Neighbor Inconsistent Pair Selection for Attribute Reduction by Rough Set Approach

Jianhua Dai, Qinghua Hu, *Senior Member, IEEE*, Hu Hu, and Debiao Huang

***Abstract*—Rough set theory, as one of the most useful soft computing methods dealing with vague and uncertain information, has been successfully applied to many fields, and one of its main applications is to perform attribute reduction. Although many heuristic attribute reduction algorithms have been proposed within the framework of the rough set theory, these methods are still computationally time consuming. In order to overcome this deficit, we propose, in this paper, two quick feature selection algorithms based on the neighbor inconsistent pair, which can reduce the time consumed in finding a reduct. At first, we propose several concepts regarding simplified decision table(U') and neighbor inconsistent pairs. Based on neighbor inconsistent pairs, we constructed two new attribute significance measures. Furthermore, we put forward two new attribute reduction algorithms based on quick neighbor inconsistent pairs. The key characteristic of the presented algorithms is that they only need to calculate U'/R once under the process of selecting the best attribute from attribute sets: $C - R$, while most existing algorithms need to calculate partition of U' for $|C - R|$ times. In addition, the proposed algorithms need only to deal with the equivalent classes in U'/R that contain at least one neighbor inconsistent pair, while most existing algorithms need to consider all objects in U'. The experimental results show that the proposed algorithms are feasible and efficient.**

***Index Terms*—Attribute reduction, neighbor consistent pair, neighbor inconsistent pair, rough set theory.**

I. Introduction

ROUGH set theory [1], [2], introduced by Pawlak, is a useful mathematical approach to deal with vague and uncertain information. In the framework of rough set theory, an object is represented as a collection of values of the given attributes in a given database, which is represented as a table whose rows and columns stand for the objects and attributes, respectively. Usually attributes are divided into two parts: conditional attributes and decision attributes. Rough set theory has been successfully applied to solve a variety of problems [1]–[10], and it has good expansibility. For example, it can be combined with soft set theory and fuzzy set theory [11], [12] and it can also be extended to neighborhood based decision-theoretic rough set models [13].

In the applications of data mining, pattern recognition, and machine learning, many of the attributes in database are irrelevant to the decision making, which means the existence of these attributes will weaken the ability of learning algorithms. To deal with this issue, attribute reduction, also called feature selection, has been suggested as a necessary preprocessing step to find a suitable subset of attributes [14]–[30]. So far, many attribute reduction algorithms based on rough sets have been proposed in order to find one reduct or all reducts [4], [21], [31]–[37]. The existing methods basically fall into two categories: one concentrates on indiscernibility relation [38]–[40]; the other on discernibility relation [2], [41]–[43]. It should be noted that this paper does not focus on attribute reduction by introducing entropies.

For discernibility relation, there exist attribute reduction algorithms based on discernibility matrix and information entropy. Skowron and Rauszer introduced the concept of discernibility matrix with respect to decision system in [41]. Boolean discernibility function was proposed for the discernibility matrix as well. Ye and Chen proposed an improved concept of discernibility matrix in order to make it fit with inconsistent decision system in [44]. Yang and Sun improved the concept of discernibility matrix again in order to reduce the time cost [45]. Recently, Chen *et al.* proposed the concept of sample pair selection based on discernibility matrix in order to find an efficient way to get a reduct [46]. However, its time consumption is still at least $O(|U|^2|C|)$.

For indiscernibility relation, we can construct a positive region which is a collection of objects that can be certainly classified into a certain class corresponding to the selected attributes by means of the partition of the universe. Attribute reduction based on the positive region has been proposed in [39] and [40], in which the simplified decision table is proposed, and some efficient sort algorithms are proposed in order to accelerate the process to find a reduct. So far the best time complexity still reaches the maximum $(O(|C||U|), O(|C|^2|U/C|))$ in [38] and [40]. Compared with attribute reduction algorithms based on discernibility matrix, it is a huge success. However, the

Manuscript received January 29, 2017; accepted April 13, 2017. Date of publication April 26, 2017; date of current version March 29, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61473259, Grant 61432011, Grant 61070074, and Grant 60703038, in part by the National Science & Technology Support Program of China under Grant 2015BAK26B00 and Grant 2015BAK26B02, and in part by the PEIYANG Young Scholars Program of Tianjin University under Grant 2016XRX-0001. *(Corresponding author: Jianhua Dai.)*

J. Dai is with the School of Computer Science and Technology, Tianjin University, Tianjin 300350, China, and also with the Key Laboratory of High Performance Computing and Stochastic Information Processing (Ministry of Education of China), Hunan Normal University, Changsha 410081, China (e-mail: david.joshua@qq.com).

Q. Hu is with the School of Computer Science and Technology, Tianjin University, Tianjin 300350, China (e-mail: huqinghua@tju.edu.cn).

H. Hu and D. Huang are with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China (e-mail: 2420967879@qq.com; 1169196994@qq.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2017.2698420

algorithms based on the positive region are very sensitive to the number of attributes, because these algorithms have to calculate sigPos value for the remaining attributes, respectively. In other words, when algorithm based on the positive region needs to select a best attribute, it has to calculate sigPos $|C-R|$ times, which means we have to compute positive region $|C-R|$ times as well. Apparently, it suffers an inherent drawback of high time cost, especially when the data contain a large number of attributes.

In order to reduce the time cost, a quick neighbor inconsistent pair selection strategy is proposed as a more efficient way to find a suitable reduct in this paper. First of all, we construct the basic concepts of the neighbor inconsistent pair selection based on rough set theory. Then, we put forward two new suitable attribute significance measures based on the neighbor inconsistent pair selection. Finally, we propose two new kinds of heuristic algorithms to find a proper reduct based on the quick neighbor inconsistent pair selection. Compared with existing algorithms based on the positive region, the key characteristic of this approach is that we only need to calculate $U^{'}/R$ once in the process of selecting the best attribute from attribute sets: $C-R$, while most existing algorithms need to calculate partition of $U^{'}$ for $|C-R|$ times. What's more, the proposed algorithms need only to deal with the equivalent classes in $U^{'}/R$ that contain at least one neighbor inconsistent pair, i.e., part of objects in $U^{'}$, while most existing algorithms need to consider all objects in $U^{'}$.

The rest of this paper is organized as follows. Some related basic notions are reviewed in Section II. The definitions and concepts of neighbor inconsistent pair are proposed in Section III. The new attribute significance measures and attribute reduction algorithms based on the quick neighbor inconsistent pair selection are presented in Section IV. The experiments to validate the efficiency and effectiveness of the proposed algorithm are conducted in Section V. Section VI concludes the whole paper.

II. Basic Notions

In this section, we briefly review the basic notions about rough sets and decision tables, which can be found in [1], [2], [38], and [40].

A decision table (also referred to as an information system) is defined as $S=<U,A,V,f>$, where $U=\{u_1,u_2,\ldots,u_n\}$ is a finite nonempty set of objects; $A=C\cup D$ is a finite nonempty set of attributes, where $C=\{c_1,c_2,\ldots,c_m\}$ is a nonempty set of conditional attributes, and D is a nonempty set of decision attributes (generally, $D=\{d\}$), $C\cap D=\emptyset$. V is a union of the value domains, i.e., $V=\cup_{a\in A}V_a$, where V_a is the value set of attribute a, called the value domain of attribute a; and $f=U\times A\rightarrow V$ is an information function, which maps an object in U to exactly one value from domains of attributes such as $\forall a\in A, x\in U$, and $f(x,a)\in V_a$. The value of attribute a for object x denotes by $f(x,a)$.

Given a decision table $S=<U,A,V,f>$, for any subset of attributes $B\subseteq A$, the indiscernibility relation generated by B on U is defined by

$$\mathrm{IND}(B)=\{(x,y)\in U^2|\forall b\in B, b(x)=b(y)\}. \quad (1)$$

It is clear that $\mathrm{IND}(B)$ is an equivalence relation, which is reflexive, symmetric, and transitive. And it determines a partition of U, denoted by $U/\mathrm{IND}(B)$ or simply U/B; an equivalence class of $\mathrm{IND}(B)$ containing x will be denoted by $[x]_B$.

For any $X\subseteq U$, the lower approximation of X with respect to B can be further defined as

$$\underline{\mathrm{apr}}_B X=\{x|[x]_B\subseteq X\}. \quad (2)$$

Let P and Q be equivalence relations over U, then the concepts of positive regions can be defined as

$$\mathrm{POS}_P Q=\bigcup_{X\in U/Q}\underline{\mathrm{apr}}_P X. \quad (3)$$

Let S be a decision table, for $x,y\in U$, if for any $a\in C$, $f(x,a)=f(y,a)$, and $f(x,d)\neq f(y,d)$, then x and y are inconsistent in U, otherwise they are consistent.

Definition 1 ([40]): Given a decision table $S=\{U,A,V,f\}$, where $A=C\cup D$. $U/C=\{[\mu_1{'}]_C,[\mu_2{'}]_C,\ldots,[\mu_m{'}]_C\}$, $U^{'}=\{\mu_1{'},\mu_2{'},\ldots,\mu_m{'}\}$, $U^{'}_{\mathrm{POS}}=\{\mu^{'}_{i_1},\mu^{'}_{i_2},\ldots,\mu^{'}_{i_t}\}$, all the objects in $U^{'}_{\mathrm{POS}}$ are consistent, all the objects in $U^{'}-U^{'}_{\mathrm{pos}}$, denoted as $U^{'}_{\mathrm{BND}}$, are inconsistent in the original decision table S, then $S^{'}=\{U^{'},A,V,f\}$ is called a simplified decision table.

Note that, all objects in $U^{'}_{\mathrm{BND}}$ are inconsistent, for computing efficiently, we denote the decision value of all objects of $U^{'}_{\mathrm{BND}}$ by $\mathrm{maxD}+1$, where maxD means the maximal value of decision attribute domain. In this way, we can change an inconsistent table into a consistent one without affecting the attribute reduction result of an original decision table in Pawlak rough set model. In what follows, we always discuss the $S^{'}$ as the processed consistent simplified decision table. For any simplified decision table $S^{'}=\{U^{'},A,V,f\}$, it has several important properties:

1) Suppose $S^{'}$ is the simplified decision table from the original S. If S is consistent, then $S=S^{'}$.
2) $\forall x_i,x_j\in U^{'}$, $\exists a\in C$, $f(x_i,a)\neq f(x_j,a)$.
3) $\forall x_i,x_j\in U^{'}$, object pair (x_i,x_j) is consistent.
4) $\mathrm{POS}_C D=|U^{'}|$.

