DYNAMIC REDUCTS IN OBJECT ORIENTED INFORMATION SYSTEM USING ROUGH SET THEORY

N. Ravi Shankar¹, T. Srikanth², B. Ravi Kumar ²
¹Department of Applied Mathematics, GIS, GITAM University, Visakhapatnam, India
²Department of Computer Science, GIS, GITAM University, Visakhapatnam, India

Abstract

We introduce dynamic reducts in object oriented information system using rough set theory. To develop dynamic reducts in object oriented information system using rough set theory, we propose object-oriented information system into rough set models. To accomplish this, we construct the class, object, and name structures using rough set theory. Next, combining class, name and object structures, we propose object-oriented information systems. Moreover, we introduce indiscernibility relations on the set of objects, lower and upper approximations, and object-oriented rough sets in the object-oriented information systems and we introduce the concept of object oriented reducts, object oriented reducts using indiscenibility matrix.

Keywords: rough set, object-orientation, discernibility matrix, reducts

1. Introduction:

Rough set theory ([1,2,3]), proposed by Z. Pawlak in 1982, offered an effective mathematical method to deal with uncertainty knowledge. Recently, rough set theory and its application have been developed rapidly, which are mainly concentrated on the generalization of rough set model, the research on uncertainty theory in rough set, rough set operations and their connections with other uncertainty operations, rough set and its contacts with other mathematical theories and so on.

Rough set theory [1, 2] provides a theoretical foundation of approximation of objects. Information systems represent characteristics of objects by attributes and its values, and for any given concepts, that is, any subsets of objects, lower and upper approximations by indiscernibility relations illustrate set-theoretic approximations of concepts. However, “traditional” rough set theory has the following two constraints:

1. All objects have the same attributes, thus it is difficult to illustrate different kinds of objects in one information system simultaneously.

2. There is no structural dependence between objects, for example, concrete / abstract relationship, and part/whole relationship. In this paper, to weaken these constraints, and explicitly treat different kinds of objects and structural hierarchy among objects when we consider approximation by rough set theory, we introduce the object-oriented paradigm[4] to the rough set theory, and propose object-oriented rough set models.

Reducts generated from information systems are sensitive to changes in the system. This can be seen by removing a randomly chosen set of objects from the original object set. Those reducts frequently occurring in random subtables can be considered to be stable, it is these reducts that are encompassed by dynamic reducts[11].

The rest of this paper is organized as follows. In Section 2, we briefly review rough set theory and object oriented paradigm. In Section 3, we constructed object oriented rough set models with the help of object-oriented information systems. In Section 4, we define indiscernibility relations of objects that reflect hierarchical structures between objects. Moreover, we propose lower and upper approximations, and object-oriented rough sets based on object-oriented information systems. In section 5, we developed object oriented reducts. In section 6, we developed algorithm for object oriented dynamic reducts and section 7 summarizes contributions of this paper.
An information system is a pair \( S = (U,A) \) where \( U \) and \( A \) are finite and nonempty sets. \( U \) is called the universe, and each element \( x \in U \) is called an object, respectively. On the other hand, each element \( a \in A \) is called an attribute, which is identified with a function \( a : U \rightarrow V_a \) that assigns a value to each object \( x \in U \), where \( V_a \) is the set of values of the function \( a \).

For any subset \( B \subseteq A \) of attributes, we construct an indiscernibility relation \( R_B \) on \( U \) as follows:

\[
xR_B y \iff a(x) = a(y), \quad \forall a \in B
\]

where \( a(x) \) means the value of the object \( x \in U \) at the attribute \( a \). \( x R_B y \) means that we cannot discern \( x \) and \( y \) by any combination of attributes in \( B \). It is clear that the indiscernibility relation \( R_B \) is an equivalence relation. We denote the equivalence class by \( [x]_{R_B} \) that contains \( x \) as \( B R_x \). The class of all equivalence classes by \( R_B \) provides a partition \( U/R_B \) of \( U \).

For a given information system \( S=(U,A) \), a given subset \( B \subseteq A \) of attributes, and any subset \( X \subseteq U \), we construct a lower approximation \( \underline{R_B}(X) \) and an upper approximation \( \overline{R_B}(X) \) of \( X \) as follows, respectively:

\[
\underline{R_B}(X) = \{ x \in U / [x]_B \subseteq X \} \quad \ldots \quad (2)
\]

\[
\overline{R_B}(X) = \{ x \in U / [x]_B \cap X \neq \phi \} \quad \ldots \quad (3)
\]

The lower approximation of \( X \) is the set of objects \( x \) that the equivalence class \( [x]_{R_B} \) of \( x \) is included to \( X \), and the upper approximation of \( X \) is the set of objects \( x \) that \( [x]_{R_B} \) has a non-empty intersection with \( X \). Note that we have the following set-inclusion relation: \( \underline{R_B}(X) \subseteq X \subseteq \overline{R_B}(X) \).

A rough set of \( X \) is a pair \( R(X) = (\underline{R_B}(X), \overline{R_B}(X)) \) of the lower approximation and the upper approximation of \( X \). The rough set \( R(X) \) provides an approximation of the set \( X \) in the information system \( S \) based on attributes in \( B \).

If we have \( \underline{R_B}(X) = X = \overline{R_B}(X) \), \( X \) is called \( R_B \)-definable. On the other hand, if we have \( \underline{R_B}(X) \subseteq X \subseteq \overline{R_B}(X) \), \( X \) is called \( R_B \)-rough.

Quality of approximation of \( X \) by the rough set \( R_B(X) \) is numerically evaluated as follows:

\[
\frac{| \underline{R_B}(X) |}{| \overline{R_B}(X) |} \quad \ldots \quad (4)
\]

where, for any set \( S \), \( |S| \) means the cardinality of \( S \). It is clear that the quality of approximation is equal to 1 if and only if \( X \) is \( R_B \)-definable.

