Incremental approaches for updating approximations in set-valued ordered information systems

Chuan Luo a, Tianrui Li a,*, Hongmei Chen a, Dun Liu b

a School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China
b School of Economics and Management, Southwest Jiaotong University, Chengdu 610031, China

ARTICLE INFO

Article history:
Received 6 October 2012
Received in revised form 17 June 2013
Accepted 18 June 2013
Available online 28 June 2013

Keywords:
Rough set theory
Incremental learning
Knowledge discovery
Set-valued information systems
Approximations

ABSTRACT

Incremental learning is an efficient technique for knowledge discovery in a dynamic database, which enables acquiring additional knowledge from new data without forgetting prior knowledge. Rough set theory has been successfully used in information systems for classification analysis. Set-valued information systems are generalized models of single-valued information systems, which can be classified into two categories: disjunctive and conjunctive. Approximations are fundamental concepts of rough set theory, which need to be updated incrementally while the object set varies over time in the set-valued information systems. In this paper, we analyze the updating mechanisms for computing approximations with the variation of the object set. Two incremental algorithms for updating the approximations in disjunctive/conjunctive set-valued information systems are proposed, respectively. Furthermore, extensive experiments are carried out on several data sets to verify the performance of the proposed algorithms. The results indicate the incremental approaches significantly outperform non-incremental approaches with a dramatic reduction in the computational speed.

© 2013 The Authors. Published by Elsevier B.V. All rights reserved.

1. Introduction

Granular Computing (GrC), a new concept for information processing based on Zadeh's “information granularity”, is a term of theories, methodologies, techniques, and tools that makes use of granules in the process of problem solving [1,2]. With the development of artificial intelligence, the study on the theory of GrC has aroused the concern of more and more researchers [3–5]. Up to now, GrC has been successfully applied to many branches of artificial intelligence. The basic notions and principles of GrC have appeared in many related fields, such as concept formation [6], data mining [7] and knowledge discovery [8,9].

Rough Set Theory (RST) is a powerful mathematical tool for dealing with inexact, uncertain or vague information [10]. It is also known as one of three primary models of GrC [11]. In recent years, there has been a rapid growth of interest in RST and its applications. It seems to be of fundamental importance to artificial intelligence and cognitive sciences, especially in the areas of machine learning, decision analysis, expert systems, inductive reasoning and pattern recognition [13–16]. The data acquired for rough set analysis is represented in form of attribute-value tables, consisting of objects (rows) and attributes (columns), called information systems [17]. In real-life applications, data in information systems is generated and collected dynamically, and the knowledge discovered by RST need to be updating accordingly [12]. The incremental technique is an effective method to update knowledge by dealing with the new added-in data set without re-implementing the original data mining algorithm [18,19]. Many studies have been done towards the topic of incremental learning techniques under RST. Considering the problem of discretization of continuous attributes in the dynamic databases, Dey et al. developed a dynamic discretization method based on RST and notions of Statistics, which merges the two tasks of discretization and reduction of attributes into a single seamless process, so as to reduce the computation time by using samples instead of the whole data to discretize the variables [20]. Considering the problem of dynamic attribute reduction, Hu et al. proposed an incremental positive region reduction algorithm based on elementary set, which can generate a new positive region reduction quickly when a new object is added into the decision information systems [28]. From the view of information theory, Wang et al. proposed an incremental attribute reduction algorithm based on three representative entropies by considering changes of data values, which can generate a feasible reduce in a much shorter time. However, the algorithm is only applicable on the case of the variation of data one by one [21]. Furthermore, Wang et al. developed a dimension incremental strategy...
for attribute reduction based on information entropy for data sets with dynamically increasing attributes [22]. Since the core of a decision table is the start point to many existing algorithms of attribute reduction, Yang et al. introduced an incremental updating algorithm of the computation of a core based on the discernibility matrix, which only inserts a new row and column, or deletes one row and updates corresponding column when updating the discernibility matrix [29]. Considering the problem of dynamic rule induction, Fan et al. proposed an incremental rule-extraction algorithm (REA) based on RST, which updates rule sets by partly modifying original rule sets without re-computing rule sets from the very beginning and the proposal approach is especially useful in a large database, since it does not re-compute the reducts/rules that are not influenced by the incremental data set [23]. Nevertheless, alternative rules which are as preferred as the original desired rules might exist since the maximum of strength index is not unique. The REA may lead to non-complete rules, then an incremental alternative rule extraction algorithm (IAREA) was proposed to exclude the repetitive rules and to avoid the problem of redundant rules [24]. Zheng et al. developed a rough set and rule tree based incremental algorithm for knowledge acquisition, which is not only obviously quicker than that of classic algorithm, but also has a better performance of knowledge learned by the proposed algorithm to a certain degree [25]. Liu et al. defined a new concept of interesting knowledge based on both accuracy and coverage of the generated rules in the information system, and presented an optimization model using the incremental matrix for generating interesting knowledge when the object set varies over time [26,27].

The main goal of RST is to synthesize approximations of concepts from the acquired data, which is a necessary step for expressing and reducing incomplete and uncertain knowledge based on RST [30–32]. The knowledge hidden in information systems can be discovered and expressed in the form of decision rules according to the lower and upper approximations [36–39]. In order to resolve the problem of high computation complexity in computing approximations under the dynamic information systems, many incremental updating algorithms have been proposed. Therefore, extensive efforts have been devoted to efficient algorithms for computing approximations. Li et al. presented an incremental method for updating approximations in an incomplete information system through the characteristic relation when the attribute set varies over time, which can deal with the case of adding and removing some attributes simultaneously in the information system [40]. Since the domain of attributes may change in real-life applications, attributes values may be added to or deleted from the domain, Chen et al. proposed the incremental updating approach of approximations while attributes values coarsening or refining in the complete and incomplete information systems [35]. Zhang et al. discussed the change of approximations in neighborhood decision systems when the object set evolves over time, and proposed two fast incremental algorithms for updating approximations when multiple objects enter into or get out of the neighborhood decision table [33]. Li et al. firstly introduced a kind of dominance matrix to calculate P-dominating sets and P-dominated sets in dominance-based rough sets approach, and proposed the incremental algorithms for updating approximations of an upward union and downward union of decision classes [34]. Instead of considering the incremental updating strategies of rough sets, Cheng proposed two incremental methods for fast computing the rough fuzzy approximations, which are established respectively based on the redefined boundary set and the relation between a fuzzy set and its cut sets [41].

However, to our best knowledge, previous studies on incremental computing approximations mainly concerned in the single-valued information systems, but little attention has been paid to the set-valued information systems. Set-valued information systems are an important type of data tables, and generalized models of single-valued information systems [42]. In many practical decision-making issues, set-valued information systems have very wide applications, which can be used in intelligent decision-making and knowledge discovery from information systems with uncertain information and set-valued information. In such systems, some of the attribute values of an object may be set-valued, which are always used to characterize the incomplete information, i.e., the values of some attributes are unknown or multi-values. On the other hand, we often encounter the scenario where the ordering of properties of the considering attributes plays a crucial role in the analysis of information systems. Considering attributes with preference-ordered domains is an important characteristic of multi-attribute decision making problems in practice. Greco et al. proposed the Dominance-based Rough Sets Approach (DRSA) [44,45]. This innovation is mainly based on the substitution of the indiscernibility relation by a dominance relation. Furthermore, Qian et al. established a rough set approach in Set-valued Ordered Information Systems (SOIS) to take into account the ordering properties of attributes in set-valued information systems, and classified the SOIS into two categories: disjunctive and conjunctive systems [43]. Since the characteristics of the set-valued information systems is different from that of single-valued information systems (such as: some of the attribute values for an object are set-valued), the method for knowledge acquisition in the single-valued information systems cannot be applied directly to the set-valued ones. For this reason, the incremental method for updating approximations in the dynamic set-valued information systems is discussed in this paper. In [46], Zhang et al. proposed an incremental method for computing approximations in set-valued information systems under the tolerance relation, when the attribute set varies with time. In this paper, we focus on updating knowledge under the variation of the object set in SOIS. Firstly, we discuss the principles of incremental updating approximations when the objects in the universe change (increase or decrease) dynamically in the conjunctive/disjunctive SOIS. Then two incremental updating algorithms are proposed based on the principles. Finally, the performances of two incremental algorithms are evaluated on a variety of data sets.

The remainder of the paper is organized as follows. In Section 2, some basic concepts of RST in SOIS are introduced. The principles and some illustrated examples for incremental updating approximations with the variation of the object set are presented in Section 3. In Section 4, we propose the incremental algorithms for computing approximations based on the updating principles. Performance evaluations are illustrated in Section 5. The paper ends with conclusions and further research topics in Section 6.

2. Preliminaries

For convenience, some basic concepts of rough sets and SOIS are reviewed in this section [42,43].

A set-valued information system is an ordered quadruple \( S = (U, C \cup \{d\}, V, f) \), where \( U = \{x_1, x_2, \ldots, x_n\} \) is a non-empty finite set of objects, called the universe. \( C \) is a non-empty finite set of condition attributes and \( d \) is a decision attribute with \( C \cap \{d\} = \emptyset \); \( V = V_C \cup V_d \), where \( V \) is the domain of all attributes, \( V_C \) is the domain of all condition attributes and \( V_d \) is the domain of the decision attribute; \( f \) is a mapping from \( U \times (C \cup \{d\}) \) to \( V \) such that \( f: U \times (C \cup \{d\}) \to 2^V \) is a set-valued mapping and \( f: U \times \{d\} \to V_d \) is a single-valued mapping.

In an information system, if the domain (scale) of a condition attribute is ordered according to a decreasing or increasing preference, then the attribute is a criterion.
**Definition 1.** A set-valued information system $S = (U, C \cup \{d\}, V, f)$ is called a SOIS if all condition attributes are criterions.

