x) = c(y)(1 \leq i \leq k)$ and $_{\forall x \in U_i \forall z \in U_j} c(x) \neq c(z)(1 \leq i < j \leq k)$. Suppose $Core_i(1 \leq i \leq k)$ be the attribute core of the sub-decision table S_i, and $red_i(1 \leq i \leq k)$ be an attribute reduction of the sub-decision table S_i. Let $R = \{c\} \cup \bigcup_{1 \leq i \leq k} red_i$. Then, there are two conclusions:

(1) $Core = \{c\} \cup \bigcup_{1 \leq i \leq k} Core_i$. (2)In the decision table S, $Pos_R(D) = Pos_C(D)$.

Theorem 2. Given a decision table $S =< U, A = C \cup D, V, f >$. $\forall c(c \in C)$, according to $U/\{c\}$, S is divided into $k(k = |IND(U/\{c\})|)$ sub-decision tables S_1, S_2,..., S_k. Where, $S_k =< U_k, (C - \{c\}) \cup D, V_k, f_k >$, satisfying $_{\forall x \in U_i \forall y \in U_i} c(x) = c(y)(1 \leq i \leq k)$ and $_{\forall x \in U_i \forall z \in U_j} c(x) \neq c(z)(1 \leq i < j \leq k)$. Suppose $Core_i(1 \leq i \leq k)$ be the attribute core of the sub decision table S_i, and $Core$ be the attribute core of the decision table S. Then, $\bigcup_{1 \leq i \leq k} Core_i \subseteq Core \subseteq \{c\} \cup \bigcup_{1 \leq i \leq k} Core_i$.

Proof: Obviously, Lemma 1, Lemma 2, Lemma 3 and Theorem 2 could be proved using basic concerts of rough set theory. We omit their proofs here due to page limits.

According to Theorem 2, an algorithm for computing attribute core based on divide and conquer could be developed.

Algorithm 2. Computing Attribute Core Based on Divide and Conquer Method
Input: A decision table $S =< U, C \cup D, V, f >$
Output: Attribute Core ($Core$) of S
Step1: (*Initiative*) $Core = \phi$;
Step2: (*Compute Attribute Core using recursive function*)
  $Get\_Core(U, 1)$;
Step3: (*Return*) return $Core$
Recursive Function Get_Core(Set $OSet$, int k)
  if $(k < 1)$ or $(|OSet| < 1)$ then return; end if
  if $(c_k \in Core)$ then return;
  else
    Suppose $C^k = c_k \cup c_{k+1} \cup ... \cup c_{|C|}$;
    For decision table $S' =< OSet, C^k \cup D, V^k, f^k >$, compute positive region $Pos_{C^k-\{c_k\}}(D)$ using Algorithm 1;
    $Pos_{C^k}(D) = \phi$;
  end if
  Let $c = c_k$, $V^c = \phi$;
  for $i = 1$ to $|OSet|$ do

$V^c = V^c \cup f(x_i, c)$;
end for
for $i = 1$ to $|V^c|$ do
$OSet_j^c = \phi$;
end for
construct a mapping function $f^{'} : V^c \rightarrow N(N = 1, 2, ..., |V^c|)$, satisfying:
$\forall x{\in}V^c \forall y{\in}V^c \ (f^{'}(x) = f^{'}(y)) \Leftrightarrow (x = y)$.
for $i = 1$ to $|OSet|$ do
let $j = f^{'}(f(x_i, c))$; $OSet_j^c = OSet_j^c \cup \{x_i\}$;
end for
for $j = 1$ to $|V^c|$ do
$Pos_{C^k}(D) = Pos_{C^k}(D) \cup \ Get\_Positive(OSet_j^c, k + 1)$;
$Get\_Core(OSet_j^c, k + 1)$;
end for
if $(Pos_{C^k - \{c\}}(D) < Pos_{C^k}(D))$ then $Core = Core \cup \{c\}$; end if
End Function

Now, let's analyze the time complexity and space complexity of Algorithm 2.