Let \( P \) and \( Q \) be equivalence relations over \( U \), then the positive, negative and boundary regions are defined as:

\[
POS_P(Q) = \bigcup_{x \in U/Q} \sim_P(X) \quad \ldots \quad (5)
\]

\[
NEG_P(Q) = U \setminus \bigcup_{x \in U/Q} \sim_P(X) \quad \ldots \quad (6)
\]

\[
BND_P(Q) = \bigcup_{x \in U/Q} \sim_P(X) \setminus \bigcup_{x \in U/Q} \sim_P(X) \quad \ldots \quad (7)
\]

The positive region comprises all objects of \( U \) that can be classified to classes of \( U/Q \) using the information contained within attributes \( P \). The boundary region \( BND_P(Q) \), is the set of objects that can possibly, but not certainly, be classified in this way. The negative region, \( NEG_P(Q) \), is the set of objects that can not be classified to classes of \( U/Q \).
Let $S = (U, C, D)$ be an information system where $C$ is the set of condition attributes and $D$ is the set of decision attributes. The set of attributes $R \subseteq C$ is called a reduct of $C$, if $S' = (U, R, D)$ is independent and $\text{POS}_C(D) = \text{POS}_{S'}(D)$. The set of all the condition attributes indispensable in $T$ is denoted by $\text{CORE}(C) = \cap \text{RED}(C)$ where $\text{RED}(C)$ is the set of all reducts of $C$.

### 2.2. Object-Oriented Paradigm

In this subsection, we review object-oriented paradigm briefly. Contents of this subsection are based on [4]. Object-orientation is a framework for software engineering based on objects and classes, which provides modeling and implementation methodologies of systems. According to [4], object-oriented paradigm is summarized as the following characteristics by Alan Kay [5]:

The object oriented paradigm draws heavily on the general systems theory as a conceptual background. A system can be viewed as a collection of entities that interact together to accomplish certain object tools. Entities may represent physical objects such as equipment and people and abstract concepts such as data files and functions. In object oriented analysis, the entities are called objects.

The object oriented paradigm places greater emphasis on the objects that encapsulate data and procedures. Computation is performed by objects communicating with each other, requesting that other objects perform actions. Objects communicate by sending and receiving messages. A message is a request for action bundled with whatever arguments may be necessary to complete the task. Each object has its own memory, which consists of other objects. Every object is an instance of a class. A class simply represents a grouping of similar objects, such as integers, or lists. The class is the repository for behavior associated with an object. That is, all objects that are instances of the same class can perform the same actions. Classes are organized into a singly-rooted tree structure, called the inheritance hierarchy. Memory and behavior associated with instances of a class are automatically available to any class.

### 3. Object-Oriented Rough Set Models

In this section, we propose object-oriented information systems that illustrate hierarchical structures of object oriented concepts. First, we propose class structures that represent abstract data forms and hierarchical structures between classes. Next, we define object structures that illustrate many kinds of objects and actual dependence among objects by has-a relationship and offers-a relationship.

Moreover, we define name structures that introduce strict constraint to guarantee consistency of structures. Name structures provide concrete design of objects, and connect the class structure and the object structure consistently. Finally, combining these structures, we provide object oriented information systems as generalization of “traditional “information systems of rough set theory.

#### 3.1. Class

**Definition 1:** A class structure $C \square$ is the following triple:

$$( C, R_c, S_c)$$

Where $C$ is a finite non-empty set, $R_c$ is an acyclic binary relation on $C$ that is $R_c$ satisfies the following property:

$$\not\exists c_1, c_2, \ldots, c_n \in C \text{ such that } c_1 R_c c_2, c_2 R_c c_3, \ldots, c_{n-1} R_c c_n, c_n R_c c_1$$

and $S_c$ is a reflexive, transitive, and asymmetric binary relation on $C$. Moreover, $C_R$ and $C_S$ satisfying the following property:

$$\forall c_i, c_j, c_k \in C, c_i S_c c_j, c_j R_c c_k \Rightarrow c_j R_c c_k \ldots$$

Each $c \in C$ is called a class that represents an abstract data form. Note that each class corresponds to a sort in many-sorted logic [6] and order-sorted logic [7].

Two relations $R_c$ and $S_c$ illustrate hierarchical structures among classes. The relation $R_c$ is called a offers-a relation, which illustrates part / whole relationship between classes. $c_i R_c c_j$ means “$c_i$ offers a $c_j$”. The relation $S_c$ is called a has -a relation, and $c_i S_c c_j$ means that “$c_i$ has a $c_j$” or $c_i$ is a part of $c_j$. 

ISSN: 0975-5462

124
Because \( C \) is a finite non-empty set, and \( R_c \) is acyclic, there is at least one class \( c \) such that \( c \) has no other class \( c' \), that is, \( c R_c c' \) for any \( c' \in C \). We call such class \( c \) an attribute, and denote the set of attributes by \( AT \). Formally, \( AT \) is defined as follows:

\[
AT = \{ c \in C / c R_c c', \forall c' \in C \} \quad \cdots \quad (8)
\]

Example 1: Let \( C = (C, R_C, S_C) \) be class structure with

\[
C = \{ \text{University,College,Department, Faculty, Student, Course, Ncollege, Ndept, Nstudents} \}
\]

and have the following relations.

- has-a relation: University \( S_c \) College,
- College \( S_c \) Department,
- University \( S_c \) Department,

………………

Offers-a relation: College \( R_c \) Department

Department \( R_c \) Courses.

Suppose moreover that \( Ncollege, Ndept \) and \( Nstudents \) are attributes.

By the Property(6), these relations illustrate connection between classes, for example, “University has a College” and “College offers Department” imply “University offers department”

(or)

“University has a Department” and “Department offers Course” imply “University offers Course”

3.2. Object

We define an object structure that illustrates hierarchical structures among objects.

Definition 2: An object structure \( O \) is the following triple:

\[
(O, R_o, S_o) \quad \cdots \quad (9)
\]

where \( O \) is a finite non-empty set, \( R_o \) is an acyclic binary relation on \( O \), and \( S_o \) is a reflexive, transitive, and asymmetric binary relation on \( O \). Moreover, similar to the definition of class, \( R_c \) and \( S_c \) satisfy the following property:

\[
\forall o_i, o_j, o_k \in O, o_i S_o o_j, o_j R_o o_k \Rightarrow o_i R_o o_k \quad \cdots \quad (10)
\]

We intend that every object \( o \in O \) is an instance of some class \( c \in C \). To represent this intention, we define a class identifier function \( id_C \) as follows.