In real problems, many ways to present the semantic interpretations of set-valued information systems have been provided [47–50]. Qian et al. summarized two types of set-valued information systems with two kinds of semantics, which are known as conjunctive ($\forall x \in U$ and $c \in C, f(x, c)$ is interpreted conjunctively) and disjunctive ($\forall x \in U$ and $c \in C, f(x, c)$ is interpreted disjunctively) set-valued information systems. According to the introduction of the following two dominance relations to these types of set-valued information systems, SOIS can be also classified into two categories: conjunctive and disjunctive SOIS [43].

Assume the domain of a criterion $a \in C$ is completely pre-ordered by an outranking relation $\succsim: x \succsim y$ means “$x$ is at least as good as (outranks) $y$ with respect to criterion $a$”. For a subset of attributes $A \subseteq C$, we define $x \succsim A \iff \forall a \in A, x \succsim a$, which means “$x$ is at least as good as (outranks) $y$ with respect to all attributes in $A$.”

**Definition 2.** Let $S = (U, C \cup \{d\}, V, f)$ be a conjunctive SOIS and $A \subseteq C$. The dominance relation in terms of $A$ is defined as:

$$
R^A_{\succsim} = \{ (y, x) \in U \times U | \exists a \in A, y \succsim a \} \\
R^A_{\succeq} = \{ (y, x) \in U \times U | \exists a \in A, y \succeq a \} \\
$$

**Example 1.** Table 1 illustrates a conjunctive SOIS, where $U = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $C = \{a_1, a_2, a_3, a_4\}$, $d$ is the decision attribute, $V_c = \{a, f, g\}$ and $V = \{1, 2, 4\}$. Here, we can obtain that $f(x_1, a_1) = \{x_1\}$, $f(x_1, a_1) = \{a, f, g\}$. Since $e, f, g \in V_c$, we have $x_2 \succeq a_1, x_1$, that is, $x_2$ is at least as good as $x_1$ with respect to $a_1$.

**Definition 3.** Let $S = (U, C \cup \{d\}, V, f)$ be a disjunctive SOIS and $A \subseteq C$. The dominance relation in terms of $A$ is defined as:

$$
R^A_{\succsim} = \{ (y, x) \in U \times U | f_{\succsim}(y, a) \} \\
R^A_{\succeq} = \{ (y, x) \in U \times U | f_{\succeq}(y, a) \} \\
$$

**Example 2.** Table 2 illustrates a disjunctive SOIS, where $U = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $C = \{a_1, a_2, a_3, a_4\}$, $D = \{d\}$, $V_c = \{0, 1\}$ and $V = \{1, 2, 4\}$. Here, we can obtain that $f(x_1, a_1) = \{1\}$, $f(x_1, a_1) = \{0, 1\}$. Since $\max(x_1, a_1) = 1 \geq \min(x_1, a_1) = 0$, we have $x_1 \succeq a_1, x_2$, that is, $x_1$ is at least as good as $x_2$ with respect to $a_1$.

**Table 1** A conjunctive set-valued ordered information system.

| U    | a_1 | a_2 | a_3 | a_4 | d   |
|------|-----|-----|-----|-----|-----|
| x_1  | {f} | {e} | {f} | {g} | {f} |
| x_2  | {e} | {f} | {e} | {g} | {f} |
| x_3  | {g} | {e} | {f} | {g} | {f} |
| x_4  | {e} | {f} | {e} | {g} | {f} |
| x_5  | {f} | {g} | {f} | {e} | {f} |
| x_6  | {f} | {f} | {e} | {f} | {f} |

**Table 2** A disjunctive set-valued ordered information system.

| U    | a_1 | a_2 | a_3 | a_4 | d   |
|------|-----|-----|-----|-----|-----|
| x_1  | 1   | 0.1 | 0   | 1.2 | 2   |
| x_2  | 0.1 | 2   | 1.2 | 0   | 4   |
| x_3  | 0   | 1.2 | 1   | 0.1 | 2   |
| x_4  | 2   | 1   | 0.1 | 0.1 | 1   |
| x_5  | 1   | 1   | 2   | 2   | 4   |
| x_6  | 0.2 | 1   | 0.1 | 0   | 2   |

Let $U/R^A_{\succsim}$ denote a classification on the universe, which is the family set $|x^A_{\succsim}| \in U$. Any element from $U/R^A_{\succsim}$ is called a dominance class with respect to $A$. Dominance classes in $U/R^A_{\succsim}$ do not constitute a partition of $U$ in general. They constitute a covering of $U$.

**Example 3** (Continuation of Examples 1 and 2). From Table 1,

$$U/R^A_{\succsim} = \{\{x_1\} \cup \{x_2\}, \{x_3\} \cup \{x_4\} \cup \{x_5\} \cup \{x_6\}\}$$

**Definition 4.** Let $S = (U, C \cup \{d\}, V, f)$ be a SOIS. $A \subseteq C$, $\forall d_i \in D$, the lower and upper approximations of $D^A_i$ with respect to the dominance relation $R^A_{\succeq} (A \in \{\land, \lor\})$ are defined respectively as follows:

$$\overline{B}^A_{\succeq}(D^A_i) = \{x \in U | \exists x^A_{\succeq} \subseteq D^A_i \}$$

**Example 4** (Continuation of Example 3).

(1) From Table 1, we have $D = \{D_1, D_2, D_3\}$, where $D_1 = \{x_3, x_4\}$, $D_2 = \{x_1, x_4\}$, $D_3 = \{x_1, x_3\}$. Thus, we get the unions of classes as follows: $D^A_1 = D_1$, $D^A_2 = D_1 \cup D_2$, $D^A_3 = D_2 \cup D_3$. From Definition 4, we have $R^A_{\succeq}(D^A_1) = \{x_3\}$, $\overline{B}^A_{\succeq}(D^A_1) = \{x_1, x_3, x_4\}$, $R^A_{\succeq}(D^A_2) = \{x_1, x_3, x_4\}$, $\overline{B}^A_{\succeq}(D^A_2) = \{x_1, x_3, x_4\}$, $R^A_{\succeq}(D^A_3) = \{x_1, x_3, x_4\}$, $\overline{B}^A_{\succeq}(D^A_3) = \{x_1, x_3, x_4\}$. 

$R^A_{\succeq}(D^A_2) = \{x_1, x_3, x_4\}$, $\overline{B}^A_{\succeq}(D^A_2) = \{x_1, x_3, x_4\}$.
(2) Analogously, from Table 2, we have \( D = \{ D_1, D_2, D_3 \} \), where
\( D_1 = \{ x_1 \}, D_2 = \{ x_2, x_3 \}, D_3 = \{ x_4, x_5 \} \). Thus, we get the unions of classes as follows:
\[
D_1^+ = D_1, \quad D_2^+ = D_1 \cup D_2, \quad D_3^+ = D_2 \cup D_3,
\]
\( D_1^- = D_1 \).
From Definition 4, we have:
\[
\overline{R_{X}^+(D_1)} = \emptyset, \quad \overline{R_{X}^+(D_2)} = \{ x_2, x_5 \}, \quad \overline{R_{X}^+(D_3)} = \{ x_4 \},
\]
\[
\overline{R_{X}^+(D_2)} = \{ x_2, x_3, x_4 \}, \quad \overline{R_{X}^+(D_3)} = \{ x_2, x_3, x_4, x_5 \},
\]
\[
\overline{R_{X}^+(D_3)} = \emptyset, \quad \overline{R_{X}^-(D_2)} = \{ x_2, x_3, x_5 \}.
\]

3. Incremental updating approximations in SOIS when the object set varies with time

With the variation of an information system, the structure of information granules in the information system may vary over time which leads to the change of knowledge induced by RST. For example, let us consider a practical information system from the test for foreign language ability of undergraduates in Shansi University, the test results can be expressed as a set-valued information system where the attributes are all inclusion increasing preferences and the value of each student under each attribute is given by an evaluation expert through a set-value [43]. However, during the process of evaluating the undergraduates language ability, data in an information system does not usually remain a stable condition. Some objects may be inserted into the original information system due to the arrival of the new students. On the other hand, some objects will be deleted from the original information system with the graduation of the senior students. Then the discovered knowledge may become invalid, or some new implicit information may emerge in the whole updated information system. Rather than restarting from scratch by the non-incremental or batch learning algorithm for each update, developing an efficient incremental algorithm to avoid unnecessary computations by utilizing the previous data structures or results is thus desired.

In this section, we discuss the variation of approximations in the dynamic SOIS when the object set evolves over time while the attribute set remains constant. For convenience, we assume the incremental learning process lasts two periods from time \( t \) to time \( t + 1 \). We denote a dynamic SOIS at time \( t \) as \( S = (U, C \cup \{ d, V, f \}) \), and at time \( t + 1 \), with the insertion or deletion of objects, the original SOIS will change into a new one, denoted as \( S' = (U, C \cup \{ d', V', f' \}) \). Similarly, we denote the union of classes and the A-dominating set as \( D_t^+ \) and \( \{ x \}_A^{k+x} \), respectively at time \( t \), which are denoted as \( (D_t^+)' \) and \( \{ (x)_A^{k+x} \}' \), respectively at time \( t + 1 \). According to Definition 4, the lower and upper approximations of \( D_t^+ \) with respect to \( A \subseteq C \) are denoted as \( R_{X}^+(D_t^+) \) and \( R_{X}^-(D_t^+) \), respectively at time \( t \), which are denoted as \( (R_{X}^+(D_t^+))' \) and \( (R_{X}^-(D_t^+))' \), respectively at time \( t + 1 \), respectively.

Here, we only discuss the incremental approach for updating approximations in the cases that a single object enter and go out of the information system. The change of multiple objects can be seen as the cumulative change of a single object. The approximations can be updated step by step through the updating principles in the case that a single object varies.