Suppose $n = |U|$, $m = |C|$. Then, the time complexity of Algorithm 2 could be approximated by the following recursive equation:

$$T(n,m) = \begin{cases} O(n \times m) + T(n_1, m-1) + T(n_2, m-1) + ... + T(n_k, m-1). \\ \qquad (n_1 + n_2 + ... + n_k = n, n > 1, m > 0) \\ 0. \quad (else) \end{cases} \tag{2}$$

According to the iterative method and solution of recursive equation [3], we can have: $T(n, m) = O(n \times m) \times m = O(n \times m^2)$.

Suppose $n = |U|$, $m = |C|$, $p = max(|V_i|)(1 \leq i \leq |C|)$. Then, the space complexity of Algorithm 2 is: $O(n + p \times m)$.

5 Experiment Results

Firstly, some data sets from UCI database are used to test Algorithms 2. Secondly, data sets KDDCUP99 are used to test the efficiency of Algorithm 2(Data sets KDDCUP99 can be downloaded at *http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html*).

5.1 Experiment Results in *UCI* Database

Data sets $Heart\_c\_ls, Pima\_India, rx\_bq\_ls, Liver\_disorder$ and $Abalone$ from UCI database (These data sets can be downloaded at http://www.ics.uci.edu) are used as test data sets. In order to compare our algorithms with existed algorithms, the algorithm in [5,7,18] and the algorithm in [19] are chosen, called Algorithm a and Algorithm b respectively. The experiment results are shown in Table 1. Where, T is running time(in second) of algorithms, and N is the cardinality of core attribute. The configuration of the PC here is P4 2.60G CPU, 256M RAM, Windows XP.

We can find from Table 1 that results of Algorithm a, Algorithm b and Algorithm 2 are valid. However, the Algorithm 2 could save some time.

Table 1. Experiment results on *UCI* database

Data Sets	*Number of Attribute*	*Number of Records*	*Algorithm a*		*Algorithm b*		*Algorithm 2*	
			T	*N*	*T*	*N*	*T*	*N*
Glass	9	214	0.016	9	0.003	9	0.001	9
Heart_c_ls	13	303	0.047	9	0.006	9	0.003	9
Australian_Credit	14	660	0.141	8	0.023	8	0.005	8
Pima_India	8	738	0.156	5	0.025	5	0.003	5
Liver_disorder	6	1260	0.063	5	0.009	5	0.005	5
Abalone	8	4177	8.031	6	1.147	6	0.041	6

5.2 Experiment Results on Data Sets KDDCUP99

In order to test the efficiency of Algorithm 2 on really huge data sets, 20 KDDCUP99 data sets are downloaded. The number of records of these data sets are 1×10^5, 2×10^5, 3×10^5,..., 20×10^5 respectively. The number of condition attributes is 41. The experiment results are shown in Fig.1. The configuration of the PC here is also P4 2.60G CPU, 256M RAM, windows XP.

We can find from Fig.1 that the efficiency of Algorithm 2 is very high on huge data sets. Besides, the time cost of our algorithm is almost linear with the number of objects. In the meantime, we test the minimum data set of Fig.1 with Algorithm a and Algorithm b, their running time are both more than 1 hour.

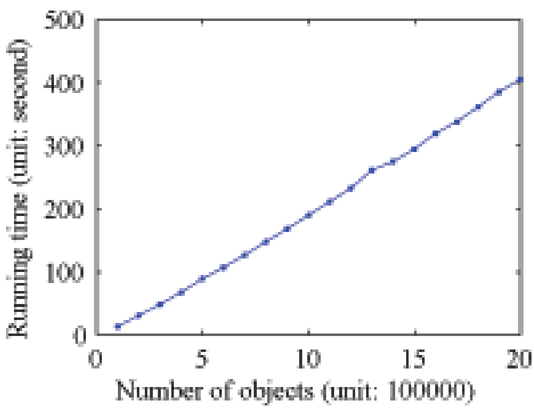

Fig. 1. Experiment results on KDD data sets

6 Conclusion

Though rough set theory is becoming more and more mature, its application in industry is still limited. An important reason is that the efficiency of many algorithms of rough set theory is too low to meet to the need of industry in huge data set environments. In this paper, the idea of divide and conquer method is used in the rough set theory, and an algorithm for computing positive region and an algorithm for computing attribute core are proposed. Experiment results show that the proposed algorithms are not only efficient, but also can deal with huge data sets. Studying on algorithms of attribute reduction and value reduction based on divide and conquer method will be our further work.