Definition 3: Let \( C = (C, R_c, S_c) \) be the class structure and \( O = (O, R_o, S_o) \) be the object structure. A function \( id_C : O \to C \) is called class identifier iff \( id_c \) a p-morphism between \( O \) and \( C \) (cf.[8],p142) that is, the function \( id_C \) satisfies the following conditions:

1. \( \forall o_i, o_j \in O, o_i R_o o_j \Rightarrow id_C(o_i) R_c id_C(o_j) \quad \cdots \quad (11) \)
2. \( \forall o_i \in O, \forall c, c_j \in C, id_C(o_i) R_c c_j \Rightarrow \exists o_j \in O \text{ s.t. } o_i R_o o_j \text{ and } id_C(o_j) = c_j \quad \cdots \quad (12) \)

and the same conditions are also satisfied for \( S_o \) and \( S_c \). \( id_C(o) = c \) means that the object \( o \) is an instance of the class \( c \).

For any object \( x \), if \( id_C(x) = a \) and \( a \in AT \), we call such object \( x \) a value object of the attribute \( a \). The value object \( x \) is an instance of the attribute \( a \) represents a “value” of the attribute. Thus, if \( y \) is another value object of \( a \), it is natural to enable us to compare the “value” of \( x \) and \( y \). We introduce the concept of “value” of value objects.

Definition 4: For any object \( x \), if \( id_C(x) = a \) and \( a \in AT \), we call such object \( x \) a value object of the attribute \( a \). We denote the “value” of the value object \( x \) by \( Val(x) \).
3.3. Name

We introduce a name structure to provide concrete design of objects, and connect the class structure and the object structure consistently. The class structure provides abstract data forms of objects, however, does not provide constraints about the number of parts and their identification. Suppose we have $c_i \ R_c \ c_j$ and we intend that any instance $o_i$ of the class $c_i$ has $m$ objects of $c_j$ as parts of $o_i$ and each object of $c_j$ should be strictly identified. Direct connection between objects and classes by the class identifier $id_c$.

**Definition 5 :** Let $C= ( C, R_C, S_C)$ be the class structure. A name structure $N$ for $C$ is the following triple:

$$\langle N, R_N, S_N \rangle \quad \ldots \quad (13)$$

where $N$ is a finite non-empty set such that $|C| \leq |N|$, $R_N$ is an acyclic binary relation on $N$, and $S_N$ is a reflexive, transitive, and asymmetric binary relation on $N$. Moreover, similar to the definition of class, $R_N$ and $S_N$ satisfy the following property:

$$\forall n_i, n_j, n_k \in N \ , \ n_i \ S_N n_j \ R_N n_k \Rightarrow n_i \ R_N n_k \quad \ldots \quad (14)$$

We call each $n \in N$ a name.

We intend that a naming function $f_n : N \rightarrow C$ provides names to each class. To introduce the naming function precisely, we define the following notations.

**Definition 6:** Let $C = ( C, R_C, S_C)$ be the class structure, $N = (N,R_N,S_N)$ be the name structure, and $f : N \rightarrow C$ be a function. For any name $n \in N$, we denote the set of names that $n$ has by:

$$H_N(n) = \{ n_j \in N / n \ R_N n_j \} \quad \ldots \quad (15)$$

Moreover, using the function $f$, we denote the set of names of a class $c \in C$ that $n$ has by

$$H_n^f (c/n) = \{ n_j \in N / n \ R_N n_j , f(n_j) = c \} \quad \ldots \quad (16)$$

**Definition 7:** Let $C = ( C, R_C, S_C)$ be the class structure, $N = (N,R_N,S_N)$ be the name structure. A function $f_n : N \rightarrow C$ is called a naming function if and only if $f_n$ is a surjective p-morphism between $N$ and $C$ and satisfies the following name preservation constraint:

For any $n_i, n_j \in N$, if $f_n(n_i) = f_n(n_j)$ then

$$H_n^f (c/n_i) = H_n^f (c/n_j) \quad \ldots \quad (17)$$

is satisfied for all $c \in C$.

Example 2: This example is continuation of Example1. Let $C = ( C, R_C, S_C)$ be the class structure in Example1, $N = (N,R_N,S_N)$ is a name structure with $N=\{university,college,department,faculty,student,college2,course,ncollege,ndepartment,nstudents\}$ and the following relationships:

Has- a relation: $S_N$ college,

$$\begin{align*}
\text{college} & \quad S_N \text{ department}, \\
\text{university} & \quad S_N \text{ Department},
\end{align*}$$

Offers-relation :

$$\begin{align*}
\text{College} & \quad R_N \text{ Department} \\
\text{Department} & \quad R_N \text{ Courses}.
\end{align*}$$

Moreover, suppose we have a naming function $f_n : N \rightarrow C$ such that

$$f_n(\text{university}) = \text{University},$$

$$f_n(\text{college}) = f_n(\text{college2}) = \text{College},$$

$$f_n(\text{department}) = \text{Department},$$

$$f_n(\text{course}) = \text{Course},$$

$$f_n(\text{faculty}) = \text{Faculty},$$

$$f_n(\text{student}) = \text{Student},$$

$$f_n(\text{ncollege}) = \text{NC},$$

$$f_n(\text{ncollege2}) = \text{NC2}.$$
\(f_n(\text{department}) = \text{Department},\)
\(f_n(\text{faculty}) = \text{Faculty},\)
\(f_n(\text{student}) = \text{Student},\)
\(f_n(\text{course}) = \text{Course},\)
\(f_n(\text{ncollege}) = \text{Ncollege},\)
\(f_n(\text{ndepartment}) = \text{Ndepartment},\)
\(f_n(\text{nstudent}) = \text{Nstudent}.\)