3.1. Principles for incrementally updating approximations with the deletion of a single object

Given a SOIS \( S = (U, C \cup \{ d, V, f \}) \) at time \( t \), the deletion of object \( x \in U \) (denotes the deleted object) will change the original information granules \( \{ x \}_A^{k+x} \) (\( x \in U \) \( A \subseteq C \)) and the union of decision classes \( D_t^+ \) (\( 1 \leq i \leq t \)). The approximations of \( D_t^+ \) will change accordingly. Here, we discuss the principles for updating approximations of \( D_t^+ \) from two cases:
(1) The deleted object belongs to \( D_t^+ \), i.e., \( x \in D_t^+ \); (2) The deleted object does not belong to \( D_t^+ \), i.e., \( x \notin D_t^+ \).

Case 1: The deleted object \( x \) belongs to \( D_t \), i.e., \( x \in D_t^+ \).

Proposition 1. Let \( S = (U, C \cup \{ d, V, f \}) \) be a SOIS, \( A \subseteq C \). When \( x \in D_t^+ \) is deleted from \( U \), for \( R_X^+(D_t^+) \), we have:

(1) If \( x \in R_X^+(D_t^+) \), then \( R_X^+(D_t^+) = R_X^+(D_t^+) \setminus \{ x \} \);
(2) Otherwise, \( R_X^+(D_t^+) = R_X^+(D_t^+) \).

Proof. When \( x \in D_t^+ \) is deleted from \( U \), we have \( U' = U \setminus \{ x \} \).
(1) If \( x \in R_X^+(D_t^+) \), then \( R_X^+(D_t^+) = R_X^+(D_t^+) \setminus \{ x \} \);
(2) Otherwise, \( R_X^+(D_t^+) = R_X^+(D_t^+) \).

Example 5 (Continuation of Example 4)

(1) For Table 1, according to Proposition 1, we compute the lower approximations of \( D_t^+ \) by deleting \( x_1 \) and \( x_2 \) from \( U \), respectively.
- Assume the object \( x_1 \) is deleted from Table 1, and \( U = U \setminus \{ x_1 \} \). We have \( x_1 \in D_3^+ \) and \( x_1 \notin R_X^+(D_3^+) \). Therefore, \( R_X^+(D_3^+) = \{ x_2, x_3, x_4 \} \).
- Assume the object \( x_2 \) is deleted from Table 1, and \( U = U \setminus \{ x_2 \} \). We have \( x_2 \in D_2^+ \) and \( x_2 \notin R_X^+(D_2^+) \). Therefore, \( R_X^+(D_2^+) = \{ x_4 \} \).

(2) For Table 2, according to Proposition 1, we compute the lower approximations of \( D_t^+ \) by deleting \( x_1 \) and \( x_2 \) from \( U \), respectively.
- Assume the object \( x_1 \) is deleted from Table 2, and \( U = U \setminus \{ x_1 \} \). We have \( x_1 \in D_4^+ \) and \( x_1 \notin R_X^+(D_4^+) \). Therefore, \( R_X^+(D_4^+) = \{ x_2, x_3, x_5 \} \).
- Assume the object \( x_2 \) is deleted from Table 2, and \( U = U \setminus \{ x_2 \} \). We have \( x_2 \in D_2^+ \) and \( x_2 \notin R_X^+(D_2^+) \). Therefore, \( R_X^+(D_2^+) = \{ x_4 \} \).

Proposition 2. Let \( S = (U, C \cup \{ d, V, f \}) \) be a SOIS, \( A \subseteq C \). When \( x \in D_t^+ \) is deleted from \( U \), for \( R_X^+(D_t^+) \), we have
\[
R_X^+(D_t^+) = R_X^+(D_t^+) \setminus \{ x \} \cup K
\]
where \( K = \{ x | x \in \{ x \}_A^{k+x} \cap K' \} \), \( K' = \cup_{x \in D_t^+} \{ x \}_A^{k+x} \).

Proof. According to Definition 4, we have \( R_X^+(D_t^+) = \cup_{x \in D_t^+} \{ x \}_A^{k+x} \).
Thus, when the object \( x \in D_t^+ \) is deleted from \( U \), the A-dominating set \( \{ x \}_A^{k+x} \) should be removed from the upper approximation.
is deleted from Table 1, and $D_2^2 \supseteq [x_3]_{A^2}^P - \{x_4\} = \{x_4, x_2, x_3, x_4, x_6\}$ and $D_2^2 \supseteq [x_6]_{A^2}^P - \{x_6\} = \{x_6\}$. Therefore, $K = \{x_4, x_6\}$ and $R_{\text{SOIS}}^A (D_2^2) = R_{\text{SOIS}}^A (D_2^2) = \{x_4, x_6\}$.

**Proposition 4.** Let $S = (U, C \cup \{d, v, f\})$ be a SOIS, $A \subseteq C$. When the object $x \notin D_2^2$ is deleted from $U$, for $R_{\text{SOIS}}^A (D_2^2)$, we have:

1. If $x \in R_{\text{SOIS}}^A (D_2^2)$, then $R_{\text{SOIS}}^A (D_2^2) = R_{\text{SOIS}}^A (D_2^2) - \{x\}$;  
2. Otherwise, $R_{\text{SOIS}}^A (D_2^2) = R_{\text{SOIS}}^A (D_2^2) - \{x\}$.

**Proof.** According to Definition 4, we have that $R_{\text{SOIS}}^A (D_2^2) = \bigcup_{i \in D_2^2} [x_i]_{A^2}^P$. Since the deleted object $x \notin D_2^2$, there exists an object $x \in D_2^2$ satisfies $x \in [x_i]_{A^2}^P$ if $x \in R_{\text{SOIS}}^A (D_2^2)$. Therefore, when $x$ is deleted, we have $\{x\} \subseteq \bigcup_{i \in D_2^2} [x_i]_{A^2}^P = R_{\text{SOIS}}^A (D_2^2)$. On the other hand, if $x \notin R_{\text{SOIS}}^A (D_2^2)$, we have $\forall x \in D_2^2, x \notin [x_i]_{A^2}^P$. Hence, the upper approximation of $D_2^2$ will remain constant, i.e., $R_{\text{SOIS}}^A (D_2^2) = R_{\text{SOIS}}^A (D_2^2)$.  

**Example 8** (Continuation of Example 4)

(1) For Table 1, according to Proposition 4, we compute the lower approximations of $D_2^2$ by deleting $x_4$ and $x_6$ from $U$, respectively.
- Assume the object $x_3$ is deleted from Table 1, and $U = U - \{x_3\}$. We have $x_2 \notin D_2^2$ and $x_4 \in R_{\text{SOIS}}^A (D_2^2)$. Therefore, $R_{\text{SOIS}}^A (D_2^2) = R_{\text{SOIS}}^A (D_2^2) - \{x_3\} = \{x_4, x_2, x_3, x_4, x_6\}$.  
- Assume the object $x_4$ is deleted from Table 2, and $U = U - \{x_4\}$. We have $x_2 \notin D_2^2$ and $x_4 \in R_{\text{SOIS}}^A (D_2^2)$. Therefore, $R_{\text{SOIS}}^A (D_2^2) = R_{\text{SOIS}}^A (D_2^2) - \{x_4\} = \{x_4, x_2, x_3, x_4, x_6\}$.

(2) For Table 2, according to Proposition 3, we compute the upper approximation of $D_2^2$ by deleting $x_4$ from $U$.

Assume the object $x_4$ is deleted from Table 2, and $U = U - \{x_4\}$. We have $x_4 \notin D_2^2$ and $x_4 \in R_{\text{SOIS}}^A (D_2^2)$. Therefore, $R_{\text{SOIS}}^A (D_2^2) = R_{\text{SOIS}}^A (D_2^2) - \{x_4\} = \{x_4, x_2, x_3, x_4, x_6\}$.

3.2. Principles for incrementally updating approximations with the insertion of a new object

Given a SOIS $(U, C \cup \{d, v, f\})$ at time $t$, when the information system is updated by inserting a new object $x_i$ (where $x_i$ denotes the inserted object) into the universe $U$ at time $t + 1$, two situations may occur: (1) $x_i$ forms a new decision class, i.e., $\forall x \in U, f(x, d) \neq f(x, d_i)$; (2) $x_i$ does not form a new decision class, i.e., $\exists x \in U, f(x, d) = f(x, d_i)$. The difference between the two situations is: in the first situation, in addition to updating the approximations of union of the existing decision classes, we need to compute the approximations for the new decision class. Firstly, for updating the approximations of the union of the existing decision classes $D_i (1 \leq i \leq r)$ when
inserting an object, we discuss the principles through two cases similar to the approach taken in the model of deletion: (1) The inserted object will belong to $D_i^x$, i.e., $x \in D_i$, and (2) the inserted object will not belong to $D_i^x$, i.e., $x \notin D_i$.

To illustrate our incremental methods for updating approximations when inserting a new object into SOIS, two tables (Tables 3 and 4) are given as follows. We assume that the objects in Table 3 will be inserted into Table 1, and the objects in Table 4 will be inserted into Table 2.

**Case 1:** The inserted object $x$ will belong to $D_i$.

**Proposition 5.** Let $S = (U, C \cup \{d\}, V, f)$ be a SOIS, $A \subseteq C$. When $x$ is inserted into $U$, for $R_A^\lessgtr (D_i^x)$, we have:

1. If $(D_i^x) \supseteq [x]_A^{\lessgtr}$, where $(D_i^x)' = D_i^x \cup \{x\}$, then $R_A^\lessgtr (D_i^x)' = R_A^\lessgtr (D_i^x) \cup \{x\}$;
2. Otherwise, $R_A^\lessgtr (D_i^x)' = R_A^\lessgtr (D_i^x)$.