TABLE I
Inconsistent Decision Table

U	c_1	c_2	d
u_1	1	1	3
u_2	2	1	4
u_3	3	1	1
u_4	3	3	2
u_5	3	3	2
u_6	2	1	2
u_7	3	1	2

Example 1: Table I shows a decision table, where $U=\{u_1,u_2,u_3,\ldots,u_7\}, C=\{c_1,c_2\}$, and $D=\{d\}$. Then, we get $U/C=\{\{u_1\},\{u_2,u_6\},\{u_3,u_7\},\{u_4,u_5\}\}$. Thus, we have $U'=\{u_1,u_{2,6},u_{3,7},u_{4,5}\}=\{x_1,x_2,x_3,x_4\}$ and $U^{'}_{\mathrm{POS}}=\{u_1,u_{4,5}\}=\{x_1,x_4\}$. At the same time, $U^{'}_{\mathrm{BND}}=U^{'}-U^{'}_{\mathrm{pos}}$

TABLE II
SIMPLE DECISION TABLE GENERATED FROM TABLE I

U'	c_1	c_2	d
x_1	1	1	3
x_2	2	1	5
x_3	3	1	5
x_4	3	3	2

TABLE III
SIMPLE DECISION TABLE

U	c_1	c_2	c_3	c_4	c_5	d
x_1	1	1	2	1	3	3
x_2	2	1	1	1	3	4
x_3	3	1	2	2	2	1
x_4	3	3	3	3	2	2
x_5	3	3	3	3	3	2
x_6	2	1	1	2	1	2
x_7	3	1	3	1	3	2

$= \{u_{2,6}, u_{3,7}\} = \{x_2, x_3\}$. Therefore, we get the simplified decision table shown in Table II.

III. NEIGHBOR INCONSISTENT PAIR SELECTION BASED ON ROUGH SET THEORY

In this section, we propose some basic definitions of the neighbor inconsistent pair selection. Note that, in this paper, we define the concepts about neighbor pair selection based on the simplified decision table serving as the theoretical foundation of our definitions.

Definition 2: Given a simplified decision table $S' = \{U', C \cup D, V, f\}$, $\forall R \subseteq C$, then we use U''_R to represent the sorted U' based on equivalence relation U'/R. Objects in the same equivalent class are viewed as neighbors in U''_R, which is defined as $U''_R = U'//R$.

For example, in Table III, let $R = \{c_3\}$, $U'/R = \{\{x1, x3\}, \{x2, x6\}, \{x4, x5, x7\}\}$, we get $U''_R = \{x1, x3, x2, x6, x4, x5, x7\}$. Compared with U', U''_R contains the same objects but reorders the objects according to the equivalent classes.

Definition 3: Given a simplified decision table $S' = \{U', C \cup D, V, f\}$, let $U' = \{x_1, x_2, \ldots, x_n\}$, then a neighbor pair denotes the objects in the pair as adjacent, such as the object pair (x_1, x_2). $\forall i \in [1, |U'|)$, and the object pair (x_i, x_{i+1}) is a neighbor pair as well.

Definition 4: Given a simplified decision table $S' = \{U', C \cup D, V, f\}$, for any subset of conditional attributes $R \subseteq C$, there are two neighbor relations on U''_R, defined as follows:

Neighbor inconsistent relation:

$$\mathrm{NIP}_R D = \{(x_i, x_{i+1}) \in U''_R \times U''_R | 1 \le i < |U''_R| \wedge (f(R, x_i) = f(R, x_{i+1}) \wedge (f(D, x_i) \neq f(D, x_{i+1})\}. \quad (4)$$

Neighbor consistent relation:

$$\mathrm{NCP}_R D = \{(x_i, x_{i+1}) \in U''_R \times U''_R | 1 \le i < |U''_R| \wedge (x_i, x_{i+1}) \notin \mathrm{NIP}_R D\}. \quad (5)$$

It is obvious that $\mathrm{NIP}_R D$ is the collection of neighbor object pairs, which are inconsistent in U''_R and $\mathrm{NCP}_R D$ is the collection of consistent neighbor object pairs. Note that, $\mathrm{NIP}_R D$ and $\mathrm{NCP}_R D$ only care about the object pairs in U''_R that are adjacent and we denote the number of total object pairs in $\mathrm{NIP}_R D$ by $|\mathrm{NIP}_R D|$.

Definition 5: Given a simplified decision table $S' = \{U', C \cup D, V, f\}$, $\forall R \subseteq C$, $\forall a \in C - R$, then a binary relation called neighbor inconsistent relation with respect to attribute a on U''_R is defined by

$$\mathrm{NIP}_R D\{a\} = \{(x_i, x_{i+1}) \in U''_R \times U''_R | (x_i, x_{i+1}) \in \mathrm{NIP}_R D \wedge f(x_i, a) \neq f(x_{i+1}, a)\}. \quad (6)$$

It's clear that $\mathrm{NIP}_R D\{a\}$ is the collection of pairs in $\mathrm{NIP}_R D$, in which objects have different values on attribute a.

Example 2: A decision table $S' = (U', C \cup D, V, f)$ is given by Table III, where $U' = \{x_1, x_2, \ldots, x_7\}$, $C = \{c_1, c_2, \ldots, c_5\}$, $D = \{d\}$. Let $R = \{c_5\}$, then $U'/R = \{\{x_1, x_2, x_5, x_7\}, \{x_3, x_4\}, \{x_6\}\}$, then we get $U''_R = \{x_1, x_2, x_5, x_7, x_3, x_4, x_6\}$. Consequ ntly, we have

$\mathrm{NIP}_R D = \{(x_1, x_2), (x_2, x_5), (x_3, x_4)\}$, $|\mathrm{NIP}_R D|$=3;
$\mathrm{NCP}_R D = \{(x_5, x_7), (x_7, x_3), (x_4, x_6)\}$, $|\mathrm{NCP}_R D|$=3;
$\mathrm{NIP}_R D\{c_1\} = \{(x_1, x_2), (x_2, x_5)\}$, $|\mathrm{NIP}_R D\{c_1\}|$=2;
$\mathrm{NIP}_R D\{c_2\} = \{(x_2, x_5), (x_3, x_4)\}$, $|\mathrm{NIP}_R D\{c_2\}|$=2;
$\mathrm{NIP}_R D\{c_3\} = \{(x_1, x2), (x_2, x_5), (x_3, x_4)\}$, $|\mathrm{NIP}_R D\{c_3\}| = 3$;
$\mathrm{NIP}_R D\{c_4\} = \{(x_2, x_5), (x_3, x_4)\}$, $|\mathrm{NIP}_R D\{c_4\}|$=2.

Theorem 1: Given a simplified decision table $S' = \{U', C \cup D, V, f\}$, $\forall R \subseteq C$ and $\forall (x_i, x_{i+1}) \in U''_R \times U''_R$, $1 \le i < |U''_R|$, then $(x_i, x_{i+1}) \in \mathrm{NIP}_R D$ or $(x_i, x_{i+1}) \in \mathrm{NCP}_R D$.

Proof: Given a certain U' and $R \subseteq C$, $\forall i \in [1, |U'|)$, the neighbor object pairs can be divided into two cases:

1) if $[x_i]_R \neq [x_{i+1}]_R$, object pair $(x_i, x_{i+1}) \in \mathrm{NCP}_R D$.
2) if $[x_i]_R = [x_{i+1}]_R$, then we can divide it into two situations:
   a) if $f(x_i, D) \neq f(x_{i+1}, D)$, according to Definition 4, object pair $(x_i, x_{i+1}) \in \mathrm{NIP}_R D$;
   b) if $f(x_i, D) = f(x_{i+1}, D)$, according to Definition 4, $(x_i, x_{i+1}) \notin \mathrm{NIP}_R D$, $(x_i, x_{i+1}) \in \mathrm{NCP}_R D$.

In summary, the theorem holds. ■

Theorem 2: Given a simplified decision table $S' = \{U', C \cup D, V, f\}$, $\forall R \subseteq C$, $i \in [1, |U'|]$, let $[x_i]_R = \{x_{i_1}, x_{i_2}, \ldots, x_{i_r}\}$, then $[x_i]_R$ is consistent $\Leftrightarrow (x_{i_j}, x_{i_{j+1}}) \in \mathrm{NCP}_R D, \forall j \in [1, r)$.

Proof: ($\Rightarrow$) Suppose that $[x_i]_R$ is consistent $\Rightarrow \forall m, n \in [1, r], m \neq n$. Object pair (x_{i_m}, x_{i_n}) is consistent $\Rightarrow \forall j \in [1, r)$, object pair $(x_{i_j}, x_{i_{j+1}})$ is consistent $\Rightarrow$ object pair $(x_{i_j}, x_{i_{j+1}}) \notin \mathrm{NIP}_R D \Rightarrow (x_{i_j}, x_{i_{j+1}}) \in \mathrm{NCP}_R D$.

($\Leftarrow$) Suppose that $\forall j \in [1, r)$, object pair$(x_{i_j}, x_{i_{j+1}}) \in \mathrm{NCP}_R D \Rightarrow \forall j \in [1, r)$, object pair$(x_{i_j}, x_{i_{j+1}}) \notin \mathrm{NIP}_R D$, since $[x_{i_j}]_R = [x_{i_{j+1}}]_R \Rightarrow \forall j \in [1, r), f(x_{i_j}, D) =$

$f(x_{i_{j+1}}, D) \Rightarrow \forall m, n \in [1, r]$, and $m \neq n$, $f(x_{i_m}, D) = f(x_{i_n}, D) \Rightarrow [x_i]_R$ is consistent. ■

Example 3: As for the decision table given by Table III, where $U^{'} = \{x_1, x_2, \ldots, x_7\}$, $C = \{c_1, c_2, \ldots, c_5\}$, $D = \{d\}$. Let $R = \{c_1, c_3\}$, then $U^{'}/R = \{\{x_1\}, \{x_2, x_6\}, \{x_3\}, \{x_4, x_5, x_7\}\}$. Since $\{x_4, x_5, x_7\}$ is consistent, we know $(x_4, x_5), (x_5, x_7) \in \mathrm{NCP}_R D$.

Based on Theorems 1 and 2, we can get the conclusion that once $[x_i]_R$ is consistent, there will be no neighbor pairs of $[x_i]_R$ belonging to $\mathrm{NIP}_R D$.

Theorem 3: Given a simplified decision table $S^{'} = \{U^{'}, C \cup D, V, f\}$, $\forall R \subseteq C$, $i \in [1, |U^{'}|]$, let $[x_i]_R = \{x_{i_1}, x_{i_2}, \ldots, x_{i_r}\}$, then $[x_i]_R$ is inconsistent $\Leftrightarrow \exists j \in [1, r)$, object pair $(x_{i_j}, x_{i_{j+1}}) \in \mathrm{NIP}_R D$.

Proof: $\forall x_i \in U^{'}$, let $[x_i]_R = \{x_{i_1}, x_{i_2}, \ldots, x_{i_r}\}$.

Suppose that $[x_i]_R$ is inconsistent, then $\exists x_{i_m}, x_{i,n} \in [x_i]_R$, $\forall a \in R$, $f(x_{i_m}, a) = f(x_{i_n}, a)$, $f(x_{i_m}, D) \neq f(x_{i_m}, D)$. Suppose that $\forall j \in [1, r)$, object pair $(x_{i_j}, x_{i_{j+1}}) \notin \mathrm{NIP}_R D$, then $\forall j \in [1, r)$, object pair $(x_{i_j}, x_{i_{j+1}}) \in \mathrm{NCP}_R D$. According to Theorem 2, $[x_i]_R$ is consistent. It results in a contradiction, i.e., $\exists j \in [1, r)$, object pair $(x_{i_j}, x_{i_{j+1}}) \in \mathrm{NIP}_R D$.