Note that we have \(H_N(\text{College/university}) = \{\text{college, college2}\},\) and \(H_N(\text{Ndepartment/college}) = H_N(\text{Ndepartment/college2}) = \{\text{ndepartment}\}.\)

Here, to illustrate connection between the classes and names, we use class diagrams of UML[9] authorized by OMG[10] as in Fig 1. For example, the class diagram “University” illustrates that University class has two objects of the College class, called “college” and “college2”, respectively, one object “student” of the Student class, and one object “faculty” of the Faculty class.

\[
\begin{array}{|c|c|}
\hline
\text{University} & \text{College} \\
\text{College} & \text{college} \\
\text{College} & \text{college2} \\
\text{Student} & \text{student} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{Department} & \text{Faculty} \\
\text{Course} & \text{course} \\
\text{Faculty} & \text{faculty} \\
\text{Student} & \text{student} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{Ndepartment} & \text{ndepartment} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{Course} & \text{Student} \\
\text{Ndepartment} & \text{ndepartment} \\
\text{Nstudent} & \text{nstudent} \\
\hline
\end{array}
\]

\[\text{Fig 1. Class diagrams in example 2}\]

**Definition 8**: Let \(O = (O, R_O, S_O)\) be the object structure and \(N = (N, R_N, S_N)\) be the name structure. A function \(a_n: O \rightarrow N\) is called a name assignment if and only if \(a_n\) is a \(p\)-morphism between \(O\) and \(N\) satisfies the following uniqueness condition:

For any \(x \in O\), if \(H_O(x) \neq \emptyset\), the restriction of \(a_n\) into \(H_O(x):\)

\[a_n/H_O(x) = H_O(x) \rightarrow N\]

is injective, where \(H_O(x) = \{ y \in O / x R_O y \}\) is the set of objects that \(x\) has \(a_n(x) = n\) means that the name of the object \(x\) is \(n\).

**Definition 9**: Let \(C = (C, R_C, S_C)\) be the class structure, \(N = (N, R_N, S_N)\) be the name structure, \(O = (O, R_O, S_O)\) be the object structure. Moreover, let \(id_C: O \rightarrow C\) be the class identifier. We say that \(C, N\) and \(O\) are well defined if and only if there exists a naming function \(f_n: N \rightarrow C\) and a name assignment \(a_n: O \rightarrow N\) such that

\[\text{id}_C = f_n \circ a_n\]

that is, \(\text{id}_C(x) = f_n(a_n(x))\) for all \(x \in O\).

**Definition 10**: Let \(C, N\) and \(O\) be well defined structures. Suppose we have \(o_1, o_2, \ldots, o_k \in O, n_1, n_2, \ldots, n_k \in N, c_1, c_2, \ldots, c_k \in C\) such that \(o_i R_O o_{i+1}\) for \(1 \leq i \leq k-1,\) and \(a_n(o_i) = n_i, f_n(n_i) = c_i\) for \(1 \leq i \leq k.\) We denote \(o_1, n_2, \ldots, n_k\) instead of \(o_i\) for \(2 \leq i \leq k\) by means of “the instance of \(c_i\) named \(n_i\) as a part of the instance of \(c_i\) as a part of \(o_i\)”.

**Example 3**: This example is continuous of Example 2. Let \(C = (C, R_C, S_C)\) and \(N = (N, R_N, S_N)\) are the same class structure and name structure in example 2, respectively. Moreover, let \(O = (O, R_O, S_O)\) be an object structure with the offers – a relationship illustrated in Fig 2 and the following has-a relationship.
Moreover, let \( a_n : O \rightarrow N \) be the following name assignment:

\[
\begin{align*}
\text{university1} & \rightarrow \text{college}, \\
\text{university2} & \rightarrow \text{college}, \\
\text{university3} & \rightarrow \text{college}, \\
\text{c1} & \rightarrow \text{college}, \\
\text{c2} & \rightarrow \text{college}, \\
\text{c3} & \rightarrow \text{college}, \\
\text{c4} & \rightarrow \text{college}, \\
\text{s1} & \rightarrow \text{student}, \\
\text{s2} & \rightarrow \text{student}, \\
\text{f1} & \rightarrow \text{faculty}, \\
\text{f2} & \rightarrow \text{faculty}, \\
\text{f3} & \rightarrow \text{faculty}, \\
\text{24} & \rightarrow \text{ncollege}, \\
\text{150} & \rightarrow \text{ndepartment}, \\
\text{1500} & \rightarrow \text{nstudent}.
\end{align*}
\]

We define the class identifier \( id_C : O \rightarrow C \) by Eq.(19) using \( a_n \) and \( f_n \) used in example 2. It is not hard to check that \( C, N \) and \( O \) are well defined.

**Object-Oriented Information System**

Using well defined class, name structure and object structures, we introduce an object oriented information system that corresponds to the information in “traditional” roughset theory.

Definition 11: Let \( C = (C, R_C, S_C) \) and \( N = (N, R_N, S_N) \), \( O = (O, R_O, S_O) \) be well defined class, name, object structures respectively. An object oriented information system \( \text{OOIS}(O, C, N) \) is the following structure:

\[
\text{OOIS}(O, C, N) = (O, C, N, o, id_c) \quad \text{… (20)}
\]

where \( id_c = f_o \circ a_n \)

The object oriented information system can be illustrate “traditional” information system as special case. In particular, for any information system \( \text{IS}(U, A) \), we can construct an object oriented information system \( \text{OOIS}(O_{\text{IS}}, C_{\text{IS}}, N_{\text{IS}}) \) that corresponds to \( \text{IS} \): First, using the information system \( \text{IS} = (U, A) \), we construct a name structure.

\[
N_{\text{IS}} = (N_{\text{IS}}, R_{N_{\text{IS}}}, S_{N_{\text{IS}}})
\]

as follows:

![Fig 2. Offers - a relation on objects in example 3](image-url)
\[ N_{IS} = A \cup \{ IS \} \]
\[ R_{N_{IS}} = \{(s,a) / a \in A\} \]
\[ S_{N_{IS}} = \{(n,n) / n \in N_{IS}\} \]

Where \( s \) is a symbol that does not appear in \( A \). We also construct an object structure
\[ O_{IS} = (O_{IS}, R_{O_{IS}}, S_{O_{IS}}) \]
as follows:
\[ O_{IS} = U \cup \{ v^a_x / \exists a, \exists x, v \in v^a_x, a(x) = v \} \]
\[ R_{O_{IS}} = \{(x, v^a_x) / x \in U\} \]
\[ S_{O_{IS}} = \{(o,o) / o \in O_{IS}\} \]

Where \( v^a_x \) is a new symbol that corresponds to the value of the object of the attribute \( a \) as a part of the object \( x \), and \( v ( v^a_x ) = v \).