**Proof.** According to Definition 4, we have $\forall x \in D_i^x$, if $[x]_A^{\lessgtr} \subseteq D_i^x$, then $x \in [x]_A^{\lessgtr} \cup \{x\}$. That is, if $D_i^x \supseteq [x]_A^{\lessgtr}$, then $(D_i^x)' \supseteq ([x]_A^{\lessgtr})';$ If $(D_i^x) \not\supseteq [x]_A^{\lessgtr}$, then $(D_i^x)' \not\supseteq ([x]_A^{\lessgtr})'$. It follows that if $x \in R_A^\lessgtr (D_i^x)$, then $x \in R_A^\lessgtr (D_i^x)'$; If $x \notin R_A^\lessgtr (D_i^x)$, then $x \notin R_A^\lessgtr (D_i^x)'$. Therefore, according to Definition 4, if $[x]_A^{\lessgtr} \subseteq (D_i^x)'$, we have $x \in R_A^\lessgtr (D_i^x)'$, and $R_A^\lessgtr ((D_i^x)') = R_A^\lessgtr ((D_i^x)) \cup \{x\}$. Otherwise, $R_A^\lessgtr (D_i^x)' = R_A^\lessgtr (D_i^x)$. □

**Example 9 (Continuation of Example 4)**

1. For Table 1, according to Proposition 5, we compute the lower approximations of $D_i^x$ when the objects $x_7$ and $x_8$ in Table 3 insert into Table 1, respectively.
   - Assume the object $x_7$ in Table 3 is inserted into Table 1, and $U = U \cup \{x_7\}$. Since $f(x_7, d) = 3$, then $(D_i^x) = D_i^x \cup \{x_7\}$. Because of $(D_i^x)' \supseteq [x_7]_A^{\lessgtr} = \{x_7, x_8\}$, we have $R_A^\lessgtr (D_i^x)' = R_A^\lessgtr (D_i^x) \cup \{x_7\} = \{x_7, x_8\}$.
   - Assume the object $x_8$ in Table 3 is inserted into Table 1, and $U = U \cup \{x_8\}$. Since $f(x_8, d) = 3$, then $(D_i^x) = D_i^x \cup \{x_8\}$. Because of $(D_i^x) \not\supseteq [x_8]_A^{\lessgtr} = \{x_7, x_8\}$, we have $R_A^\lessgtr ((D_i^x)) = R_A^\lessgtr ((D_i^x)) = \{x_7, x_8\}$.

**Table 3**

| $U$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ |
|-----|-------|-------|-------|-------|-------|
| $x_7$ | $\{e,f\}$ | $\{e\}$ | $\{e\}$ | $\{e,f,g\}$ | 4 |
| $x_8$ | $\{g\}$ | $\{e,f\}$ | $\{f\}$ | $\{f,g\}$ | 2 |
| $x_9$ | $\{f\}$ | $\{f\}$ | $\{e,f\}$ | $\{e,f,g\}$ | 1 |
| $x_{10}$ | $\{e\}$ | $\{e\}$ | $\{e\}$ | $\{e\}$ | 1 |
| $x_{11}$ | $\{e,f\}$ | $\{f\}$ | $\{e\}$ | $\{e\}$ | 3 |
| $x_{12}$ | $\{e,f\}$ | $\{e,g\}$ | $\{f\}$ | $\{f\}$ | 3 |

**Table 4**

The object inserted into the disjunctive set-valued ordered information system (Table 2).

1. For Table 2, according to Proposition 4, we compute the lower approximation of $D_i^x$ when the objects $x_7$ and $x_8$ in Table 4 insert into Table 2, respectively.
   - Assume the object $x_7$ in Table 4 is inserted into Table 2, and $U = U \cup \{x_7\}$. Since $f(x_7, d) = 2$, then $(D_i^x) = D_i^x \cup \{x_7\}$. Because of $(D_i^x) \not\supseteq [x_7]_A^{\lessgtr} = \{x_7, x_8\}$, we have $R_A^\lessgtr (D_i^x) = R_A^\lessgtr (D_i^x) \cup \{x_7\} = \{x_7, x_8\}$.
   - Assume the object $x_8$ in Table 4 is inserted into Table 2, and $U = U \cup \{x_8\}$. Since $f(x_8, d) = 2$, then $(D_i^x) = D_i^x \cup \{x_8\}$. Because of $(D_i^x) \not\supseteq [x_8]_A^{\lessgtr} = \{x_7, x_8\}$, we have $R_A^\lessgtr (D_i^x) = R_A^\lessgtr (D_i^x) \cup \{x_7\} = \{x_7, x_8\}$.

**Proposition 6.** Let $S = (U, C \cup d, V, f)$ be a SOIS, $A \subseteq C$. When $x$ is inserted into $U$, for $R_A^\lessgtr (D_i^x)$, we have

$R_A^\lessgtr (D_i^x)' = R_A^\lessgtr (D_i^x) \cup \{x\}$.

**Proof.** When the object $x$ is inserted into $U$, $U = U \cup \{x\}$. According to Definition 4, we have $R_A^\lessgtr (D_i^x)' = \bigcup_{x \in D_i^x} [x]_A^{\lessgtr} \cup \{x\}$. Since $(D_i^x)' = D_i^x \cup \{x\}$, then we have $R_A^\lessgtr (D_i^x)' = \bigcup_{x \in D_i^x} [x]_A^{\lessgtr} \cup \{x\}$.

Because $\forall x \in U$, $([x]_A^{\lessgtr}) = \{x\}$ or $([x]_A^{\lessgtr}) = \{x\}$, and $x \in [x]_A^{\lessgtr}$, we can obtain that $R_A^\lessgtr (D_i^x)' = \bigcup_{x \in D_i^x} [x]_A^{\lessgtr} \cup \{x\}$. □

**Example 10 (Continuation of Example 4)**

1. For Table 1, according to Proposition 6, we compute the upper approximations of $D_i^x$ when the objects $x_7$ in Table 3 inserts into Table 1.
   - Assume the object $x_7$ in Table 3 inserts into Table 1, and $U = U \cup \{x_7\}$. Since $f(x_7, d) = 3$, then $(D_i^x) = D_i^x \cup \{x_7\}$. Because of $(D_i^x) \supseteq [x_7]_A^{\lessgtr} = \{x_7, x_8\}$, we have $R_A^\lessgtr (D_i^x)' = R_A^\lessgtr (D_i^x) \cup \{x_7\} = \{x_7, x_8\}$.

2. For Table 2, according to Proposition 6, we compute the upper approximations of $D_i^x$ when the object $x_7$ in Table 4 inserts into Table 2.
   - Assume the object $x_7$ in Table 4 inserts into Table 2, and $U = U \cup \{x_7\}$. Since $f(x_7, d) = 2$, then $(D_i^x) = D_i^x \cup \{x_7\}$. Because of $(D_i^x) \not\supseteq [x_7]_A^{\lessgtr} = \{x_7, x_8\}$, we have $R_A^\lessgtr (D_i^x)' = R_A^\lessgtr (D_i^x) \cup \{x_7\} = \{x_7, x_8\}$.

**Case 2:** The inserted object $x$ will not belong to $D_i$.

**Proposition 7.** Let $S = (U, C \cup d, V, f)$ be a SOIS, $A \subseteq C$. When $x$ is inserted into $U$, for $R_A^\lessgtr (D_i^x)$, we have

$R_A^\lessgtr (D_i^x)' = R_A^\lessgtr (D_i^x) \cup \{x\}$.
where $K = \{x | x \in R^{\alpha_3}_A(D^*_0), \bar{x} \in \{[x]^{\alpha_3}_A\}\}$.

**Proof.** When the object $\bar{x}$ is inserted into $U$, since $\bar{x} \notin x (x \in D_i)$, we have $U' = U \cup \{x\}$, $(D'_0)^* = D^*_0$. Since $x \in (D'_0)^*$, $(\{x\}^{\alpha_3}_A)^* = \{x\}^{\alpha_3}_A$ or $(\{x\}^{\alpha_3}_A)^* = \{x\}^{\alpha_3}_A \cup \{x\}$. We have if $\{x\}^{\alpha_3}_A \not\subseteq D^*_0$, then $\{x\}^{\alpha_3}_A \not\subseteq (D'_0)^*$. That is, if $x \not\in R^{\alpha_3}_A(D^*_0)$, then $x \not\in R^{\alpha_3}_A(D'_0)$. Hence, we only consider the object $x \in R^{\alpha_3}_A(D^*_0)$, i.e., $\{x\}^{\alpha_3}_A \subseteq D^*_0$. When $\bar{x}$ is deleted, there may exist that $(\{x\}^{\alpha_3}_A)^* = \{x\}^{\alpha_3}_A \cup \{x\}$. Then $(\{x\}^{\alpha_3}_A)^* \not\subseteq (D'_0)^*$, i.e., $x \not\in R^{\alpha_3}_A(D'_0)$. Therefore, we have $R^{\alpha_3}_A(D'_0)^* = R^{\alpha_3}_A(D^*_0) - K$, where $K = \{x | x \in R^{\alpha_3}_A(D^*_0), \bar{x} \in \{[x]^{\alpha_3}_A\}\}$. $\square$

**Example 11** (Continuation of Example 4)

(1) For Table 1, according to Proposition 7, we compute the lower approximations of $D^*_2$ when the object $x_9$ in Table 3 inserts into Table 1.

Assume the object $x_9$ in Table 3 inserts into Table 1, and $U = U \cup \{x_9\}$. Since $f(x_9,d) = 1$, then $D^*_2$ remains unchanged. Because of $\forall x \in D^*_2 = \{x_1,x_2,x_3,x_4,x_5\}$, $x_9 \notin x$, that is, $x_9 \notin \{[x]^{\alpha_3}_C\}$. Hence $R^{\alpha_3}_C(D^*_2)^* = R^{\alpha_3}_C(D^*_2) \cup \{x_9\} = \{x_1,x_2,x_3,x_4,x_5\}$.

(2) For Table 2, according to Proposition 8, we compute the upper approximations of $D^*_2$ when the object $x_9$ in Table 4 inserts into Table 2.

Assume the object $x_9$ in Table 4 inserts into Table 2, and $U = U \cup \{x_9\}$. Since $f(x_9,d) = 1$, then $D^*_2$ remains unchanged. Because of $\forall x \in D^*_2 = \{x_1,x_2,x_3,x_5\}$, $x_9 \subset x$, that is, $x_9 \notin \{[x]^{\alpha_3}_C\}$. Hence $R^{\alpha_3}_C(D^*_2)^* = R^{\alpha_3}_C(D^*_2)$.