Suppose that $\exists j \in [1, r)$, object pair $(x_{i_j}, x_{i_{j+1}}) \in \mathrm{NIP}_R D$. Then $\exists j \in [1, r)$, object pair $(x_{i_j}, x_{i_{j+1}})$ is inconsistent $\Rightarrow [x_i]_R$ is inconsistent. ■

Suppose $[x_i]_R$ is inconsistent, by Theorem 3, we know there exists a neighbor pair of $[x_i]_R$ belonging to $\mathrm{NIP}_R D$.

Theorem 4: Given a simplified decision table $S^{'} = \{U^{'}, C \cup D, V, f\}$, then $|\mathrm{NIP}_R D| = 0 \Leftrightarrow |\mathrm{POS}_R D| = |\mathrm{POS}_C D|$.

Proof: $|\mathrm{NIP}_R D| = 0 \Leftrightarrow |\mathrm{NCP}_R D| + 1 = |U^{'}| \Leftrightarrow \forall R \subseteq C$, $i \in [1, |U^{'}|]$, let $[x_i]_R = \{x_{i_1}, x_{i_2}, \ldots, x_{i_r}\}$, then $\forall j \in [1, r)$, object pair$(x_{i_j}, x_{i_{j+1}}) \in \mathrm{NCP}_R D \Leftrightarrow \forall R \subseteq C$, $i \in [1, |U^{'}|]$, $[x_i]_R$ is consistent $\Leftrightarrow \forall m, n \in [1, |U^{'}|]$, object pair $(x_m, x_n) \in U_R^{''} \times U_R^{''}$, then object pair (x_m, x_n) is consistent $\Leftrightarrow |\mathrm{POS}_R D| = |\mathrm{POS}_C D|$. ■

According to Theorem 4, it is clear that once $\forall [x_i]_R \in U^{'}/R$ is consistent, then $|\mathrm{NIP}_R D| = 0$. In other words, $|\mathrm{NIP}_R D| = 0$ can be a new mark denoting that the attribute subset R is a reduct candidate.

Example 4: As for the decision table given by Table III, where $U^{'} = \{x_1, x_2, \ldots, x_7\}$, $C = \{c_1, c_2, \ldots, c_5\}$, $D = \{d\}$. Let $R = \{c_1, c_4\}$, then $U^{'}/R = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4, X_5\}, \{x_6\}, \{X_7\}\}$. Consequently, we have $U_R^{''} = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$. Then, $\mathrm{NIP}_R D = \emptyset$. It means $|\mathrm{NIP}_R D| = 0$. At the same time, it is easy to know that $|\mathrm{POS}_R D| = 7$.

Note that, for any simplified decision table $S^{'} = \{U^{'}, C \cup D, V, f\}$, $\mathrm{POS}_C D = |U^{'}|$.

Theorem 5: Given a simplified decision table $S^{'} = \{U^{'}, C \cup D, V, f\}$, $\forall R \subseteq C$, then $|\mathrm{NIP}_R D| \neq 0 \Leftrightarrow |\mathrm{POS}_R D| < |\mathrm{POS}_C D|$.

Proof: First, we know that $|\mathrm{NIP}_R D| \geq 0$, so $|\mathrm{NIP}_R D| \neq 0 \Leftrightarrow |\mathrm{NIP}_R D| > 0$.

($\Rightarrow$) Suppose that $|\mathrm{NIP}_R D| > 0$. Then $\exists x_i, x_j \in U_R^{''}$, object pair $(x_i, x_j) \in \mathrm{NIP}_R D \Rightarrow \exists x_i, x_j \in U^{'}$, $[x_i]_R = [x_j]_R$, and $f(x_i, D) \neq f(x_j, D) \Rightarrow \exists x_i \in U^{'}$, $[x_i]_R \notin \mathrm{POS}_R D \Rightarrow |\mathrm{POS}_R D| \leq |\mathrm{POS}_C D| - |[x_i]_R| < |\mathrm{POS}_C D|$.

($\Leftarrow$) Suppose that $|\mathrm{POS}_R D| < |\mathrm{POS}_C D|$. Then $\exists x_i \in U^{'}$, $[x_i]_R \notin \mathrm{POS}_R D \Rightarrow \exists x_i \in U^{'}$, $[x_i]_R$ is inconsistent. According to Theorem 3, we have $\exists j \in [1, r)$, object pair$(x_{i_j}, x_{i_{j+1}}) \in \mathrm{NIP}_R D \Rightarrow |\mathrm{NIP}_R D| > 0$.

In summary, the theorem holds. ■

Based on Theorem 5, we can be sure that $|\mathrm{NIP}_R D| \neq 0$ means that R is not a reduct candidate.

Theorem 6: Given a simplified decision table $S^{'} = \{U^{'}, C \cup D, V, f\}$ and a certain subset of conditional attributes $R \subset C$, then $\forall a \in (C - R)$, $|\mathrm{NIP}_R D\{a\}| \leq |\mathrm{NIP}_R D|$.

Proof: According to Definition 5, $\forall a \in R$, $\mathrm{NIP}_R D\{a\} \subseteq \mathrm{NIP}_R D \Rightarrow \forall a \in R$, $|\mathrm{NIP}_R D\{a\}| \leq |\mathrm{NIP}_R D|$. ■

By means of Theorem 6, we can easily classify the relationship between $\mathrm{NIP}_R D\{a\}$ and $\mathrm{NIP}_R D$.

Theorem 7: Given a simplified decision table $S^{'} = \{U^{'}, C \cup D, V, f\}$ and a certain subset of conditional attributes $R \subset C$, then $\forall a \in (C - R)$, $|\mathrm{NIP}_R D\{a\}| = 0 \Leftrightarrow |\mathrm{NIP}_R D| = 0$.

Proof: Suppose that $\forall a \in (C - R)$, $|\mathrm{NIP}_R D\{a\}| = 0$, then it can be divided into two cases:

If $|\mathrm{NIP}_R D| = 0$, then the theorem holds.

If $|\mathrm{NIP}_R D| > 0$, then for $\forall x_i, x_j \in U^{'}$, $(x_i, x_j) \in \mathrm{NIP}_R D$, and $\forall a \in (C - R)$, it has $f(x_i, a) = f(x_j, a) \Rightarrow$ for $\forall x_i, x_j \in U^{'}$, $(x_i, x_j) \in \mathrm{NIP}_R D$, and $\forall a \in C$, it has $f(x_i, a) = f(x_j, a)$. $\Rightarrow \exists x_i, x_j \in U^{'}$, and $(x_i, x_j) \in \mathrm{NIP}_R D$, $\forall a \in C, f(x_i, a) = f(x_j, a)$. However, since $\forall x_i, x_j \in U^{'}$, $\exists a \in C$, $f(x_i, a) \neq f(x_j, a)$. This causes a contradiction. Hence, $|\mathrm{NIP}_R D| = 0$.

Suppose that $|\mathrm{NIP}_R D| = 0$, then $\mathrm{NIP}_R D = \emptyset \Rightarrow \forall a \in (C - R)$, $\mathrm{NIP}_R D\{a\} = \emptyset \Rightarrow \forall a \in (C - R)$, $|\mathrm{NIP}_R D\{a\}| = 0$.

In summary, the theorem holds. ■

Theorem 8: Given a simplified decision table $S^{'} = \{U^{'}, C \cup D, V, f\}$ and two subsets of conditional attributes $R_1 \subset R_2 \subseteq C$, then $|\mathrm{NIP}_{R_2} D| \leq |\mathrm{NIP}_{R_1} D|$.

Proof: Suppose $(x_i, x_{i+1}) \in \mathrm{NIP}_{R_2} D$, $1 \leq i \leq |U^{''}|$. According to Definition 4, we know $(x_i, x_{i+1}) \in \mathrm{NIP}_{R_1} D$. Thus, $\mathrm{NIP}_{R_2} D \subseteq \mathrm{NIP}_{R_1} D$. It means $|\mathrm{NIP}_{R_2} D| \leq |\mathrm{NIP}_{R_1} D|$. ■

Theorem 9: Given a simplified decision table $S^{'} = \{U^{'}, C \cup D, V, f\}$ and two subsets of conditional attributes $R_1 \subset R_2 \subset C$, then $\forall a \in (C - R_2)$, $|\mathrm{NIP}_{R_2} D\{a\}| \leq |U^{'}| - |\mathrm{POS}_{R_1} D\{a\}|$.

Proof: Suppose $(x_i, x_{i+1}) \in \mathrm{NIP}_{R_2} D\{a\}$, $1 \leq i \leq |U^{''}|$. According to Definition 5, we know $(x_i, x_{i+1}) \in \mathrm{NIP}_{R_1} D\{a\}$. Thus, $\mathrm{NIP}_{R_2} D\{a\} \subseteq \mathrm{NIP}_{R_1} D\{a\}$. It means $|\mathrm{NIP}_{R_2} D\{a\}| \leq |\mathrm{NIP}_{R_1} D\{a\}|$. ■

By Theorems 7 and 9, we find that $|\mathrm{NIP}_R D\{a\}|$ has the potential to be a good measure for attribute significance.

Theorem 10: Given a simplified decision table $S^{'} = \{U^{'}, C \cup D, V, f\}$, $\forall R \subseteq C$, $\forall a \in C$, if $|\mathrm{NIP}_R D| = 0$ and $|\mathrm{POS}_{C-a} D| < |\mathrm{POS}_C D|$, then $a \in R$.

Proof: Suppose that $\exists a \in C$ satisfying $|\mathrm{POS}_{C-a} D| < |\mathrm{POS}_C D|$, and $a \notin R$. Then $|\mathrm{POS}_R D| < |\mathrm{POS}_C D|$. By Theorem 4, $|\mathrm{NIP}_R D| = 0 \Rightarrow |\mathrm{POS}_R D| = |\mathrm{POS}_C D|$. It causes a contradiction. Thus, the theorem holds. ■

By Theorem 10, it is clear that if $\exists a \in C$ belongs to core attributes of a simplified decision table, then $a \in R$ when $|\mathrm{NIP}_R D| = 0$.

IV. Quick Neighbor Inconsistent Pair Selection Based on Rough Set Theory

A. Attribute Significance Measurement by Neighbor Inconsistent Pair

In this section, we use neighbor inconsistent pair to measure attribute significance, so we only consider the neighbor inconsistent pairs in U''_R (R is an intermediate reduct). Here, two methods of measuring the attribute significance are proposed and analyzed in detail.

Definition 6: Given a simplified decision table S', let $R \subseteq C$, and $a \in (C-R)$. Then the first type of significance of attribute a is defined by

$$\mathrm{sig}_{\mathrm{NIP}}(R,a,D) = |\mathrm{NIP}_R D\{a\}|. \tag{7}$$

$\mathrm{sig}_{\mathrm{NIP}}\{R,a,D\}$ can be viewed as a measure of attribute importance based on the ability for an attribute $a \in (C-R)$ to make a neighbor inconsistent pair into a neighbor consistent pair on the object array of U'/R. Hence, if $\exists a' \in (C-R), \mathrm{sig}_{\mathrm{NIP}}(R,a',D) \geq \mathrm{sig}_{\mathrm{NIP}}(R,a,D), \forall a \in (C-R)$, then we can say attribute a' is the most important attribute.