We set a class structure
\[ C_{IS} = (C_{IS}, R_{C_{IS}}, S_{C_{IS}}) \]
as follows:
\[ C_{IS} = N_{IS}, R_{C_{IS}} = R_{N_{IS}}, S_{C_{IS}} = S_{N_{IS}} \]

Finally, we construct a name assignment
\[ (a_n)_{N_{IS}} : O_{IS} \rightarrow N_{IS} \]
by
\[ (a_n)_{N_{IS}}(o) = \begin{cases} 
  s \text{ if } o \in U \\
  a \text{ if } o \in v^a_x, \exists x \in U 
\end{cases} \]
The function \( (a_n)_{N_{IS}} \) becomes a name assignment: if \( o \in U \), then we have
\[ H_o(o) = \{ v^a_x / a \in A \}, \]
that is, the set of value objects about \( o \), and by the construction of value objects \( v^a_o \), each \( v^a_o \) and \( a = (a_n)_{N_{IS}} v^a_o \in N_{IS} \) corresponds one to one. Otherwise, we have \( o = v^a_x \), and therefore \( H_o(o) = \phi \). We define the naming function \( (f_n)_{C_{IS}} : N_{IS} \rightarrow C_{IS} \)
by \( (f_n)_{C_{IS}}(n) = n \in C_{IS} \) for all \( n \in N_{IS} \).

Using \( (a_n)_{N_{IS}} \), we get \( id_{C_{IS}} = (f_n)_{C_{IS}} o (a_n)_{N_{IS}} \).

\[ \text{OOIS (O_{IS}, C_{IS}, N_{IS}) satisfies the following property: } a(x) = v \Leftrightarrow \text{val} (x,a) = v, \forall x \in U, \forall a \in A. \]

4. Indiscernability Relations and object orient Rough Sets

4.1 Equivalence as instance

We provide indiscernability relation on \( O \) based on structural aspects of objects. At first, we introduce an equivalence relation that illustrates "equivalence as instances of a class". To evaluate equivalence of instances, we use names of parts of instances as follows.

**Definition 12**: Let \( \text{OOIS (O}_{C},N_{C}) \) be the object oriented information system. We define a binary relation \( \sim \) on \( O \) recursively as follows:
\[ x \sim y \Leftrightarrow x \text{ and } y \text{ satisfy the following two conditions:} \]
1. \( \text{id}_c(x) = \text{id}_c(y) \)
2. \[ \begin{cases} 
  x.n \sim y.n, \forall n \in H_n(a_n(x)) \text{ if } H_n(a_n(x)) \neq \phi, \\
  \text{val}(x) = \text{val}(y) 
\end{cases} \]
where \( H_n(a_n(x)) \) is the set of names that \( a_n(x) \) has, and defined by Eq.(15).

**Example 4**: This example is continuation of Example3. Let \( \text{OOIS (O}_{C},N_{C}) \) be the object-oriented information system based on the class structure \( C \), the object structure \( O \) and the name structure \( N \) used in example 3. We construct the equivalence relation \( \sim \) on \( O \) by Eq.(20). Equivalence classes by \( \sim \) are as follows:
\[ \text{[university1]}_\sim = \{ \text{university 1} \}, \]
\[ \text{[university2]}_\sim = \{ \text{university 2} \}, \]
[university3] = {university 3},
[s1] = {s1,s2,s4}, [s2] = {s2},
[d1] = {d1}, [d2] = {d1,d2},
[c1] = {c1,c2}, [c3] = {c3},
[16] = {university2.college2.dept}

Note that \( n(a(c1)) = n(a(c3)) = n(a(c4)) = college \), \( n(a(c4)) = college2 \) and \( f_{n(college)} = f_{n(college2)} = college \). By the name preservation constraint, we have \( H_N(Ndepartment/college) = H_N(Ndepartment/college2) = \{ndepartment\} \) as illustrated in example 2. Therefore, for example, we can evaluate the equivalence of \( c1 (= university.college) \) and \( c4(=university3.college2) \) by evaluating the equivalence of university1.college.department and university3.college2.department.

4.2. Object-Oriented Rough Sets

Definition 13: Let OOIS \((O,C,N)\) be the object oriented information system, and \( \sim \) be the equivalence relation defined by Eq. (20). For any non-empty subset \( B \subseteq N \), we define a binary equivalence relation \( \sim_B \) on \( O \) as follows:

\[
\sim_B(x) = \{ y \in O / \n(x) \sim_B x \subseteq y \} \quad \ldots (23)
\]

Moreover, the object –oriented rough set \( \sim_B(X) \) of \( X \) by \( \sim_B \) is the following pair:

\[
( \sim_B(X), \sim_B(X) ) \quad \ldots (25)
\]

- X is roughly \( \sim_B \)–definable, if and only if \( \sim_B(X) \neq \phi \) and \( \sim_B(X) \neq U \).
- X is internally \( \sim_B \)–undefinable, if and only if \( \sim_B(X) = \phi \) and \( \sim_B(X) \neq U \).
- X is externally \( \sim_B \)–undefinable, if and only if \( \sim_B(X) = \phi \) and \( \sim_B(X) = U \).
- X is totally \( \sim_B \)–undefinable, if and only if \( \sim_B(X) = \phi \) and \( \sim_B(X) = U \).