References

1. Pawlak, Z.: Rough sets. International Journal of Computer and Information Sciences 11, 341–356 (1982)
2. Wang, G.Y.: Rough set theory and knowledge acquisition (in Chinese). Xi'an Jiaotong University Press, Xi'an (2001)
3. Fu, Q.X, Wang, X.D.: Algorithms and data structure (in Chinese). Publishing House of Electronics Industry, Beijing (2003)
4. Yu, X.X, Cui, G.H, Zhou, H.M.: Fundmental of Computer Algorithms (in Chinese). Huazhong University Press, Wuhan (2001)
5. Skowron, A., Rauszer, C.: The discernibility functions matrics and functions in information systems. In: Slowinski, R. (ed.) Intelligent Decision Support - Handbook of Applications and Advances of the Rough Sets Theory, pp. 331–362. Kluwer Academic Publisher, Dordrecht (1992)
6. Bazan, J., Skowron, A., Synak, P.: Dynamic reductions as tool for exacting laws for decision table. Lecture Note in Artificial Intelligence, vol. 869, pp. 346–355. Springer, Berlin (1994)
7. Hu, X.H., Cercone, N.: Learning in relational database: A rough set approach. International Journal of Computional Intelligence 11(2), 323–338 (1995)
8. Jelonek, J., et al.: Rough set reduction of attributes and their domains for neural networks. Computional Intelligence 11(2), 338–347 (1995)
9. Ziarko, W., Shan, N.: Data-based acquisition and incremental modification classfication rules. Computational Intelligence 11(2), 357–370 (1995)
10. Nguyen, H.S, Nguyen, S.H.: Some efficient algorithms for rough set methods. In: Proceedings of the Sixth International Conference, Information Procesing and Management of Uncertainty in Knowledge-Based Systems(IPMU"96), July 1-5, Granada, Spain, vol. 2, pp. 1451–1456 (1996)
11. Susmaga, R.: Experiments in incremental computation of reducts. In: Skowron, A., Olkowski, A. (eds.) Rough Sets in Data Mining and Knowledge Discovery, Springer, Berlin (1998)
12. Bazan, B.J H.S., Nguyen, S.H., Nguyen, P.: Rough set algorithms in classification problem. In: Polkowski, L., Tsumoto, S., Lin, T.Y. (eds.) Rough Set Methods and Applications, pp. 49–88. Physica-Verlag, Heidelberg (2000)
13. Ye, D.Y., et al.: An improvement to Jelonek's attribute reduction algorithm (in Chinese). Acta. Electronica Sinica 28(12), 81–82 (2000)
14. Wang, J., Wang, J.: Reduction algorithms based on discernibility matrix: the ordered attributed method. Journal of Computer Science and Technology 11(6), 489–504 (2001)
15. Liu, S.H., Sheng, Q.J., Wu, B., et al.: Research on efficient algorithms for Rough set methods (in Chinese). Chinese Journal of Computers 30(7), 1086–1088 (2002)
16. Wang, G.Y., Yu, H., Yang, D.C.: Decision table reduction based on conditional information entropy. Chinese Journal of computer (in Chinese) 25(7), 759–766 (2002)

17. Liu, S.H., Cheng, Q.J., Shi, Z.Z.: A new method for fast computing positve region. Journal of Computer Research and Development (in Chinese) 40(5), 637–642 (2003)
18. Ye, D.Y., Chen, Z.J.: A new discernibility matrix and the computation of a core. Acta. Electronica Sinica (in Chinese) 30(7), 1086–1088 (2002)
19. Wang, G.Y.: The computation method of core attribute in decision table. Chinese Journal of Computer (in Chinese) 26(5), 611–615 (2003)