However, $\mathrm{sig}_{\mathrm{NIP}}(R,a,D)$ has one small defect. Let us take object pair (x_1,x_2) and (x_4,x_5) in Table III as an example. Object pair (x_1,x_2) has two attributes, i.e., c_1 and c_3 with different attribute values. In this situation, both $\mathrm{sig}_{\mathrm{NIP}}(R,a,c_1)$ and $\mathrm{sig}_{\mathrm{NIP}}(R,a,c_3)$ will be added with 1. As for (x_4,x_5), there is only c_5, so $\mathrm{sig}_{\mathrm{NIP}}(R,a,c_5)$ will be added with 1 as well. However, we usually believe c_5 is more important than c_1 or c_3 from the viewpoint of Skowron's discernibility matrix. To overcome this shortcoming, we put forward another attribute significance measure in the following.

Definition 7: Given a simplified decision table S', let $R \subseteq C$, and $a \in (C-R)$. Then the second type of significance of attribute a is defined by

$$\mathrm{diff}(x_i,x_j,a) = \begin{cases} 0, \ if \ f(x_i,a) == f(x_j,a) \\ \frac{1}{|\{c|f(x_i,c)\neq f(x_j,c),\forall_c \in C\}|}, \mathrm{else} \end{cases} \tag{8}$$

$$\mathrm{sig}_{\mathrm{NIP2}}(R,a,D) = \sum_{i=1}^{N-1} \mathrm{diff}(x_i,x_{i+1},a). \tag{9}$$

If $\exists a' \in (R-C), \ \mathrm{sig}_{\mathrm{NIP2}}(R,a',D) \geq \mathrm{sig}_{\mathrm{NIP2}}(R,a,D), \forall a \in (C-R)$, then attribute a' is the most important attribute.

$\mathrm{sig}_{\mathrm{NIP2}}(R,a,D)$ is a weighted version of $\mathrm{sig}_{\mathrm{NIP}}(R,a,D)$, the difference between them is that $\mathrm{sig}_{\mathrm{NIP}}(R,a,D)$ is always added with 1, but $\mathrm{sig}_{\mathrm{NIP2}}(R,a,D)$ is always added with $1/\mathrm{diff}(x_i,x_j,a)$. In this way, we can measure the attribute significance in a more accurate way.

Attribute significance measures based on neighbor inconsistent pair are different from attribute significance measures based on positive region in the following three aspects:

1) The neighbor inconsistent pair only cares about the neighbor pair, in which the two objects belong to the same equivalent class. In other words, it focuses on the neighbor pair in an equivalent class. However, the positive region pays attention to all the object pairs that are in the same equivalent class. In other words, the positive region focuses on the equivalent class, rather than the object pairs.
2) For the attribute significance measure based on the neighbor inconsistent pair, since $\mathrm{sig}_{\mathrm{NIP}}(R,a,D)$ and $\mathrm{sig}_{\mathrm{NIP2}}(R,a,D), \ \forall a \in (C-R)$ are based on U'/R, we can get all $\mathrm{sig}_{\mathrm{NIP}}(R,a,D)$ or $\mathrm{sig}_{\mathrm{NIP2}}(R,a,D)$ for $\forall a \in (C-R)$ by calculating U'/R only once in the process of selecting the best attribute from attribute sets: $C-R$ (see Algorithms 6 or 7). On the contrary, for the attribute significance measure based on the positive region, one need to calculate $U'/\{R,a\}$ when computing $\mathrm{sig}_{\mathrm{POS}}(R,a,D)$, for $\forall a \in (C-R)$. Hence, one need to compute by $|C-R|$ times when getting all $\mathrm{sig}_{\mathrm{POS}}(R,a,D)$ for $\forall a \in (C-R)$. In other words, the algorithm based on $\mathrm{sig}_{\mathrm{NIP}}(R,a,D)$ or $\mathrm{sig}_{\mathrm{NIP2}}(R,a,D)$ is likely to be much quicker than the algorithm based on $\mathrm{sig}_{\mathrm{POS}}(R,a,D)$.
3) From the discussions about the concepts of the neighbor inconsistent pair, we can draw a conclusion that as the subset of conditional attributes $R \subseteq C$ contains more conditional attributes, $|\mathrm{NIP}_R D|$ becomes increasingly smaller until $|\mathrm{NIP}_R D| = 0$. In other words, the equivalence classes covered by selected attributes will be neglected during the attribute selection procedure. In contrast, $|\mathrm{POS}_R D|$ becomes increasingly larger until $|\mathrm{POS}_R D| = |U'|$. In each iteration, the positive region needs to be searched in the whole universe U.

B. Algorithms Based on Quick Neighbor Inconsistent Pair Selection

In this section, we expound on attribute reduction algorithms based on the neighbor inconsistent pair selection. In the following, we first introduce a quick method that can change a decision table into a simplified one (see Algorithm 1). Then an improved counting sort algorithm for a single attribute (see Algorithm 2) is discussed. Consequently, we propose an improved algorithm to calculate $\mathrm{sig}_{\mathrm{NIP}}(R,a,D)$ (see Algorithm 3) as well as an algorithm to calculate $\mathrm{sig}_{\mathrm{NIP2}}(R,a,D)$ (see Algorithm 4). An algorithm is put forward to test whether the initial reduct is necessary or not (see Algorithm 5). Finally, we present our algorithms for attribute reduction (see Algorithms 6 and 7).

We show the detailed process of the algorithms as follows:

In this paper, in order to make our algorithm adaptable to the inconsistent decision table, we have to transform the original decision table into a consistent one. The process of the transition is shown in the Algorithm 1, which can be divided into two stages as follows:

1) The first stage is from steps 1 to 15. Initialize parameters MaxC and MinC, and sort U with counting sort algorithm to make the objects in the same equivalent class be neighbors.
2) The second stage is from steps 16–35. If the near neighbor pair is inconsistent, then execute step 26 to transform it into a consistent neighbor pair.

Algorithm 1: Getting simplified decision table.

Input: A decision table $S = (U, C \cup D, V, f)$, where $U = \{x_1, x_2, \ldots, x_n\}$, $C = \{c_1, c_2, \ldots, c_m\}$, $D = \{d\}$.
Output: A simplified decision table $S^{'} = (U^{'}, C \cup D, V, f)$, where $U^{'} = \{x_1^{'}, x_2^{'}, \ldots, x_r^{'}\}$.

Let $\mathrm{MaxC}_i = \max(f(x_j, c_i))$, $\mathrm{MinC}_i = \min(f(x_j, c_i))$ for $\forall x_j \in U$, where $i \in [1, m]$. $\mathrm{Seq} = \{1, 2, \ldots, n\}$.
for i = 1 to m **do**
  for j = MinC_i to MaxC_i **do**
    {Count[j]=0;}
  end for
  for j = 1 to n **do**
    {Count[$f(x_j, c_i)$]=Count[$f(x_j, c_i)$] + 1;
    TempSeq[j]=Seq[j];}
  end for
  for k = MinC_i to MaxC_i **do**
    {Count[k]=Count[k]\,+\, Count[k-1];}
  end for
  for j = n to 1 **do**
    {Seq[Count[$f(x_{\mathrm{TempSeq}[j]}, c_i)$]] = TempSeq[j];
    Count[$f(x_{\mathrm{TempSeq}[j]}, c_i)$]=Count[$f(x_{\mathrm{TempSeq}[j]}, c_i)$]-1;}
  end for
end for
i = 2, inconsFlag = 0, r =0;
while $i \leq n$ **do**
  startPos = i-1;
  while $f(x_{\mathrm{Seq}(i)}, c_j) = f(x_{\mathrm{Seq}(i-1)}, c_j)$ for all $c_j \in C$, **do**
    if $f(x_{\mathrm{Seq}(i)}, D) \neq f(x_{\mathrm{Seq}(i-1)}, D)$,
    {inconsFlag = 1;}
    i= i + 1;
    if $i > n$ **then**
      break;
    end if
  end while
  if inconsFlag = 1 **then**
    {inconsFlag = 0;
    r = r + 1;
    $x_r^{'} = x_{\mathrm{Seq(startPos)}}$;
    $f(x_r^{'}, D) = \mathrm{maxD} + 1$;}
  else
    {r = r + 1;
    $x_r^{'} = x_{\mathrm{Seq(startPos)}}$;}
  end if
  if i == n **then**
    r = r + 1;
    $x_r^{'} = x_{\mathrm{Seq}(i)}$;
  end if
  i = i + 1;
end while
return $S^{'}$.

Algorithm 2: Counting sort for a single attribute.

Input: A Simplified Decision Table $S^{'} = \{U^{'}, C \cup D, V, f\}$ where $U^{'} = \{x_1^{'}, x_2^{'}, \ldots, x_n^{'}\}$, the index of selected conditional attribute: attInd, object serialization:objTag[], remained object number:objTagNum, the partition of object serialization:dividTag[][2], partition number:dividTagNum.
Output: new object serialization:newObjTag[].

Let $\mathrm{maxC} = \max(f(x_j^{'}, C_{\mathrm{attInd}}))$, $\mathrm{minC} = \min(f(x_j^{'}, C_{\mathrm{attInd}}))$ for $\forall x_j \in U$.
totalNum = 0;
newObjTagNum = 0;
for j = 1 : dividTabNum **do**
  for i = minC : maxC **do**
    Count[i] = 0;
  end for
  for i = dividTag(j, 1) to dividTag(j, 2) **do**
    Count[f($x^{'}_{\mathrm{objTag(i)}}$, C_{attInd})]=Count[f($x^{'}_{\mathrm{objTag(i)}}$, C_{attInd})] +1;
  end for
  Count(minC) = Count(minC) + totalNum;
  for i = minC + 1 to maxC **do**
    Count[i] = Count[i] + Count[i-1];
  end for
  for i = dividTag(j, 2) to dividTag(j, 1) **do**
    newObjTag(Count($f(x^{'}_{\mathrm{objTag}(i)}, C_{\mathrm{attInd}})$))= objTag($i$); count($f(x^{'}_{\mathrm{objTag}(i)}, C_{\mathrm{attInd}})$) =count($f(x^{'}_{\mathrm{objTag}(i)}, C_{\mathrm{attInd}})$)−1;
  end for
  totalNum = dividTag(j, 2) - dividTag(j, 1) + 1 + totalNum
  ;
end for
return newObjTag.

Through Algorithm 1, we can get a simplified table $S^{'} = \{U^{'}, C \cup D, V, f\}$, which has no inconsistent objects.

In Algorithm 2, input variable attInd represents the index of the attributes selected currently. objTag represents the objects index of inconsistent pairs in $U_R^{''}$ (R represents the selected attributes). objTagNum is the object number of objTag. dividTag[][2] is division of equivalence class, and dividTag[i][1] means the starting position of *i*th equivalence class in objTag and dividTag[i][2] is the end position. Using the parameters above, we can have a detailed description of the inconsistent pairs of $U_R^{''}$.

Algorithms 3 and 4 are key steps to measure candidate attributes of QNIPS and QNIPS2, respectively. The output parameter inConsAtt is the sigNIPS/sigNIPS2 value of candidate attributes.