The quality of approximation of X by the rough set \( \sim_B \) is defined by \( \frac{|\sim_B(X)|}{|\sim_B(X)|} \) and some properties of approximations are as follows.

1. \( \sim_B(X) \subseteq X \subseteq \sim_B(X) \)
2. \( \sim_B(\phi) = \sim_B(\phi) = \phi, \sim_B(O) = \sim_B(O) = O \).
3. \( \sim_B(X \cup Y) = \sim_B(X) \cup \sim_B(Y) \)
4. \( \sim_B(X \cap Y) = \sim_B(X) \cap \sim_B(Y) \).
5. \( X \subseteq Y \) implies \( \sim_B(X) \subseteq \sim_B(Y) \) and \( \sim_B(X) \subseteq \sim_B(Y) \)
Definition 15: Let $O = (O, R_O, S_O)$ be the object structure and $D = \{d_1, d_2, \ldots, d_n\}$ be set of decision attribute values and $d \notin AT$ where $AT$ is set of all condition attributes. A function $g_n : O \rightarrow D$ is called decision function if and only if $g_n$ is a surjective p-morphism and satisfies the following constraint:

$$
\bigwedge c_i \Rightarrow d_i \quad \text{where} \quad c_i \in AT \quad \text{and} \quad d_i \in D \quad (i = 1, 2, \ldots |AT|) \quad \ldots \quad (26)
$$

Example 5: This example is continuation of Example 4. Suppose $B = \{\text{college}\}$, using the equivalence relation $\sim$ constructed in example 4, we construct the equivalence relation $\sim_B$ and equivalence classes by $\sim_B$ as follows:

- $[\text{university1}]_{\sim_B} = \{\text{university1, university3}\}$,
- $[\text{university2}]_{\sim_B} = \{\text{university2}\}$,
- $[\text{c1}]_{\sim_B} = O - \{\text{university1, university2, university3}\}$.

The equivalence classes $[\text{university1}]_{\sim_B}$ correspond to the set of objects with the “24 college” and $[\text{university2}]_{\sim_B}$ is the set of objects with “16 colleges”. On the other $[\text{c1}]_{\sim_B}$ represents the set of objects that has no instance of the College class. Note that university1 and university3 are in the same university class $[\text{university1}]_{\sim_B}$ even though $\text{id}_C(\text{university1}) \neq \text{id}_C(\text{university3})$.

Let $X = \{\text{university2, university3}\}$. The lower approximation $\sim_B(X)$ and upper approximation $\overline{\sim_B(X)}$ of $X$ by $\sim_B$ are constructed as follows, respectively.

- $\sim_B(X) = \{x \in O/ [x]_{\sim_B} \subseteq X\} = \{\text{university2}\}$.
- $\overline{\sim_B(X)} = \{x \in O/ [x]_{\sim_B} \cap X \neq \emptyset\} = \{\text{university1, university2, university3}\}$.

Thus object oriented rough set $\sim_B(X)$ is:

$\sim_B(X) = (\sim_B(X), \overline{\sim_B(X)}) = (\{\text{university2}\}, \{\text{university1, university2, university3}\})$

and the quality of approximation of $X = \{\text{university2, university3}\}$ by the roughest $\sim_B(X)$ is $\frac{1}{3}$.

5. Object oriented reducts

We provide definition of reducts in the object oriented rough set models and propose algorithms for calculating reducts.

Consider the attributes that preserve the indiscernibility relation and, consequently, set approximation. There are usually several subsets of attributes and those which are minimal are called reducts.

Positive, Negative and Boundary Regions:

Let $P$ and $Q$ be equivalence relations over $O$, then the positive, negative and boundary regions are defined as:

$$
POS_p(Q) = \bigcup_{x \in O / Q} \sim_p(X) \quad \ldots \quad (27)
$$

$$
NEG_p(Q) = O - \bigcup_{x \in O / Q} \sim_p(X) \quad \ldots \quad (28)
$$

$$
BND_p(Q) = \bigcup_{x \in O / Q} \overline{\sim_p(X)} - \bigcup_{x \in O / Q} \overline{\sim_p(X)} \quad \ldots \quad (29)
$$
The positive region comprises all objects of O that can be classified to classes of O/Q using the information contained within attributes P. The boundary region \( BND_p(Q) \), is the set of objects that can possibly, but not certainly, be classified in this way. The negative region, \( NEG_p(Q) \), is the set of objects that can not be classified to classes of O/Q.

**Dispensable & Indispensable attributes**

Let \( c \in AT \). Attribute \( c \) is dispensable in OOIS if \( POS_{AT}(D) = POS_{AT-\{c\}}(D) \), otherwise attribute \( c \) is indispensable in OOIS.

The AT-positive region of D:

\[
POS_{AT}(D) = \bigcup_{X \in O/D} \sim_{AT}(X) \quad \cdots \quad (30)
\]

**Independent Object oriented Information system**:

OOIS \( (O,C,N) \) is independent if all \( c \in AT \) are indispensable in OOIS.

**Reduct and Core**:

- The set of attributes \( R \subseteq AT \) is called a reduct of \( AT \), if OOIS'\( (O,R,N) \) is independent and \( POS_R(D) = POS_{AT}(D) \).
- The set of all the condition attributes indispensable in T is denoted by \( \text{CORE}(AT) = \wedge \text{RED}(AT) \) where \( \text{RED}(AT) \) is the set of all reducts of \( AT \).