Based on the above analysis, we can compute the approximations of the union of existing decision classes $D^*_i (1 \leq i \leq r)$ when inserting a new object into SOIS. However, when a new object is inserted into the universe, it might happen that $\bar{x}$ will form a new decision class, i.e., $\forall x \in U, f(\bar{x}, d) \neq f(x,d)$. Then the universe $U = U \cup \{\bar{x}\}$ will be divided into $r + 1$ partitions, such as: $D = \{D_1, \ldots, D_r, D_{new}, D_{new,1}, \ldots, D_{new,r}\}$, where $|D| = r + 1, D_{new} = \{\bar{x}\}$. At this point, in addition to updating the approximations of the existing unions of decision classes, we need to compute the unions of new decision class $D_{new}$: $D_{new}^* = D_{new,1}^* \cup \{\bar{x}\}$.

**Proposition 9.** Let $S = (U \cup D, U \cup D, f)$ be a SOIS, $A \subseteq C$. When $\bar{x}$ is inserted into $U$, if $\forall x \in U, f(\bar{x}, d) \neq f(x,d)$, then the lower approximation of the union of the new decision class $D^*_2$ can be computed as follows:

1. If $\{x\}^{\alpha_3}_A \not\subseteq D^*_2$ then $R^{\alpha_3}_A(D^*_2)^* = R^{\alpha_3}_A(D^*_2) \cup \{\bar{x}\}$
2. Otherwise, $R^{\alpha_3}_A(D^*_2)^* = R^{\alpha_3}_A(D^*_2)$.

**Proof.** When the object $\bar{x}$ is inserted into $U$, since $\bar{x} \notin x (x \in D_i)$, we have $U' = U \cup \{x\}$. $(D'_0)^* = D^*_0$. Then, $\forall x \in (D'_0)^*$, if $x \not\in \{x\}^{\alpha_3}_A$, then $(\{x\}^{\alpha_3}_A)^* = \{x\}^{\alpha_3}_A \cup \{x\}$. According to Definition 4, we have if $x \in R^{\alpha_3}_A(D^*_0)$, that is, $R^{\alpha_3}_A(D^*_0)^* = R^{\alpha_3}_A(D^*_0) \cup \{\bar{x}\}$; Otherwise, if $x \notin R^{\alpha_3}_A(D^*_0)$, that is, $(\{x\}^{\alpha_3}_A)^* = \{x\}^{\alpha_3}_A$. Then we have $R^{\alpha_3}_A(D^*_0)^* = R^{\alpha_3}_A(D^*_0)$. $\square$

**Example 12** (Continuation of Example 4)

(1) For Table 1, according to Proposition 8, we compute the lower approximations of $D^*_2$ when the object $x_9$ and $x_{10}$ in Table 3 insert into Table 1, respectively.

- Assume the object $x_9$ in Table 3 inserts into Table 1, and $U = U \cup \{x_9\}$. Since $f(x_9,d) = 1$, then $D^*_2$ remains unchanged. Because of $\forall x \in D^*_2 = \{x_1,x_2,x_3,x_4,x_5\}$, $x_9 \subset x_9$, that is, $x_9 \notin \{[x]^{\alpha_3}_C\}$.

(2) For Table 2, according to Proposition 8, we compute the upper approximations of $D^*_2$ when the object $x_9$ and $x_{10}$ in Table 4 insert into Table 2, respectively.

- Assume the object $x_9$ in Table 4 inserts into Table 2, and $U = U \cup \{x_9\}$. Since $f(x_9,d) = 1$, then $D^*_2$ remains unchanged. Because of $\forall x \in D^*_2 = \{x_1,x_2,x_3,x_5\}$, $x_9 \subset x_9$, that is, $x_9 \notin \{[x]^{\alpha_3}_C\}$. Hence $R^{\alpha_3}_C(D^*_2)^* = R^{\alpha_3}_C(D^*_2)$.

Example 13 (Continuation of Example 4)

(1) For Table 1, according to Proposition 9, we compute the lower approximations of $D_{new}$ when the object $x_{11}$ and $x_{12}$ in Table 3 insert into Table 1, respectively.
• Assume the object $x_{11}$ in Table 3 inserts into Table 1, and $U' = U \cup \{x_{11}\}$. Since $\forall x \in U, f(x,d) \neq f(x_{11},d)$, and $f(D_{\text{new}}) < f(x_{11},d) < f(D_{\text{new}})$, we have $D_{\text{new}} = D_{\text{old}}^x$.

• Assume the object $x_{12}$ in Table 3 inserts into Table 1, and $U' = U \cup \{x_{12}\}$. Since $\forall x \in U, f(x,d) \neq f(x_{12},d)$ and $f(D_{\text{new}}) < f(x_{12},d) < f(D_{\text{new}})$, we have $D_{\text{new}} = D_{\text{old}}^x$.

(2) For Table 2, according to Proposition 9, we compute the lower approximations of $D_{\text{new}}$ when the object $x_{11}$ and $x_{12}$ in Table 4 are respectively inserted into Table 2.

• Assume the object $x_{11}$ in Table 4 inserts into Table 1, and $U' = U \cup \{x_{11}\}$. Since $\forall x \in U, f(x,d) \neq f(x_{11},d)$ and $f(D_{\text{new}}) < f(x_{11},d) < f(D_{\text{new}})$, then $D_{\text{new}} = D_{\text{old}}^x$. Because of $[x_{11}]_{A^{\text{old}}} \subseteq D_{\text{new}}$, we have $R_{A^{\text{old}}}^x(D_{\text{new}}) = R_{A^{\text{old}}}^x(D_{\text{old}}^x) \cup \{x_{11}\}$.

• Assume the object $x_{12}$ in Table 4 inserts into Table 1, and $U' = U \cup \{x_{12}\}$. Since $\forall x \in U, f(x,d) \neq f(x_{12},d)$ and $f(D_{\text{new}}) < f(x_{12},d) < f(D_{\text{new}})$, then $D_{\text{new}} = D_{\text{old}}^x$. Because of $[x_{12}]_{A^{\text{old}}} \not\subseteq D_{\text{new}}$, we have $R_{A^{\text{old}}}^x(D_{\text{new}}) = R_{A^{\text{old}}}^x(D_{\text{old}}^x) = \emptyset$.

**Proposition 10.** Let $S = (U, C \cup d, V, f)$ be a SOIS, $A \subseteq C$. When $\tilde{x}$ is inserted into $U$, if $\forall x \in U, f(x, \tilde{d}) = f(x, d)$, then the upper approximation of the union of the new decision class $D_{\text{new}}$ can be computed as follows:

$$R_{A^{\text{old}}}^x(D_{\text{new}}) = R_{A^{\text{old}}}^x(D_{\text{old}}^x) \cup [x_{11}]_{A^{\text{old}}}$$

where $D_{\text{new}} = D_{\text{old}}^x \cup \{\tilde{x}\}$.

**Proof.** When the object $\tilde{x}$ inserts into $U$, $U' = U \cup \{\tilde{x}\}$. According to Definition 4, we have $R_{A^{\text{old}}}^{\text{new}}(D_{\text{new}}) = \bigcup_{x_{11} \in U'} [x_{11}]_{A^{\text{old}}}$. Since $D_{\text{new}} = D_{\text{old}}^x \cup \{\tilde{x}\}$, then $R_{A^{\text{old}}}^{\text{new}}(D_{\text{new}}) = \bigcup_{x_{11} \in U'} [x_{11}]_{A^{\text{old}}}$. Because $\forall x \in U$, $[x_{11}]_{A^{\text{old}}} = [x_{11}]_{A^{\text{old}}} \cup \{\tilde{x}\}$ or $[x_{11}]_{A^{\text{old}}} = [x_{11}]_{A^{\text{old}}}$ and $\tilde{x} \in [x_{11}]_{A^{\text{old}}}$, we can obtain that $R_{A^{\text{old}}}^{\text{new}}(D_{\text{new}}) = \bigcup_{x_{11} \in U'} [x_{11}]_{A^{\text{old}}} \cup [x_{11}]_{A^{\text{old}}} = R_{A^{\text{old}}}^{\text{new}}(D_{\text{old}}^x) \cup [x_{11}]_{A^{\text{old}}}$. □

**Example 14 (Continuation of Example 4)**

(1) For Table 1, according to Proposition 9, we compute the lower approximations of $D_{\text{new}}$ when the object $x_{11}$ in Table 3 inserts into Table 1.

Assume the object $x_{11}$ in Table 3 inserts into Table 2, and $U' = U \cup \{x_{11}\}$. Since $\forall x \in U, f(x,d) \neq f(x_{11},d)$ and $f(D_{\text{new}}) < f(x_{11},d)$, then $D_{\text{new}} = D_{\text{old}}^x$. Because of $[x_{11}]_{C^{\text{old}}} \subseteq D_{\text{new}}$, we have $R_{C^{\text{old}}}^x(D_{\text{new}}) = R_{C^{\text{old}}}^x(D_{\text{old}}^x) \cup \{x_{11}\}$.

(2) For Table 2, according to Proposition 9, we compute the lower approximations of $D_{\text{new}}$ when the object $x_{11}$ in Table 4 inserts into Table 2.

Assume the object $x_{11}$ in Table 4 inserts into Table 2, and $U' = U \cup \{x_{11}\}$. Since $\forall x \in U, f(x,d) \neq f(x_{11},d)$ and $f(D_{\text{new}}) < f(x_{11},d)$, then $D_{\text{new}} = D_{\text{old}}^x$. Because of $[x_{11}]_{C^{\text{old}}} \not\subseteq D_{\text{new}}$, we have $R_{C^{\text{old}}}^x(D_{\text{new}}) = R_{C^{\text{old}}}^x(D_{\text{old}}^x) = \emptyset$.