In our algorithm, we need to choose a random attribute as the initial reduct. Since it is selected randomly, we design Al-

Algorithm 3: Calculate sigNIPS.

Input: A Simplified Decision Table $S^{'} = (U', C \cup D)$ where $U^{'} = \{x_1^{'}, x_2^{'}, \ldots, x_n^{'}\}$, the index of selected conditional attributes: attInd, object serialization:objTag[], remained object number:objTagNum, the partition of object serialization:dividTag[][2], partition number:dividTagNum.

Output: inConsAtt[], newObjTag[], newObjTagNum, newDividTag[][2], newDividTagNum.

$\text{inConsAtt}_i = 0 (i = 1, \ldots, |C|)$;
i = 2;
newObjTagNum = 0;
newDividTagNum = 0;
unEqualFlag = 0;
posNum = 0;
for j = 1 : dividTagNum **do**
  i = dividTag(j, 1) + 1;
  while i ≤ dividTag(j, 2) **do**
    startPos = i - 1;
    while $f(x^{'}_{\text{objTag}(i)}, C_{\text{attInd}}) = f(x^{'}_{\text{objTag}(i-1)}, C_{\text{attInd}})$ **do**
      if $f(x^{'}_{rmobjTag(i)}, D) \neq f(x^{'}_{\text{objTag}(i-1)}, D)$ **then**
      unEequalFalg = 1;
      if $f(x^{'}_{\text{objTag}(i)}, C_k) \neq f(x^{'}_{\text{objTag}(i-1)}, C_k)$, $\forall C_k \in (C - R)$ **then**
        $\text{inConsAtt}_k = \text{inConsAtt}_k + 1$;
      end if
    end if
    i = i + 1;
    if i > dividTag(j, 2) **then**
      break;
    end if
  end while
  endPos = i − 1;
  if unEqualFlag = 1 **then**
    unEqualFlag = 0;
    newDividTagNum = newDividTagNum +1;
    newDividTag(newDividTagNum, 1) = startPos-posNum;
    newDividTag(newDividTagNum, 2) = endPos-posNum;
    for k = startPos : endPos **do**
      newObjTagNum = newObjTagNum + 1;
      newObjTag(newObjTagNum) = objTag(k);
    end for
  else
    posNum = endPos-startPos + 1 + posNum;
  end if
  if i == dividTag(j, 2) **then**
    posNum = posNum + 1;
  end if
  i = i + 1;
end while
end for
return inConsAtt, newObjTag, newObjTagNum, newDividTag, newDividTagNum.

Algorithm 4: Calculate sigNIPS2.

Input: A Simplified Decision Table $S^{'} = (U', C \cup D)$ where $U^{'} = \{x_1^{'}, x_2^{'}, \ldots, x_n^{'}\}$, the index of selected conditional attributes: attInd, object serialization:objTag[], remained object number:objTagNum, the partition of object serialization:dividTag[][2], partition number:dividTagNum.

Output: inConsAtt[], newObjTag[], newObjTagNum, newDividTag[][2], newDividTagNum.

$\text{inConsAtt}_i = 0 (i = 1, \ldots, |C|)$; i = 2;
newObjTagNum = 0;
newDividTagNum = 0;
unEqualFlag = 0;
posNum = 0;
for j = 1 : dividTagNum **do**
  i = dividTag(j, 1) + 1;
  while i ≤ dividTag(j, 2) **do**
    startPos = i − 1;
    while $f(x^{'}_{\text{objTag}(i)}, C_{\text{attInd}}) = f(x^{'}_{\text{objTag}(i-1)}, C_{\text{attInd}})$ **do**
      if $f(x^{'}_{\text{objTag}(i)}, D) \neq f(x^{'}_{\text{objTag}(i-1)}, D)$ **then**
      unEequalFalg = 1;
      if $f(x^{'}_{\text{objTag}(i)}, C_k) \neq f(x^{'}_{\text{objTag}(i-1)}, C_k)$, $\forall C_k \in (C - R)$ **then**
        $\text{inConsAtt}_k = \text{inConsAtt}_k +$ $\text{diff}(x^{'}_{\text{objTag}(i)}, x^{'}_{\text{objTag}(i-1)}, C_k)$;
      end if
    end if
    i = i + 1;
    if i > dividTag(j, 2) **then**
      break;
    end if
  end while
  endPos = i - 1;
  if unEqualFlag = 1 **then**
    unEqualFlag = 0;
    newDividTagNum = newDividTagNum + 1;
    newDividTag(newDividTagNum, 1) = startPos-posNum;
    newDividTag(newDividTagNum, 2) = endPos-posNum;
    for k = startPos : endPos **do**
      newObjTagNum = newObjTagNum + 1;
      newObjTag(newObjTagNum) = objTag(k);
    end for
  else
    posNum = endPos-startPos+1+posNum;
  end if
  if i == dividTag(j, 2) **then**
    posNum = posNum + 1;
  end if
  i = i + 1;
end while
end for
return inConsAtt, newObjTag, newObjTagNum, newDividTag, newDividTagNum.

Algorithm 5: Test the initial reduct.

Input: A Simplified Decision Table $S^{'} = (U^{'}, C \cup D)$ where $U^{'} = \{x_1^{'}, x_2^{'}, \ldots, x_n^{'}\}$, $C = \{c_1, c_2, \ldots, c_m\}$, reduct R,
initial reduct $R^{'}$,
Output: reduct R.

```
1: R = R - R';
2: Let {MaxC_i = max(f(x_j', c_i)), MinC_i = min(f(x_j', c_i))
   for ∀x_j' ∈ U', where c_i ∈ R. Seq = {1, 2, ..., n}.
3: for i = 1 to m do
4:   if c_i ∈ R then
5:     for j = MinC_i to MaxC_i do
6:       {Count[j]=0;}
7:     end for
8:     for j = 1 to n do
9:       {Count[f(x_j', c_i)]=Count[f(x_j', c_i)]+1;
         TempSeq[j]=Seq[j];}
10:    end for
11:    for k = MinC_i to MaxC_i do
12:      {Count[k]=Count[k] + Count[k-1];}
13:    end for
14:    for j = n to 1 do
15:      {Seq[Count[f(x'_TempSeq[j], c_i)]] = TempSeq[j];
         Count[f(x'_TempSeq[j], c_i)] = Count[f(x'_TempSeq[j], c_i)]−1;}
16:    end for
17:  end if
18: end for
19: i = 2, endFlag = 0;
20: while i ≤ |U'| do
21:   while f(x'_Seq[i], c_j) = f(x'_Seq[i−1], c_j) for all c_j ∈ R do
22:     if f(x'_Seq[i], D) ≠ f(x'_Seq[i−1], D) then
23:       {R = R ∪ R';
          endFlag = 1;
          break ;}
24:     else
25:       i = i + 1;
26:     end if
27:   end while
28:   if endFlag = 1 then
29:     break;
30:   end if
31:   i = i + 1;
32: end while
33: return R;
```

gorithm 5 to check its necessity. We divide Algorithm 5 into the following two parts:

1) Steps 1 to 18, use counting sort to get $U^{''}_{R-R'}$($R^{'}$ is initial reduct).
2) Steps 19–32, judging whether there exits neighbor inconsistent pairs. If there exits near inconsistent neighbor pairs, we say $R^{'}$ is necessary, otherwise $R^{'}$ is unnecessary.

Algorithm 6: QNIPS.

Input: A decision table $S = (U, C \cup D)$, where $U = \{x_1, x_2, \ldots, x_n\}$, $C = \{c_1, c_2, \ldots, c_m\}$, D = {d}.
Output: An attribute reduct R

```
1: Change S into the simplified decision table by Algorithm 1;
2: Randomly select a conditional attribute c ∈ C as initial reduct:R = c, c' = c;
   objTag[i] = i, ∀i ∈ [1, |U|];
   objTagNum = |U|;
   dividTag(1, 1) = 1;
   dividTag(1, 2) = |U|;
   dividTagNum = 1;
3: while objTagNum > 0 do
4:   sort with newly add conditional attribute c' by Algorithm 2;
5:   get inConsAtt, objTag, objTagNum, dividTag, dividTagNum by Algorithm 3
6:   if objTagNum > 0 and inConsAtt_i = max(inConsAtt) then
7:     R = R ∪ c_i, c' = c_i;
8:   end if
9: end while
10: Test whether the initial reduct c is necessery by Algorithm 5;
11: return R;
```

Here, we propose an attribute selection algorithm based on the adjacent inconsistent pair (see Algorithm 6). It follows the general process of attribute selection algorithm. In our method, we select a random attribute as the initial reduct. Then, we use forward selection method to conduct the selecting process. By judging whether objTagNum is equal to 0 to stop the feature selection (objTagNum $= 0$ means $U^{''}_R$ is consistent). Finally, we need to check whether the initial reduct is effective. If the initial reduct is not effective, we delete it from the reduct R. As for Algorithm 7, it repeats the same process as Algorithm 6.

Before analyzing the time complexity, we should pay attention to several variables: objTag[], objTagNum, dividTag[][2], and dividTagNum. Let us take Table III as an example. At first: objTag = [1, 2, 3, 4, 5, 6, 7], objTagNum = 7, dividTag[1][1] = 1, dividTag[1][2] = 2, dividTagNum = 1. Variable objTag[] contains the object subscript sequentially, and objTagNum represents the number of subscripts contained in objTag[]. dividTag contains the lower bound and upper bound of the index of a partition in objTag, and dividTagNum denotes the number of the partition in objTag. Let $R = c_1$, then if $U^{'}/R = \{\{x_1\}, \{x_2, x_6\}, \{x_3, x_4, x_5, x_7\}\}$, we can get that objTag = [1, 2, 6, 3, 4, 5, 7], objTagNum = 7, dividTag[1][1] = 1, dividTag[1][2] = 1; dividTag[2][1] = 2, dividTag[2][2] = 3;dividTag[3][1] = 4, dividTag[3][2] = 7; dividTagNum = 3. By Theorem 2, we know that $[x_1]$ do not contribute to

Algorithm 7: QNIPS2.

Input: A decision table $S = (U, C \cup D)$, where $U = \{x_1, x_2, \ldots, x_n\}$, $C = \{c_1, c_2, \ldots, c_m\}$, D = {d}.
Output: A attribute reduct R

1: Change S into the simplified decision table by Algorithm 1;
2: Randomly select a conditional attribute $c \in C$ as initial reduct:$R = c, c^{'} = c$;
objTag[i] = i, $\forall i \in [1, |U|]$;
objTagNum = $|U|$;
dividTag(1, 1) = 1;
dividTag(1, 2) = $|U|$;
dividTagNum = 1;
3: **while** objTagNum > 0 **do**
4: sort with newly add conditional attribute $c^{'}$ by Algorithm 2;
5: get inConsAtt, objTag, objTagNum, dividTag, dividTagNum by Algorithm 4
6: **if** objTagNum > 0 and $\mathrm{inConsAtt}_i$ = max(inConsAtt) **then**
7: $R = R \cup c_i, c^{'} = c_i$;
8: **end if**
9: **end while**
10: Test whether the initial reduct c is necessery by Algorithm 5;
11: **return** R;

$\mathrm{sig}_{\mathrm{NIP}}(R, a, D)$ $\forall a \in (C - R)$. Hence, in Algorithms 3 or 4, we neglect $[x_1]$, and we can reduce the time cost without affecting the accuracy in calculating $\mathrm{sig}_{\mathrm{NIP}}(R, a, D)$. In other words, what we get actually is objTag = [2, 6, 3, 4, 5, 7], objTagNum = 6, dividTag[2][1] = 1, dividTag[2][2] = 2; dividTag[3][1] = 3, dividTag[3][2] = 6, dividTagNum = 2.