**Feature Dependency and significance**:

An important issue in data analysis is discovering dependencies between attributes. Intuitively, a set of attributes \( Q \) depends totally on a set of attributes \( P \), denoted \( P \Rightarrow Q \), if all attribute values from \( Q \) are uniquely determined by values of attributes from \( P \). If there exists a functional dependency between values of \( Q \) and \( P \), then \( Q \) depends totally on \( P \). In object oriented rough set theory, dependency is defined in the following way:

For \( P,Q \subset AT \), it is said that \( Q \) depends on \( P \) in a degree \( k \) ( \( 0 \leq k \leq 1 \)), denoted \( P \Rightarrow^k Q \), if

\[
k = \gamma_p(Q) = \frac{|POS_p(Q)|}{|O|} \quad \cdots \quad (31)
\]

By calculating the change in dependency when a feature is removed from the set of considered possible features, an estimate of the significance of that feature can be obtained. The higher the change in dependency, the more significant the feature is. If the significance is 0, then the feature is dispensable. More formally, given \( P,Q \) and a feature \( x \in P \), the significance of feature \( x \) upon \( Q \) is defined by

\[
\sigma_p(Q,a) = \gamma_p(Q) - \gamma_{P-\{a\}}(Q). \quad \cdots \quad (32)
\]

**Reducts**

For many application problems, it is necessary to maintain a concise form of the object oriented information system. One way to complement this is to search for a minimal representation of the original data set. For this, the concept of a reduct is introduced and defined as a minimal subset \( R \) of the initial attribute set \( AT \) such that for a given set of attributes \( D \), \( \gamma_R(D) = \gamma_{AT}(D) \). From the literature, \( R \) is a minimal subset if \( \gamma_{R-\{a\}}(D) \neq \gamma_R(D) \) for all \( a \in R \). This means that no attributes can be removed from the subset affecting the dependency degree. Hence, a minimal subset by this definition may not be the global minimum(a reduct of smallest cardinality). A given dataset may have many reduct sets, and the collection of all reducts is denoted by

\[
R_{all} = \{X / X \subseteq AT, \gamma_X(D) = \gamma_{AT}(D) ; \gamma_{X-\{a\}}(D) \neq \gamma_X(D), \forall a \in X \} \quad \cdots \quad (33)
\]

The intersection of all the sets in \( R_{all} \) is called the core, the elements of which are those attributes that cannot be eliminated without introducing more contradictions to the representation of the dataset.

For many tasks, a reduct of minimal cardinality is ideally searched for. That is, an attempt is to be made to locate a single element of the reduct set \( R_{min} \subseteq R_{all} : \)

\[
R_{min} = \{X / X \in R_{all}, \forall Y \in R_{all}, |X| \leq |Y| \} \quad \cdots \quad (34)
\]

**Discernibility Matrix**:

Many applications of rough sets make use of discernibility matrix for finding rules or reducts. A discernibility matrix of a decision table \( (O, AT \cup D) \) is a symmetric \( |O| \times |O| \) matrix with entries defined by
\[ c_{ij} = \{a \in AT / a(x_i) \neq a(x_j) \} \ i, j = 1, ..., |O| \] ... (35)

Each \( c_{ij} \) contains those attributes that differ objects \( i \) and \( j \).

A discernibility function \( f_{OOS} \) for an object oriented information system is a boolean function of \( m \) Boolean variables \( a_1, ..., a_m \) corresponding to the membership of attributes \( a_1, a_2, ..., a_m \) to a given entry of the discernibility matrix defined as below:

\[ f_{OOS}(a_1, ..., a_m) = \wedge \{ c_{ij} / 1 \leq j \leq |O|, c_{ij} \neq \emptyset \} \] ... (36)

where \( c_{ij} = \{ a^* / a \in c_{ij} \} \). By finding the set of all prime implicants of the discernibility function, all the minimal reducts of a system may be determined.

**Example 6:** Consider the following table which shows an example of object oriented data set.

<table>
<thead>
<tr>
<th>( x \in O )</th>
<th>college</th>
<th>department</th>
<th>student</th>
<th>( \Rightarrow ) Granting Research Fund (GRF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>university1</td>
<td>5</td>
<td>10</td>
<td>100</td>
<td>Yes</td>
</tr>
<tr>
<td>university2</td>
<td>6</td>
<td>12</td>
<td>110</td>
<td>No</td>
</tr>
<tr>
<td>university3</td>
<td>8</td>
<td>14</td>
<td>100</td>
<td>Yes</td>
</tr>
<tr>
<td>university4</td>
<td>5</td>
<td>10</td>
<td>100</td>
<td>No</td>
</tr>
<tr>
<td>university5</td>
<td>6</td>
<td>12</td>
<td>110</td>
<td>Yes</td>
</tr>
<tr>
<td>university6</td>
<td>7</td>
<td>15</td>
<td>100</td>
<td>Yes</td>
</tr>
<tr>
<td>university7</td>
<td>8</td>
<td>14</td>
<td>100</td>
<td>No</td>
</tr>
<tr>
<td>university8</td>
<td>9</td>
<td>15</td>
<td>140</td>
<td>No</td>
</tr>
</tbody>
</table>

The indiscernibility classes defined by \( R = \{N\text{college}, N\text{department}, N\text{student} \} \) are \( \{ \text{university1, university4}, \{ \text{university2, university5}, \{ \text{university3, university7}, \{ \text{university6}, \{ \text{university8} \} \} \} \} \). Let \( G = \{ x / GRF(x) = Yes \} \).

\( G_1 = \{ o / G(o) = Yes \} \)

\[ = \{ \text{university1, university3, university5, university6} \} \]

\( RG_1 = \{ \text{university6} \} \)

\( RG_1 = \{ \text{university1, university4, university2, university5, university3, university7, university6} \} \)

\( G_2 = \{ o / G(o) = Yes \} \)

\[ = \{ \text{university2, university4, university7, university8} \} \]

\( RG_2 = \{ \text{university8} \} \)

\( RG_2 = \{ \text{university1, university4, university2, university5, university3, university7, university8} \} \)
Let $P = \{N_{college}, N_{department}, N_{student}\}$, $Q = \{GRF\}$.

$$POS_p(Q) = \bigcup_{x \in U/Q} P(x)$$

$$= RG1 \cup RG2$$

$$= \{\text{university6, university8}\}$$

$$NEG_p(Q) = O - \bigcup_{x \in U/Q} P(x)$$

$$= O - (RG1 \cup RG2)$$

$$= O \cup = \phi$$

$$BND_p(Q) = \bigcup_{x \in U/Q} P(x) - \bigcup_{x \in U/Q} P(x)$$

$$= \{\text{university1, university3, university4, university5, university7, university8}\}$$

REDUCT1 = $\{N_{department}, N_{college}\}$

REDUCT2 = $\{N_{college}, N_{student}\}$

CORE = REDUCT1 $\cap$ REDUCT2

= $\{N_{college}\}$

Let $C = \{N_{college}, N_{Department}, N_{student}\}$ and $D = \{GRF\}$ be set of condition and decision attributes respectively.