4. **Static (non-incremental) and incremental algorithms for computing approximations in SOIS with the variation of the object set**

In this section, we design static and incremental algorithms on the variation of the object set in SOIS corresponding to Sections 2 and 3, respectively.

---

**Table 5**

A description of data sets.

| Data sets      | Abbreviation | Samples | Attributes | Classes | Source |
|----------------|--------------|---------|------------|---------|--------|
| 1              | Mushroom     | 8124    | 22         | 2       | UCI    |
| 2              | Congressional voting records | 435     | 16         | 2       | UCI    |
| 3              | Audiology (Standardized) | 226     | 69         | 24      | UCI    |
| 4              | Dermatology  | 336     | 34         | 6       | UCI    |
| 5              | Artificial data 1 | 10,000  | 5          | 5       | Data generator |
| 6              | Artificial data 2 | 5000    | 8          | 5       | Data generator |

**Table 6**

A comparison of static and incremental algorithms versus different data sizes when deleting the objects.

| Num. of data sets | Mushroom | CVR | Audiology | Dermatology | AD1 | AD2 |
|-------------------|---------|-----|-----------|-------------|-----|-----|
|                   | Static  | Incre.| Static    | Incre.      | Static | Incre. |
| 1 st              | 0.017   | 0.001| 0.008     | 0.003       | 0.015 | 0.005|
| 2nd               | 0.244   | 0.003| 0.016     | 0.006       | 0.016 | 0.007|
| 3rd               | 0.838   | 0.006| 0.025     | 0.009       | 0.022 | 0.008|
| 4th               | 1.982   | 0.010| 0.034     | 0.011       | 0.034 | 0.014|
| 5th               | 3.991   | 0.016| 0.047     | 0.014       | 0.032 | 0.011|
| 6th               | 6.887   | 0.025| 0.071     | 0.021       | 0.047 | 0.015|
| 7th               | 11.272  | 0.035| 0.088     | 0.025       | 0.070 | 0.020|
| 8th               | 17.559  | 0.050| 0.135     | 0.032       | 0.100 | 0.027|
| 9th               | 24.083  | 0.067| 0.161     | 0.042       | 0.115 | 0.025|
| 10th              | 34.505  | 0.083| 0.199     | 0.052       | 0.144 | 0.033|

(Question: What is the purpose of the static and incremental algorithms discussed in the text?)

**Answer:** The purpose of the static and incremental algorithms is to compute the lower approximations of decision classes in SOIS with the variation of the object set, which is crucial for managing data in a dynamic environment where objects may be inserted or deleted.
4.1. The static algorithm for computing approximations in SOIS

Algorithm 1 is a static (non-incremental) algorithm for computing the lower and upper approximations in SOIS while the object set in the information system is changed. In Step 2, we compute all the decision classes, and the set of decision classes are preference-ordered according to the increasing order of class indices. Step 3–7 compute all the upward unions of classes based on the set of decision classes. Step 9–11 compute all the A-dominating sets. Step 12–21 compute the lower and upper approximations in SOIS based on Definition 4.

Algorithm 1: The static algorithm for computing approximations in SOIS

\begin{verbatim}
Input: A SOIS \( S = (U, C \cup \{d\}, V, f) \).
Output: The lower and upper approximations in SOIS.

begin
  Compute \( D = \{D_1, D_2, \ldots, D_r\} \); \hfill // Compute all the decision classes, where \( r \) is the total number of the decision classes, and the classes from \( D \) are preference ordered according to increasing order of class indices.
  for \( 1 \leq i \leq r \) do
    Compute \( D_i^c = D_i^+ \cup D_i \); \hfill // Compute all the unions of class \( D_i \), \( 1 \leq i \leq r \);
  end
  Let \( R_{A_d}^\Delta(D_i^c) \leftarrow 0 \), \( R_{A_d}^\Sigma(D_i^c) \leftarrow 0 \);
  for each \( x \in U \) do
    Compute \( \{x\}_A^\Delta \); \hfill // Compute all the A-dominating sets with respect to \( \forall x \in U \);
    end
  for \( 1 \leq i \leq r \) do
    for each \( x \in U \) do
      if \( \{x\}_A^\Delta \subseteq D_i^c \) then
        \( R_{A_d}^\Sigma(D_i^c) = R_{A_d}^\Sigma(D_i^c) \cup \{x\} \);
      end
      if \( x \in D_i^c \) then
        \( R_{A_d}^\Delta(D_i^c) = R_{A_d}^\Delta(D_i^c) \cup \{x\}_A^\Delta \);
      end
    end
  end
  return \( R_{A_d}^\Delta(D_i^c), R_{A_d}^\Sigma(D_i^c) \).
end
\end{verbatim}

Fig. 1. A comparison of static (Algorithm 1) and incremental (Algorithm 2) algorithms versus the data sizes when inserting the objects.
Table 7
A comparison of static and incremental algorithms versus different updating rates when deleting the objects.

| Delete rate (%) | Mushroom | CVR | Audiology | Dermatology | AD1 | AD2 |
|-----------------|----------|-----|-----------|-------------|-----|-----|
|                 | Static   | Incre. | Static   | Incre. | Static   | Incre. | Static   | Incre. | Static   | Incre. | Static   | Incre. |
| 10              | 28.725   | 0.146 | 0.166    | 0.046 | 0.116    | 0.033 | 32.22    | 0.033 | 45.715   | 0.165 | 5.377    | 0.165 |
| 20              | 20.562   | 0.143 | 0.134    | 0.031 | 0.098    | 0.038 | 0.074    | 0.030 | 0.273    | 0.029 | 3.148    | 0.131 |
| 30              | 16.268   | 0.114 | 0.098    | 0.028 | 0.028    | 0.021 | 0.010    | 0.016 | 0.138    | 0.021 | 2.136    | 0.097 |
| 40              | 8.695    | 0.146 | 0.076    | 0.011 | 0.068    | 0.025 | 0.072    | 0.019 | 0.080    | 0.019 | 1.354    | 0.073 |
| 50              | 2.847    | 0.123 | 0.023    | 0.029 | 0.022    | 0.020 | 0.017    | 0.016 | 0.322    | 0.033 | 1.636    | 0.077 |
| 60              | 1.144    | 0.125 | 0.015    | 0.027 | 0.018    | 0.024 | 0.054    | 0.015 | 0.440    | 0.024 | 0.868    | 0.057 |
| 70              | 0.325    | 0.125 | 0.014    | 0.029 | 0.031    | 0.015 | 0.044    | 0.015 | 0.044    | 0.015 | 0.176    | 0.030 |
| 80              | 0.025    | 0.161 | 0.008    | 0.027 | 0.018    | 0.031 | 0.015    | 0.031 | 0.044    | 0.016 | 0.004    | 0.031 |

**Algorithm 2**: An incremental algorithm for updating approximations when deleting an object from the universe in SOIS

Input:
1. Original A-dominating sets at time t: \( U/R_A^{\Delta} = (\{x_1\}^{\Delta_A}, \{x_2\}^{\Delta_A}, \ldots, \{x_n\}^{\Delta_A}) \), where \( A \subseteq C, n = |U| \);
2. Original unions of decision classes at time t: \( D_i^\gamma \), where \( D = \{D_1, D_2, \ldots, D_r\} \), and \( 1 \leq i \leq r \);
3. Original lower and upper approximations at time t: \( R_A^{\Delta}(D_i^\gamma), \bar{R}_A^{\Delta}(D_i^\gamma) \);
4. The object being deleted from U in SOIS: \( \bar{x} \).

Output: The updated lower and upper approximations at time \( t + 1 \) after the deletion of \( \bar{x} \) from \( U: \overline{R_A^{\Delta}(D_i^\gamma)}, \overline{\bar{R}_A^{\Delta}(D_i^\gamma)} \).

```
begin
for 1 \leq i \leq r do
if \( \bar{x} \in D_i^\gamma \) then // Update the approximations on the case of the deleted object \( \bar{x} \) belongs to \( D_i^\gamma \):
if \( \bar{x} \in \overline{R_A^{\Delta}(D_i^\gamma)} \) then // Update the lower approximations by Proposition 1:
\[ \overline{R_A^{\Delta}(D_i^\gamma)}' = \overline{R_A^{\Delta}(D_i^\gamma)} - \{\bar{x}\} \]
else
\[ \overline{R_A^{\Delta}(D_i^\gamma)}' = \overline{R_A^{\Delta}(D_i^\gamma)} \]
end
\[ \overline{\bar{R}_A^{\Delta}(D_i^\gamma)}' = \overline{\bar{R}_A^{\Delta}(D_i^\gamma)} - \{\bar{x}\} \] // Update the upper approximations by Proposition 2;
for each \( x \in \{x\}^{\Delta_A} \) do
for each \( x' \in D_i^\gamma - \{x\} \) do
if \( x \in \{x'\}^{\Delta_A} \) then
\[ \overline{R_A^{\Delta}(D_i^\gamma)}' = \overline{R_A^{\Delta}(D_i^\gamma)}' \cup \{x\} \]
end
end
end
else // Update the approximations in the case of the deleted object \( \bar{x} \) does not belong to \( D_i^\gamma \):
Let \( R_A^{\Delta}(D_i^\gamma)' = R_A^{\Delta}(D_i^\gamma) \); // Update the lower approximations by Proposition 3;
for each \( x \in \{D_i^\gamma - R_A^{\Delta}(D_i^\gamma)\} \) do
if \( \bar{x} \in \{x\}^{\Delta_A} \) then
\[ \{x\}^{\Delta_A} = \{x\}^{\Delta_A} - \{\bar{x}\} \]
end
if \( D_i^\gamma \supseteq \{x\}^{\Delta_A} \) then
\[ \overline{R_A^{\Delta}(D_i^\gamma)}' = \overline{R_A^{\Delta}(D_i^\gamma)}' \cup \{x\} \]
end
end
if \( \bar{x} \in \overline{\bar{R}_A^{\Delta}(D_i^\gamma)} \) then // Update the upper approximations by Proposition 4:
\[ \overline{\bar{R}_A^{\Delta}(D_i^\gamma)}' = \overline{\bar{R}_A^{\Delta}(D_i^\gamma)} - \{\bar{x}\} \]
else
\[ \overline{\bar{R}_A^{\Delta}(D_i^\gamma)}' = \overline{\bar{R}_A^{\Delta}(D_i^\gamma)} \]
end
end
return \( \overline{R_A^{\Delta}(D_i^\gamma)}, \overline{\bar{R}_A^{\Delta}(D_i^\gamma)} \).
end
```
4.2. The incremental algorithm for updating approximations in SOIS when deleting an object from the universe

Algorithm 2 is an incremental algorithm for updating approximations in SOIS while deleting an object from the universe. Step 3–16 update the approximations of the union of classes $D^p_i$, when the deleted object $x$ belongs to the union of classes $D^p_i$. Step 19–27 compute the lower approximations of $D^p_i$ by Proposition 1. Step 28–33 compute the upper approximations of $D^p_i$ by Proposition 2.