In the following, we analyze the time complexity of the algorithms.

In Algorithm 1, the time complexity of steps 3–14 is $O(|U|)$, so the time complexity for steps 2–14 is $O(|U||C|)$. As for steps 17–34, it is clear that the time complexity is $O(|U||C|)$. In summary, the time complexity of Algorithm 1 is $O(|U||C|)$.

In Algorithm 2, the time complexity of steps 3–16 is $O(\mathrm{dividTag}[j][2] - \mathrm{dividTag}[j][1])$. Since dividTag represents the result of partition of objTag, and objTagNum $\leq |u^{'}|$, which means that (dividTag[1][2] − dividTag[1][1])+(dividTag[2][2]−dividTag[2][1]) + . . . + (dividTag[dividTagNum][2]−dividTag[dividTagNum][1]) = objTagNum. So, the time complexity of steps 2–17 is $O(|U^{'}|)$. In summary, the time complexity for Algorithm 2 is $O(|U^{'}|)$.

As for Algorithm 3, the time complexity of steps 9–11 is $O(|C - R|)$, the time complexity of steps 3–31 is $O((\mathrm{dividTag}[j][2] - \mathrm{dividTag}[j][1]) * |C - R|)$. Since (dividTag[1][2] − dividTag(dividTag[2][2]−dividTag[2][1])+ . . . + (dividTag[dividTagNum][2]−dividTag[dividTagNum][1]) = objTagNum. So its time complexity is $O(|U^{'}||C - R|)$. In summary, the time complexity of Algorithm 3 is $O(|U^{'}||C|)$. We can obtain the same conclusion for Algorithm 4.

For Algorithm 5, the time complexity of steps 4–17 is $O(|U^{'}|)$, so the time complexity for steps 3–18 is $O(|U^{'}|(|R| - 1))$. As for steps 19–32, the time complexity is at most $(|U^{'}||R|)$. In summary, the time complexity for Algorithm 5 is $O(|U^{'}||R|)$.

For Algorithm 6, the time complexity of steps 1, 2, and 10 is $O(|U||C|)$, $O(1)$, and $O(|U^{'}||R|)$. As for steps 4 and 5, the time complexity is $O(|U^{'}|)$ and $O(|U^{'}||C|)$, steps 6–8 is $O(1)$, so the time complexity of steps 3–9 is at most $O(|U^{'}||C^2|)$. Thus, the time complexity of Algorithm 6 is $\max(O(|C||U|), O(|C|^2|U/C|))$ (Note that $|U^{'}| = |U/C|$). We can draw the same conclusion for Algorithm 7.

TABLE IV
WHOLE RUNNING TIME OF DIFFERENT METHODS

Data Set	SPS	xuPos	NIPS	QNIPS	QNIPS2
audio	0.9178	0.1228	0.0189	0.0052	0.0057
autos	0.2140	0.0048	0.0042	0.0017	0.0023
breast	0.2819	0.0076	0.0105	0.0029	0.0029
car	9.9925	0.0120	0.0423	0.0114	0.0120
chess	out of memory	0.3124	0.4938	0.0544	0.0709
cnae	out of memory	12.9671	3.5118	0.4650	0.5541
connect	out of memory	23.6656	12.8140	2.2110	2.6144
ipums	out of memory	0.8463	0.4497	0.1819	0.1962
mush	out of memory	0.2383	0.2508	0.0706	0.0721
nursery	out of memory	0.2195	0.4328	0.1052	0.1140

V. EXPERIMENTAL STUDY

In this section, we compare two kinds of quick neighbor inconsistent pair selection algorithms QNIPS (employ sigNIPS) and QNIPS2 (employ sigNIPS2) with three other attribute reduction methods, including the reduction method based on sample selection (denoted by SPS) [46], the reduction method based on the positive-region (denoted by xuPos) [40], and the reduction method based on the neighbor inconsistent pair selection (denoted by NIPS) [47].

A. Experiment Setup

Twelve datasets taken from UCI[1] are used to conduct experiments. The details of the datasets are shown in Table V. The hardware environment: pentium(R) D CPU 2.93 GHz, 2.00 GB Memory. The software environment: MATLAB 7.0.

B. Experimental Comparison

For the purpose of comprehensive analysis, we assume that a good attribute reduction method should satisfy three conditions.

1) Speed : the time cost should be as less as possible.
2) Length : the reduct obtained by the algorithm should be as short as possible, i.e., contains as less attributes as possible.
3) Accuracy : the reduct obtained should contain information as much as possible.

In the following, we conduct our experimental comparison in keeping with the above three aspects. Note that, because of its

[1]http://www. ics. uci.e.u/mlearn/MLRepository. html

TABLE V
DETAILED INFORMATION OF THE DATASETS USED IN THE EXPERIMENTS

	Data Set	Abbreviation	Objects	Conditional attributes	Data type	Decision classes
1	Audiology (Standardized)	audio	226	69	Categorical	24
2	Autos	autos	203	8	Categorical	6
3	Breast Cancer	breast	286	9	Categorical	2
4	CNAE 9	cnae	1080	856	Categorical	9
5	Car Evaluation	car	1728	6	Categorical	4
6	Chess (King Rook vs King Pawn)	chess	3196	36	Categorical	2
7	Mushroom	mush	8124	22	Categorical	3
8	Nursery	nursery	12960	8	Categorical	5
9	KDD ipums la 99 small	ipums	8844	60	Categorical	9
10	Connect 4	connect	67557	42	Categorical	3
11	Amazon Commerce reviews set	amazon	1500	10000	Categorical	50
12	Poker Hand	poker	1025010	10	Categorical	10

TABLE VI
CLASSIFICATION PERFORMANCE RESULT BY C4.5 CLASSIFIERS

data sets	Full attributes		xuPos		QNIPS		QNIPS2		NIPS		SPS	
	avg	std	avg	std	avg	std	avg	std	avg	std	avg	std
audio	76.4675	0.0211	74.4191	0.0236	73.4152	0.2712	71.9933	0.1632	73.4152	0.2712	71.7887	0.0628
autos	65.1269	0.1948	65.1269	0.1948	65.1269	0.1948	65.1269	0.1948	65.1269	0.1948	65.1269	0.1948
breast	68.7407	0.1529	68.7407	0.1529	68.7407	0.1529	67.1434	0.1637	68.7407	0.1529	67.1433	0.1637
car	93.8238	0.0060	93.8238	0.0060	93.8238	0.0060	93.8238	0.0060	93.8238	0.0060	93.8238	0.0060
chess	99.4531	7.0711E-5	99.3675	3.6383E-4	98.9847	4.0110E-4	98.9912	5.01261E-4	98.9847	4.0110E-4	out of memory	out of memory
cnae	87.1917	0.1000	87.8079	0.0404	86.6373	0.07458	86.9801	0.0550	86.6373	0.07458	out of memory	out of memory
connect	79.9840	0.0011	79.9599	0.0016	79.9404	0.0016	79.9404	0.0016	79.9404	0.0016	out of memory	out of memory
ipums	83.5092	0.0030	83.6826	0.0014	84.4351	0.0033	84.3342	0.0015	84.4351	0.0033	out of memory	out of memory
mush	100.0	0.0	100.0	0.0	99.7292	0.0	99.8031	2.0195E-28	99.7292	0.0	out of memory	out of memory
nursery	98.6980	4.1083E-4	98.6980	4.1083E-4	98.6980	4.1083E-4	98.6980	4.1083E-4	98.6980	4.1083E-4	out of memory	out of memory

TABLE VII
CLASSIFICATION PERFORMANCE RESULT BY NAIVEBAYES CLASSIFIERS

data sets	Full attributes		xuPos		QNIPS		QNIPS2		NIPS		SPS	
	avg	std	avg	std	avg	std	avg	std	avg	std	avg	std
audio	72.6726	0.0204	70.7316	0.0098	67.1145	0.0536	68.5421	0.1027	67.1145	0.0536	66.7267	0.1401
autos	58.2117	0.0635	60.0964	0.05195	60.0965	0.0519	60.0965	0.0519	60.0965	0.0519	60.0965	0.0519
breast	72.6087	0.0238	72.6087	0.0238	72.6087	0.0238	73.2828	0.0562	72.6087	0.0238	73.2828	0.0562
car	85.5092	0.0046	85.5092	0.0046	85.5092	0.0046	85.5092	0.0046	85.5092	0.0046	85.5092	0.0046
chess	87.6825	0.0081	88.6634	7.9830E-4	88.0362	0.0073	89.3577	0.0053	88.0362	0.0073	out of memory	out of memory
cnae	94.2681	0.0049	90.1704	4.6049E-4	88.7211	0.0070	90.9571	0.0042	88.7211	0.0070	out of memory	out of memory
connect	72.1421	1.2511E-5	72.1670	3.7183E-6	72.1827	1.2602E-5	72.1827	1.2602E-5	72.1827	1.2602E-5	out of memory	out of memory
ipums	82.9980	4.7728E-4	83.7799	8.2796E-4	84.6179	9.4008E-4	85.2825	0.0011	84.6179	9.4008E-4	out of memory	out of memory
mush	95.7644	1.3685E-4	99.0522	4.3194E-5	98.4969	9.5714E-5	98.9168	2.0195E-28	98.4969	9.5714E-5	out of memory	out of memory
nursery	90.3182	1.5886E-4	90.3182	1.5886E-4	90.3182	1.5886E-4	90.3182	1.5886E-4	90.3182	1.5886E-4	out of memory	out of memory

high space complexity, we can only get our experimental data for SPS from four small datasets (i.e., audio, autos, breast, and car) under our experimental environment.

1) Length: The results of comparison in the number of selected attributes by different attribute reduction methods are shown in Fig. 1. From the result presented in Fig. 1, we find that each algorithm performs almost as well as others.

2) Accuracy: In order to conduct the comparison in performance among these algorithms, we employ NaiveBayes and C4.5 classifiers as the validation functions. Both NaiveBayes and C4.5 classifiers are taken from or implemented in Weka [48]. The average value of classification accuracies and standard deviations of classification accuracies are obtained from ten times tenfolds crossvalidation of NaiveBayes and C4.5, and the results are shown in Tables VI and VII, respectively.

From Tables VI and VII, we know that there is no big difference among these comparative methods. Compared with QNIPS, QNIPS2 performs a little better.

3) Speed: We can find the result of comparison of the whole running time in Table IV. One of the distinctive features in this table is that SPS is only suitable for small datasets because of its high space complexity. What is more, its time cost is also much higher than other algorithms. It is clear that SPS takes more time than other algorithms to find a reduct. To make a better comparison among the algorithms, we can pay attention to Fig. 2. We set the running time of QNIPS as the standard unit in this figure, so that we can easily find the comparison result from it. It shows that QNIPS2 is slightly slower than QNIPS. As for algorithm NIPS, it is much slower than QNIPS and QNIPS2, but it is quicker than xuPos when the dataset containing many attributes. As for xuPos, it is much slower than QNIPS and

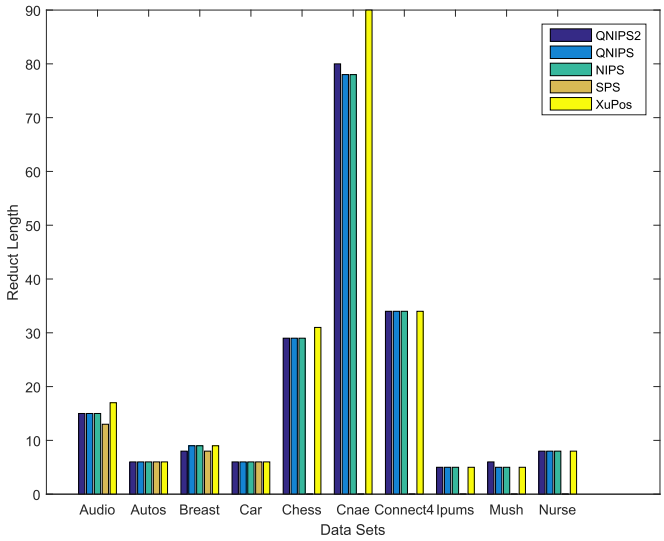


Fig. 1. Comparison on the sizes of selected attributes.

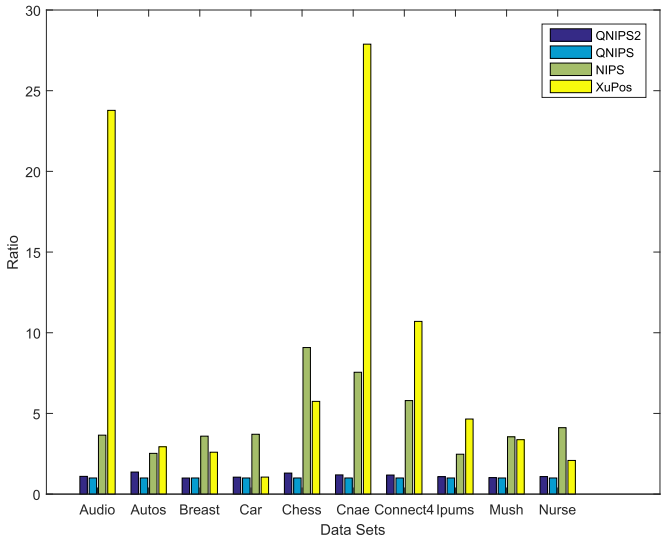


Fig. 2. Comparison of the whole running time.

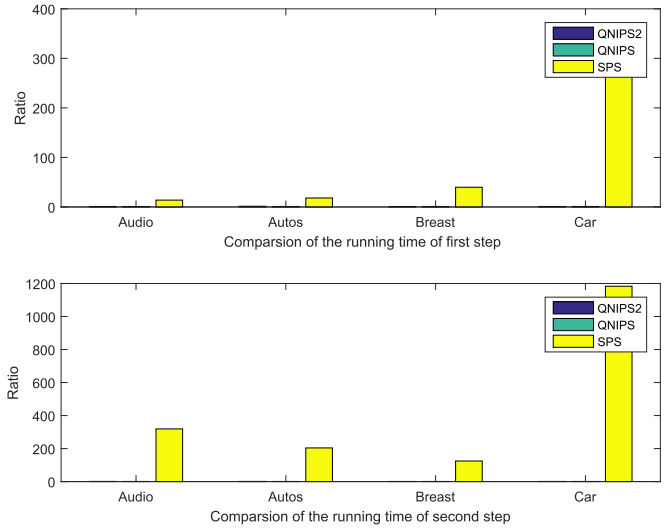


Fig. 3. Comparison with SPS.

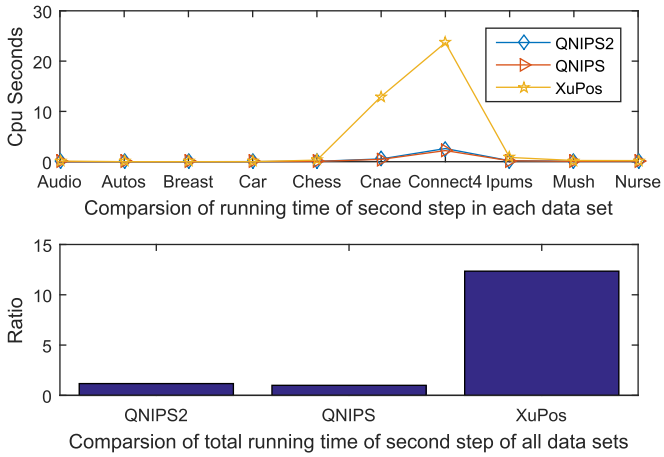


Fig. 4. Comparison with xuPos.

QNIPS2, and it is quicker than NIPS only if the dataset contains a few attributes.

Actually, we can divide the attribute reduction algorithms into two parts: preprocessing decision table and finding a reduct. As for the first step, the compared methods have the same time complexity. Hence, we conduct the following comparison.

1) Comparison with SPS: Fig. 3 shows the result of comparison of the running time of the first and second step, respectively, among SPS, QNIPS, and QNIPS2. We still set the running time of QNIPS as the standard unit. As we can see from the picture, compared with SPS, the time consumption of QNIPS and QNIPS2 can be neglected, in both the first and the second steps. In summary, QNIPS and QNIPS2 is much faster than SPS. Note that, since SPS can only get reduct from dataset: audio, autos, breast, and car, so we only make the comparison based on these four datasets.
2) Comparison with xuPos: Fig. 4 shows the results. From the upper part of Fig. 4, we can find the comparison of running time of the second part on each dataset among QNIPS, QNIPS2, and xuPos. We can get the result of comparison of total running time of all the datasets among QNIPS, QNIPS2, and xuPos in the lower part of Fig. 4, in which we set the running time of QNIPS as a standard unit as well. It is obvious that QNIPS and QNIPS2 are much quicker than xuPos in the second step. What is more, if we pay attention to datasets audio and cane, we can draw the conclusion that the more attributes contained in a dataset, the less time consumed in QNIPS and QNIPS2 relative to xuPos.
3) Comparison with NIPS: The comparison result with NIPS is shown in Fig. 5. From the upper part of Fig. 5, we can find the comparison of running time of the second part on each dataset among QNIPS, QNIPS2, and NIPS. The

TABLE VIII
COMPARISON IN DATASET: AMAZON

	QNIPS		QNIPS2		NIPS		xuPos	
	value	ratio	value	ratio	value	ratio	value	ratio
time1	3.3665	1.0000	3.3862	1.0059	3.3416	0.9926	3.4268	1.0179
time2	1.2299	1.0000	1.4736	1.1981	6.9487	5.6496	65.2018	53.0121
total time	4.5965	1.0000	4.8599	1.0573	10.2903	2.2387	69.0376	15.0197

TABLE IX
COMPARISON IN DATASET: POKER

	QNIPS		QNIPS2		NIPS		xuPos	
	value	ratio	value	ratio	value	ratio	value	ratio
time1	4.2030	1.0000	4.3170	1.0271	4.2064	1.0008	4.3692	1.0395
time2	8.0823	1.0000	9.0930	1.1251	42.3340	5.2379	32.2687	3.9925
total time	12.2853	1.0000	13.4100	1.0915	46.5404	3.7883	36.6379	2.9823

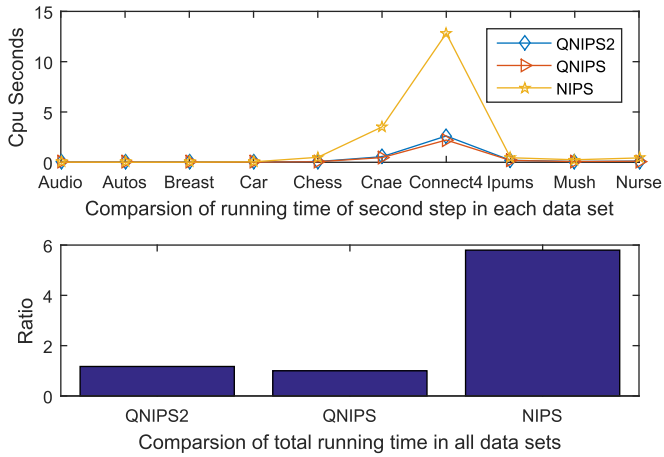


Fig. 5. Comparison with NIPS.

comparison result of the total running time on all datasets is shown in the lower part of Fig. 5, in which we set the running time of QNIPS as standard unit as well. In a summary, it is obvious that QNIPS and QNIPS2 are much quicker than QNIPS in the second step.

4) Comparison in large datasets: To further verify the conclusions drawn above, we conduct the comparison on large datasets among different algorithms. Note that, in Tables VIII and IX, time1 means the running time of the first step, time2 means the running time of the second step, total time means the whole running time. What is more, we still set the running time of QNIPS as standard unit. Since SPS cannot obtain results on these two large datasets, SPS is not listed in the tables. As we can see from Table VIII, there is no big difference among the running time of the first step, however, the running time of the second step varies widely. QNIPS and QNIPS2 are obviously faster than NIPS and xuPos. From Table IX, we can draw a similar conclusion. Note that, from Table V, we know that dataset Amazon contains much more attributes than dataset Poker. Through the comparison results in these two large datasets (see Tables VIII and IX), we can still draw the conclusion that the more attributes are contained in a dataset, the less time is consumed in QNIPs and QUIPS2 in comparison with the above xuPos.

In summary, we can conclude thatQNIPS and QNIPS2 can get a reduct with much less time cost than SPS, xuPos, and NIPS.

VI. CONCLUSION

In this paper, we first propose the concepts of neighbor pair, neighbor inconsistent relation, and neighbor consistent relation. Based on the proposed concepts, through systematic theoretical analysis, we construct two kinds of attribute significance measures from the viewpoint of pair selection. Consequently, we develop two quick neighbor inconsistent pair selection algorithms for attribute reduction in the framework of rough sets. The key characteristic of the proposed algorithms is that they only need to calculate $U^{'}/R$ once, while most existing algorithms based on the positive region have to calculate partition of $U^{'}$ for $|C-R|$ times. What is more, our algorithms only need to deal with the equivalent classes in $U^{'}/R$ that contain at least one neighbor inconsistent pair, i.e., part of objects in $U^{'}$, while others always need to consider all objects in $U^{'}$. Experiments have shown that our algorithms are much faster than the existing rough set attribute reduction algorithms based on the positive region while maintaining the same performance as the other strategies.

Our future work aims to construct an efficient way to search all reducts for a given decision table and apply our approach to other kinds of information systems.