4.3. The incremental algorithm for updating approximations in SOIS when inserting an object into the universe

Algorithm 3 is an incremental algorithm for updating approximations in SOIS while inserting an object into the universe. Step 2 compute the $A$-dominating set with respect to the inserted object $\tilde{x}$. Step 3–25 update the approximations of the union of classes $D^p_i$, when the inserted object $\tilde{x}$ will belong to the union of classes $D^p_i$. Step 28–33 compute the upper approximations of $D^p_i$ by Proposition 3.
Step 5–10 compute the lower approximations of $D_i^n$ by Proposition 5. Step 11 compute the upper approximation of $D_i^n$ by Proposition 6. Step 13–24 update the approximations of the union of classes $D_i^n$, when the inserted object $x$ will not belong to the union of classes $D_i^n$. Step 13–18 compute the lower approximations of $D_i^n$ by Proposition 7. Step 19–24 update the approximations of $D_i^n$ by Proposition 8. Step 26–35 compute the approximation of the union of new decision class $D_n^{new}$, if the inserted object does not belong to any existed decision classes. Step 29–33 compute the lower approximation of $D_i^{new}$ by Proposition 9. Step 34 compute the upper approximation of $D_i^{new}$ by Proposition 10.

5. Experimental evaluations

In this section, in order to evaluate the performance of the proposed incremental algorithms, we conduct a series of experiments to compare the computational time between the non-incremental algorithm and the incremental algorithms for computing approximations based on standard data sets.

The algorithms are implemental using the JAVA programming language in Eclipse 3.5 with Java Virtual Machine (JVM) 1.6 (available at http://www.eclipse.org/platform). Experiments are performed on a computer with 2.66 GHz CPU, 4.0 GB of memory and 32-bit Windows 7 OS. We download four data sets from the machine learning data repository, University of California at Irvine [51], where the basic information of data sets is outlined in Table 5. The data sets 1–4 in Table 5 are all incomplete information systems with missing values. In our experiment, we represent all the missing values by the set of all possible values of each attribute. Then this type of data sets can be regarded as a special case of the set-valued information system. Besides, we also use the set-valued data generator to generate two artificial data sets 5–6 in order to test the efficiency of the proposed algorithms, which are also outlined in Table 5.

Generally, we perform the experimental analysis with applying the non-incremental algorithm along with our proposed incremental algorithms when the objects inserting into or deleting from the information system, respectively. In order to present more informative comparative data and acquire more dependable results in our experiments, we compare the computational efficiency of the algorithms according to the following two aspects:

1. Size of the data set: To compare the computational efficiency and distinguish the computational times used by the non-incremental and incremental algorithms with different-sized data sets, we divide each of the six data sets into 10 parts of equal size, respectively. The first part is regarded as the 1st data set, the combination of the first part and the second part is viewed as the 2nd data set, the combination of the 2nd data set and the third part is regarded as the 3rd data set, and so on. The combination of all ten parts is viewed as the 10th data set.

2. Update ratio of the data set: The size of updated objects which inserting into or deleting from the universe may different, that is, the update ratio, i.e., the ratio of the number of updating (deleting or inserting) data and original data, may different. Here, in order to analyze the influence of the update ratio on the efficiency of algorithms, we compare the computational time of the static and incremental algorithms with different update ratios. That is to say, for each data sets, we conduct the comparison experiments with the same original data size, but different update ratios, i.e., deleting ratios and inserting ratios.

5.1. A comparison of computational efficiency between static and incremental algorithms with the deletion of the objects

To compare the efficiency of static (Algorithm 1) and incremental (Algorithm 2) algorithms for computing approximations when
deleting the objects from the data sets. Firstly, we compare the two algorithms on the six data sets in Table 5 with the same updating ratio (the ratio of the number of deleting data and original data), but different sizes of the original data. Here, we assume that the updating ratio is equal to 5%. The experimental results are shown in Table 6. More detailed changing trendline of each of two algorithms with the increasing size of data sets are illustrated in Fig. 1. Secondly, we compare the computational time of the two algorithms with the same size of original data, but different updating ratios for each data sets (from 5% to 100%). we show the experimental results in Table 7. More detailed changing trendline of each of two algorithms with the increasing updating ratio of data sets are presented in Fig. 2.

In each sub-figures (a)–(f) of Fig. 1, the x-coordinate pertains to the size of the data set (the 10 data sets starting from the smallest one), while the y-coordinate presents the computational time. We use the star lines to denote the computational time of the static algorithm on different sizes of data sets, and the plus lines denote the computational time of the incremental algorithm on different sizes of data sets when deleting the objects into the universe. It is easy to see the computational time of the both algorithms usually increases with the increase of the size of data sets according to Table 6 and Fig. 1. As the important advantage of the incremental algorithm shown in Table 6 and Fig. 1, when deleting the objects from the universe, we find that the incremental algorithm is much faster than the static algorithm for computing the approximations. Furthermore, the differences become larger and larger when increasing the size of data sets.

In each sub-figures (a)–(f) of Fig. 2, the x-coordinate pertains to the ratio of the number of the deleting data and original data, while the y-coordinate concerns the computational time. According to the experimental results in Table 7 and Fig. 2, we find that, for the static algorithm, the computational time for computing approximations with deletion of the objects from the universe is decreasing monotonically along with the increase of deleting ratios. It is because with the increase of ratios, the size of the universe decreases gradually. On the contrary, for incremental algorithm, we can see that the computational efficiency for com-

| Num. of data sets | Mushroom | CVR | Audiology | Dermatology | AD1 | AD2 |
|-------------------|----------|-----|-----------|-------------|-----|-----|
| 1st               | 0.023    | 0.001 | 0.003     | 0.008       | 0.005 | 0.005 |
| 2nd               | 0.284    | 0.016 | 0.017     | 0.005       | 0.014 | 0.007 |
| 3rd               | 1.059    | 0.053 | 0.024     | 0.007       | 0.018 | 0.012 |
| 4th               | 2.656    | 0.132 | 0.034     | 0.010       | 0.024 | 0.010 |
| 5th               | 5.441    | 0.259 | 0.057     | 0.012       | 0.033 | 0.010 |
| 6th               | 9.085    | 0.480 | 0.081     | 0.013       | 0.046 | 0.012 |
| 7th               | 18.200   | 1.238 | 0.102     | 0.016       | 0.061 | 0.014 |
| 8th               | 24.843   | 1.576 | 0.141     | 0.021       | 0.079 | 0.017 |
| 9th               | 31.714   | 1.805 | 0.191     | 0.025       | 0.100 | 0.020 |
| 10th              | 60.312   | 2.576 | 0.247     | 0.038       | 0.133 | 0.025 |

Table 8: A comparison of static and incremental algorithms versus different data sizes when inserting the objects.

Fig. 3. A comparison of static (Algorithm 1) and incremental (Algorithm 3) versus the size of data when inserting the objects.
puting approximations is changing smoothly along with the increase of deleting ratios. It is easy to find out the incremental algorithm always performs faster than the non-incremental algorithm for computing approximations until a threshold of the deleting ratio. The threshold differs depending on the data sets. For example, in Fig. 2(a), (e), and (f), the thresholds of ratios are around 85%; in Fig. 2(b) and (c), the thresholds of ratios are around 65%; in Fig. 2(d), the incremental algorithm consistently outperforms the static algorithm even in the value of 90%.

5.2. A comparison of computational efficiency between static and incremental algorithms with the insertion of the objects

Similar to the experiment schemes for comparing the efficiencies between static and incremental algorithms when deleting the objects from the universe, we also adopt such schemes to compare the performance of algorithms on the case of inserting the objects into the universe. Firstly, we compare the two algorithms, i.e., Algorithm 1 and Algorithm 3, on the six data sets in Table 5 with the same updating ratio (the ratio of the number of inserting data and original data), but different sizes of the original data. Here, we assume the updating ratio is equal to 5%. The experimental results are shown in Table 8. More detailed change trendline of each of two algorithms with the increasing size of data sets are presented in Fig. 3. Secondly, we compare the computational times of the two algorithms with the changing of updating ratios for each data sets. We show the experimental results in Table 9, and more detailed change trendline of each of two algorithms with the increasing size of data sets are given in Fig. 4.