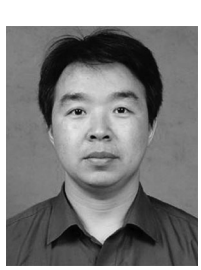
REFERENCES

[1] Z. Pawlak, "Rough sets," *Int. J. Comput. Inf. Sci.*, vol. 11, no. 5, pp. 341–356, 1982.
[2] Z. Pawlak, *Rough Sets: Theoretical Aspects of Reasoning About Data*. Norwell, MA, USA: Kluwer, 1991.
[3] J. Dai and Q. Xu, "Attribute selection based on information gain ratio in fuzzy rough set theory with application to tumor classification," *Appl. Soft Comput.*, vol. 13, no. 1, pp. 211–221, 2013.
[4] Q. Hu, W. Pedrycz, D. Yu, and J. Lang, "Selecting discrete and continuous features based on neighborhood decision error minimization," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 40, no. 1, pp. 137–150, Feb. 2010.

[5] H. Chen, T. Li, C. Luo, S.-J. Horng, and G. Wang, "A rough set-based method for updating decision rules on attribute values' coarsening and refining," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 12, pp. 2886–2899, Dec. 2014.
[6] J. Dai, "Rough set approach to incomplete numerical data," *Inf. Sci.*, vol. 241, pp. 43–57, 2013.
[7] D. Liang and D. Liu, "A novel risk decision making based on decision-theoretic rough sets under hesitant fuzzy information," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 2, pp. 237–247, Apr. 2015.
[8] Y. Y. Yao, "The superiority of three-way decisions in probabilistic rough set models," *Inf. Sci.*, vol. 181, pp. 1080–1096, 2011.
[9] S. Zhao, H. Chen, C. Li, X. Du, and H. Sun, "A novel approach to building a robust fuzzy rough classifier," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 4, pp. 769–786, Aug. 2015.
[10] H. Chen, T. Li, C. Luo, S.-J. Horng, and G. Wang, "A decision-theoretic rough set approach for dynamic data mining," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 6, pp. 1958–1970, Dec. 2015.
[11] F. Feng, X. Liu, V. Leoreanu-Fotea, and Y. B. Jun, "Soft sets and soft rough sets," *Inf. Sci.*, vol. 181, no. 6, pp. 1125–1137, 2011.
[12] F. Feng, C. Li, B. Davvaz, and M. I. Ali, "Soft sets combined with fuzzy sets and rough sets: A tentative approach," *Soft Comput.*, vol. 14, no. 9, pp. 899–911, 2010.
[13] W. Li, Z. Huang, X. Jia, and X. Cai, "Neighborhood based decision-theoretic rough set models," *Int. J. Approx. Reason.*, vol. 69, pp. 1–17, 2016.
[14] J. Liang, F. Wang, C. Dang, and Y. Qian, "A group incremental approach to feature selection applying rough set technique," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 2, pp. 294–308, Feb. 2014.
[15] P. Maji, "A rough hypercuboid approach for feature selection in approximation spaces," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 16–29, Jan. 2014.
[16] J. Wang, P. Zhao, S. C. Hoi, and R. Jin, "Online feature selection and its applications," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 3, pp. 698–710, Mar. 2014.
[17] D. Chen, L. Zhang, S. Zhao, Q. Hu, and P. Zhu, "A novel algorithm for finding reducts with fuzzy rough sets," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 2, pp. 385–389, Apr. 2012.
[18] J. Dai, W. Wang, H. Tian, and L. Liu, "Attribute selection based on a new conditional entropy for incomplete decision systems," *Knowl.-Based Syst.*, vol. 39, pp. 207–213, 2013.
[19] M. I. Ali, "Another view on reduction of parameters in soft sets," *Appl. Soft Comput.*, vol. 12, no. 6, pp. 1814–1821, 2012.
[20] J. Chen, Y. Lin, G. Lin, J. Li, and Z. Ma, "The relationship between attribute reducts in rough sets and minimal vertex covers of graphs," *Inf. Sci.*, vol. 325, pp. 87–97, 2015.
[21] M. Dash and H. Liu, "Consistency-based search in feature selection," *Artif. Intell.*, vol. 151, no. 1, pp. 155–176, 2003.
[22] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, 2003.
[23] W. Pedrycz and G. Vukovich, "Feature analysis through information granulation and fuzzy sets," *Pattern Recognit.*, vol. 35, no. 4, pp. 825–834, 2002.
[24] D. Ślezak, "Approximate entropy reducts," *Fundam. Inf.*, vol. 53, no. 3, pp. 365–390, 2002.
[25] Y. Qian and J. Liang, "Combination entropy and combination granulation in rough set theory," *Int. J. Uncertain., Fuzziness Knowl.-Based Syst.*, vol. 16, no. 2, pp. 179–193, 2008.
[26] J. Liang and Z. Xu, "The algorithm on knowledge reduction in incomplete information systems," *Int. J. Uncertain., Fuzziness Knowl.-Based Syst.*, vol. 10, no. 1, pp. 95–103, 2002.
[27] C. Wang, M. Shao, Q. He, Y. Qian, and Y. Qi, "Feature subset selection based on fuzzy neighborhood rough sets," *Knowl.-Based Syst.*, vol. 111, no. 1, pp. 173–179, 2016.
[28] C. Wang *et al.*, "A fitting model for feature selection with fuzzy rough sets," *IEEE Trans. Fuzzy Syst.*, vol. PP, pp. 1–13, 2016, doi: 10.1109/TFUZZ.2016.2574918.
[29] J. Dai, Q. Hu, J. Zhang, H. Hu, and N. Zheng, "Attribute selection for partially labeled categorical data by rough set approach," *IEEE Trans. Cybern.*, vol. PP, pp. 1–12, 2016, doi: 10.1109/TCYB.2016.2636339.
[30] J. Dai, H. Han, Q. Hu, and M. Liu, "Discrete particle swarm optimization approach for cost sensitive attribute reduction," *Knowl.-Based Syst.*, vol. 102, pp. 116–126, 2016.
[31] Y. Qian, J. Liang, W. Pedrycz, and C. Dang, "Positive approximation: An accelerator for attribute reduction in rough set theory," *Artif. Intell.*, vol. 174, no. 9, pp. 597–618, 2010.
[32] M. Inuiguchi, Y. Yoshioka, and Y. Kusunoki, "Variable-precision dominance-based rough set approach and attribute reduction," *Int. J. Approx. Reason.*, vol. 50, no. 8, pp. 1199–1214, 2009.
[33] N. Parthalain, Q. Shen, and R. Jensen, "A distance measure approach to exploring the rough set boundary region for attribute reduction," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 3, pp. 305–317, Mar. 2010.
[34] Q. He, C. Wu, D. Chen, and S. Zhao, "Fuzzy rough set based attribute reduction for information systems with fuzzy decisions," *Knowl.-Based Syst.*, vol. 24, no. 5, pp. 689–696, 2011.
[35] J. Dai, W. Wang, and Q. Xu, "An uncertainty measure for incomplete decision tables and its applications," *IEEE Trans. Cybern.*, vol. 43, no. 4, pp. 1277–1289, Aug. 2013.
[36] M. Yang, "An incremental updating algorithm for attribute reduction based on improved discernibility matrix," *Chin. J. Comput.*, vol. 30, pp. 815–822, 2007.
[37] M. Aggarwal, "Rough information set and its applications in decision making," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 2, pp. 265–276, Apr. 2017.
[38] J. Qian, D. Q. Miao, Z. H. Zhang, and W. Li, "Hybrid approaches to attribute reduction based on indiscernibility and discernibility relation," *Int. J. Approx. Reason.*, vol. 52, pp. 212–230, 2011.
[39] S. Nguyen and H. Nguyen, "Some efficient algorithms for rough set methods," in *Proc. Int. Conf. Inf. Process. Manage. Uncertain. Knowl. Based Syst.*, 1996, pp. 1451–1456.
[40] Z. Xu, Z. Liu, B. Yang, and W. Song, "Quick attribute reduction algorithm with complexity of $max(o(c||u|), o(|c|(2)|u/c|))$|," *Chin. J. Comput.*, vol. 29, no. 3, pp. 391–399, 2006.
[41] A. Skowron and C. Rauszer, "The discernibility matrices and functions in information systems," in *Intelligent Decision Support*. New York, NY, USA: Springer, 1992, pp. 331–362.
[42] D. Chen, C. Wang, and Q. Hu, "A new approach to attribute reduction of consistent and inconsistent covering decision systems with covering rough sets," *Inf. Sci.*, vol. 177, no. 17, pp. 3500–3518, 2007.
[43] J. Wang and J. Wang, "Reduction algorithms based on discernibility matrix: The ordered attributes method," *J. Comput. Sci. Technol.*, vol. 16, no. 6, pp. 489–504, 2001.
[44] D. Ye and Z. Chen, "A new discernibility matrix and the computation of a core," *Acta Electron. Sinica*, vol. 30, no. 7, pp. 1086–1088, 2002.
[45] Y. Ming and Z. Sun, "Improvement of discernibility matrix and the computation of a core," *J. Fudan Univ.(Natural Sci.)*, vol. 43, no. 5, pp. 865–868, 2004.
[46] D. Chen, S. Zhao, L. Zhang, Y. Yang, and X. Zhang, "Sample pair selection for attribute reduction with rough set," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 11, pp. 2080–2093, Nov. 2012.
[47] D. Huang, "Rough set feature selection based on neighbor inconsistent pair," Master's thesis, College of Computer Science, Zhejiang Univ., Hangzhou, China, 2016.
[48] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. San Mateo, CA, USA: Morgan Kaufmann, 2011.

Jianhua Dai received the B.Sc., M.Eng., and Ph.D. degrees in computer science from Wuhan University, Wuhan, China, in 1998, 2000, and 2003, respectively.

He was with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China, from 2005 to 2015. He is currently a Full Professor with the School of Computer Science and Technology, Tianjin University, Tianjin, China. He has published more than 80 research papers in refereed journals and conferences. His current research interests include artificial intelligence, machine learning, data mining, evolutionary computation, and soft computing.

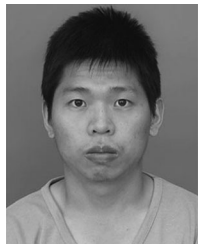


Qinghua Hu (SM'13) received the B.S., M.S., and Ph.D. degrees from the Harbin Institute of Technology, Harbin, China, in 1999, 2002, and 2008, respectively.

He was a Postdoctoral Fellow with the Department of Computing, Hong Kong Polytechnic University, Hong Kong, from 2009 to 2011. He is currently a Full Professor and the Vice Dean of the School of Computer Science and Technology, Tianjin University, Tianjin, China. He has authored more than 100 journal and conference papers. His current research interests include rough sets, granular computing, and data mining for classification and regression.

Hu Hu received the B.Sc. degree in computer science and technology from the Hefei University of Technology, Hefei, China, in 2015. He is currently working toward the graduation degree with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China.

His current research interests include machine learning, data mining, and rough sets.



Debiao Huang received the B.Sc. degree in computer science and technology from Northeast University, Shenyang, China, in 2013. He is currently working toward the graduate degree with the College of Computer Science, Zhejiang University, Hangzhou, China.

His research interests include machine learning, data mining, and rough sets.