In each sub-figures (a)–(f) of Fig. 3, the x-coordinate pertains to the size of the data set (the 10 data sets starting from the smallest one), while the y-coordinate presents the computational time. We use the star lines to denote the computational time of static algorithm (Algorithm 1) on different sizes of data sets, and the plus lines denote the computational time of incremental algorithm (Algorithm 3) on different sizes of data sets when inserting the objects into the universe. Obviously, according to Table 8 and Fig. 3, we can find that the computational time of the both algorithms usually

| Insert rate (%) | Mushroom | CVR | Audiology | Dermatology | AD1 | AD2 |
|----------------|----------|-----|-----------|-------------|-----|-----|
| 10             | 6.069    | 0.553 | 0.067     | 0.021       | 0.038 | 0.015 |
| 20             | 7.728    | 1.431 | 0.090     | 0.036       | 0.044 | 0.024 |
| 30             | 9.907    | 2.605 | 0.094     | 0.045       | 0.048 | 0.035 |
| 40             | 12.510   | 3.975 | 0.105     | 0.059       | 0.055 | 0.041 |
| 50             | 15.942   | 5.809 | 0.122     | 0.064       | 0.064 | 0.050 |
| 60             | 20.223   | 8.150 | 0.146     | 0.073       | 0.073 | 0.060 |
| 70             | 25.321   | 11.082| 0.152     | 0.087       | 0.087 | 0.075 |
| 80             | 26.332   | 14.240| 0.188     | 0.093       | 0.093 | 0.086 |
| 90             | 31.280   | 18.128| 0.225     | 0.106       | 0.106 | 0.099 |
| 100            | 36.221   | 22.036| 0.248     | 0.120       | 0.120 | 0.117 |

Table 9
A comparison of static and incremental algorithms versus different updating rates when inserting the objects.
increases with the increasing size of data sets. However, the incremental algorithm is much faster than the static algorithm for computing the approximations when inserting the objects into the universe. Furthermore, the differences between static and incremental algorithms are getting larger when increasing the data size.

In each sub-figures (a)–(f) of Fig. 4, the x-coordinate pertains to the ratio of the number of the inserted objects and original data, while the y-coordinate concerns the computational time. According to the experimental results as shown in Table 9 and Fig. 4, we find that the computational time of both static (Algorithm 1) and incremental (Algorithm 3) algorithms are increasing monotonically along with the increasing of insert ratios. It is easy to get the incremental algorithm is always faster than the static algorithm when the inserting ratio increases from 10% to 100% according to Fig. 4(a)–(e). In Fig. 4(f), we find the incremental algorithm is much faster than the static algorithm when the inserting ratio is less than 85%, but slower than the static algorithm when the inserting ratio is more than 85%.

6. Conclusions

The incremental technique is an effective way to maintain knowledge in the dynamic environment. In this paper, we proposed incremental methods for updating approximations in SOIS when the information system is updated by inserting or deleting objects. Through discussing the principles of updating approximations by deleting objects from the information system and inserting objects into the information system, respectively, we proposed the incremental algorithms for updating approximations based on SOIS in terms of inserting or deleting an object. Experimental studies pertaining to four UCI data sets and two artificial data sets showed that the incremental algorithms can improve the computational efficiency for updating approximations when the object set in the information system varies over time. In real-world applications, an information system may be updated by inserting and deleting some objects at the same time. In our further work, we will focus on improving the incremental algorithm for updating knowledge by deleting and deleting some objects simultaneously. Furthermore, as an information system consists of the objects, the attributes, and the domain of attributes values, all of the elements in the system will change as time goes by under the dynamic environment. In the future, the variation of attributes and the domain of attributes values in SOIS will also be taken into consideration in terms of incremental updating knowledge.

Acknowledgements

This work is supported by the National Science Foundation of China (Nos. 61175047, 61100117 and 71201133) and NSAF (No. U1230117), the Youth Social Science Foundation of the Chinese Education Commission (11YJC630127) and the Fundamental Research Funds for the Central Universities (SWJTU11ZT08, SWJTU12CX091, SWJTU12CX117).

References

[1] L.A. Zadeh, Towards a theory of fuzzy informarion granulation and its centrality in human reasoning and fuzzy logic, Fuzzy Sets and Systems 90 (2) (1997) 111–127.
[2] L.A. Zadeh, Fuzzy logic = computing with words, IEEE Transaction On Fuzzy Systems 4 (1) (1996) 103–111.
[3] T.Y. Lin, Granular computing on binary relation I: data mining and neighborhood systems, in: A. Skowron, L. Polkowski (Eds.), Rough Sets in Knowledge Discovery, Physica-Verlag, 1998, pp. 107–121.
[4] A. Skowron, J. Stepaniuk, Information granules: towards foundations of granular computing, International Journal of Intelligent Systems 16 (2001) 57–85.
[5] A. Bargiela, W. Pedrycz, Granular Computing: An Introduction, Kluwer Academic Publishers, Boston, 2002.
[6] Y.Y. Yao, Relational interpretations of neighborhood operators and rough set approximation operators, Information Science 101 (1998) 239–259.
[7] V.S. Ananthanarayana, M.M. Narasimha, D.K. Subramanian, Tree structure for efficient data mining using rough sets, Pattern Recognition Letter 24 (2003) 857–864.
[8] Y.Y. Yao, N. Zhong, Potential applications of granular computing in knowledge discovery and data mining, in: Proceedings of World Multiconference on Systemics, Cybernetics and Informatics, 1999, pp. 573–580.
[9] Y.Y. Yao, Perspectives of granular computing in: Proceedings of 2005 IEEE International Conference on Granular Computing, vol. I, 2005, pp. 85–90.
[10] Z. Pawlak, Rough sets, International Journal of Computer and Information Sciences 11 (1982) 341–356.
[11] W.Y. Ma, Granular computing: basic issues and possible solutions, in: Proceedings of 5th Joint Conference on Information Sciences, 2000, pp. 186–189.
[12] W. Pedrycz, R. Weber, Special issue on soft computing for dynamic data mining, Applied Soft Computing 8 (4) (2008) 1281–1282.
[13] H.S. Nguyen, Knowledge discovery by relation approximation: a rough set approach, Lecture Notes in Computer Science 4062 (2006) 103–106.
[14] A. Cannavada, S.K. Pal, A novel fuzzy rough granular neural network for classification, International Journal of Computational Intelligence Systems 4 (5) (2011) 1042–1051.
[15] G.Q. Zhang, Z. Zheng, J. Lu, Q. He, An algorithm for solving rule sets-based bievel decision problems, Computational Intelligence 27 (2) (2011) 235–259.
[16] H. Li, X. Zhou, Risk decision making based on decision-theoretic rough sets: a three-way view decision model, International Journal of Computational Intelligence Systems 4 (1) (2011) 1–11.
[17] J. Komorowski, Z. Pawlak, L. Polkowski, A rough set perspective on data and knowledge, in: S.K. Pal, A. Skowron (Eds.), Rough Fuzzy Hybridization, Springer-Verlag, 1999, pp. 107–121.
[18] N. Shan, W. Ziarko, Data-based acquisition and incremental modification of classification rules, Computational Intelligence 11 (2) (1995) 357–370.
[19] R. Michalski, Knowledge repair mechanisms: evolution vs. revolution, in: Proceedings of the Third International Workshop on Machine Learning, Skytop, PA, 1985, pp. 116–119.
[20] P. Dey, S. Dey, S. Datta, J. Sil, Dynamic discruduction using rough sets, Applied Soft Computing 11 (5) (2011) 3887–3897.
[21] F. Wang, J.Y. Liang, Y.H. Qian, Attribute reduction: a dimension incremental strategy, Knowledge-Based Systems 39 (2013) 95–108.
[22] Z. Wang, J.Y. Liang, C.Y. Dang, Attribute reduction for dynamic data sets, Applied Soft Computing 13 (1) (2013) 676–689.
[23] Y.N. Fan, T.L. Tseng, C.C. Chern, C.C. Huang, Rule induction based on an incremental rough set, Expert Systems with Applications 36 (9) (2009) 11439–11450.
[24] C.C. Huang, T.L. Tseng, Y.N. Fan, C.H. Hsu, Alternative rule induction methods based on incremental object using rough set theory, Applied Soft Computing 13 (1) (2013) 372–389.
[25] Z. Zheng, G.Y. Wang, RRBA: a rough set and rule tree based incremental knowledge acquisition algorithm, Fundamenta Informaticae 59 (2004) 299–313.
[26] D. Liu, T.R. Li, D. Ruan, W.L. Zou, An incremental approach for inducing dynamic knowledge information systems, Fundamenta Informaticae 94 (2009) 245–260.
[27] D. Liu, T.R. Li, D. Ruan, J.B. Zhang, Incremental learning optimization on knowledge discovery in dynamic business intelligent systems, Journal of Global Optimization 51 (2011) 325–344.
[28] F. Hu, G.Y. Wang, H. Huang, Y. Wu, Incremental attribute reduction based on elementary sets, in: Proceedings of 10th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, Lecture Notes in Computer Science, Regina, Canada, 2005, pp. 185–193.
[29] M. Yang, An incremental updating algorithm of the computation of a core based on the improved discernibility matrix, Chinese Journal of Computers 29 (5) (2006) 407–413.
[30] J. Komorowski, Z. Pawlak, L. Polkowski, A. Skowron, Rough sets: tutorial, in: S.K. Pal, A. Skowron (Eds.), Rough Fuzzy Hybridization, A New Trend in Decision Making, Springer, 1999, pp. 3–98.
[31] Z. Pawlak, A. Skowron, Rough sets: some extensions, Information Sciences 177 (1) (2007) 28–40.
[32] D. Dmtsch, G. Edgida, Roughian: rough information analysis, International Journal of Intelligent Systems 16 (2001) 121–147.
[33] J.B. Zhang, T.R. Li, D. Ruan, D. Liu, Neighborhood rough sets for dynamic data mining, International Journal of Intelligent Systems 27 (2012) 317–342.
[34] S.Y. Li, T.R. Li, D. Liu, Incremental updating approximations in dominance-based rough sets approach under the variation of the attribute set, Knowledge-Based Systems 40 (2013) 17–26.
[35] H.M. Chen, T.R. Li, D. Ruan, J.H. Lin, C.X. Hu, A rough-set based incremental approach for updating approximations under dynamic maintenance environments, IEEE Transactions on Knowledge and Data Engineering 6 (1) (2010) 7–24.
[36] A. An, N. Shan, C. Chan, N. Cercone, W. Ziarko, Discovering rules for water demand prediction: an enhanced rough-set approach, Engineering Application and Artificial Intelligence 9 (6) (1996) 645–653.