Discernibility Matrix is

<table>
<thead>
<tr>
<th>$X \in O$</th>
<th>U1</th>
<th>U2</th>
<th>U3</th>
<th>U4</th>
<th>U5</th>
<th>U6</th>
<th>U7</th>
<th>U8</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>---</td>
<td>${NC, ND, NS}$</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>${NC, ND}$</td>
<td>${NC, ND, NS}$</td>
</tr>
<tr>
<td>U2</td>
<td>${NC, ND, NS}$</td>
<td>---</td>
<td>${NC, ND, NS}$</td>
<td>---</td>
<td>---</td>
<td>${NC, ND, NS}$</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>U3</td>
<td>---</td>
<td>${NC, ND, NS}$</td>
<td>---</td>
<td>${NC, ND}$</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>${NC, ND, NS}$</td>
</tr>
<tr>
<td>U4</td>
<td>---</td>
<td>---</td>
<td>${NC, ND}$</td>
<td>---</td>
<td>${NC, ND, NS}$</td>
<td>${NC, ND}$</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>U5</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>${NC, ND, NS}$</td>
<td>---</td>
<td>---</td>
<td>${NC, ND, NS}$</td>
<td>${NC, ND, NS}$</td>
</tr>
<tr>
<td>U6</td>
<td>---</td>
<td>${NC, ND, NS}$</td>
<td>---</td>
<td>${NC, ND}$</td>
<td>---</td>
<td>---</td>
<td>${NC, ND}$</td>
<td>${NC, NS}$</td>
</tr>
<tr>
<td>U7</td>
<td>${NC, ND}$</td>
<td>---</td>
<td>---</td>
<td>${NC, ND, NS}$</td>
<td>---</td>
<td>${NC, ND}$</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>U8</td>
<td>${NC, ND, NS}$</td>
<td>---</td>
<td>${NC, ND, NS}$</td>
<td>---</td>
<td>${NC, ND, NS}$</td>
<td>---</td>
<td>${NC, NS}$</td>
<td>---</td>
</tr>
</tbody>
</table>

where NC, ND, NS represents N_{college}, N_{Department}, N_{student} respectively.

From table2, the decision relative discernibility function is (with duplicates removed):

$$f_{D}(N_{college}, N_{department}, N_{student}) = (N_{college} \lor N_{department} \lor N_{student}) \land (N_{college} \lor N_{department}) \land (N_{college} \lor N_{student})$$

$$= (N_{college} \lor N_{department}) \land (N_{college} \lor N_{student})$$

Therefore, the minimal reducts are $\{N_{college}, N_{department}\}$ and $\{N_{college}, N_{student}\}$.

6. Dynamic reducts

Let OGIS (O, AT $\cup \{d\}$) be a decision table. By P(OGIS) we denote the set of all subtables of OGIS. If F is a family of subtables of OGIS, then

$$DR(OGIS,F) = RED(OGIS,d) \cap \{ \bigcap_{OOGIS' \in F} RED(OGIS',d) \}$$

defines the set of F-dynamic reducts of OGIS.

From this definition of dynamic reduct it follows that a relative reduct of OGIS is dynamic if it is also a reduct of all subtables from a given family F. In most cases, this is too restrictive, so a more, general notation of dynamic reducts is required.
By introducing a threshold, $0 \leq \varepsilon \leq 1$, the concept of $(F, \varepsilon)$-dynamic reducts can be defined:
\[ DR_\varepsilon(OOIS, F) = \{OOIS' \in RED(OOIS, d) : S_F(OOIS') \geq \varepsilon \} \]  
(37)

where $S_F(OOIS') = \frac{|\{OOIS' \in F : OOIS' \in RED(OOIS', d)\}|}{|F|}$ is the F-stability coefficient of C. This gives the restriction that a dynamic reduct must appear in every generated subtable. Now, a reduct is considered to be dynamic if it appears in a certain percentage of subtables, determined by the value $\varepsilon$. For example, by setting $\varepsilon$ to 0.5, a reduct is considered to be dynamic if it appears in at least half of the subtables. Note that if $F = \{OOIS\}$, then $DR(\{OOIS\}, F) = RED(\{OOIS\}, d)$. Dynamic reducts may then be calculated according to the algorithm given below. First, all reducts are calculated for the given object oriented information system, OOIS. Then, the new subsystem $OOIS_j$ are generated by randomly deleting one or more rows from OOIS. All reducts are found for each object oriented subsystem, and the dynamic reducts are computed using $S_F(OOIS^*, R)$ which denotes the significance factor of reduct C within all reducts R.

**Dynamic Reduct algorithm :**
- Dynamic RED(OOIS, $\varepsilon$, $n_t$)
- OOIS, the object oriented decision table;
- $\varepsilon$, the dynamic reduct threshold;
- $n_t$, the number of iterations.

1. $R \leftarrow \{ \} $
2. $OOIS \leftarrow$ calculate all reducts (OOIS)
3. for (j =1 ; j <= $n_t$ ; j++)
4. $OOIS \leftarrow$ delete random rows (OOIS)
5. $R \leftarrow R \cup$ calculate AllReducts (OOIS$_j$)
6. $\forall$ $OOIS^* \in OOIS$
7. If $S_F(OOIS^*, R) \geq \varepsilon$
8. output $OOIS^*$

This is based on the hope that by finding stable reducts they will be more representative of the real world, i.e., it is more likely that they will be reducts for unseen data. A comparison of dynamic and nondynamic approaches can be found in [4], where various methods were tested on extracting laws from decision tables. In the experiments, the dynamic method and the conventional rough set method both performed well. In fact, it appears that the Rough set method has, on average, a lower error rate of classification than the dynamic rough set method.

**Conclusion :** We have introduced dynamic reducts in object oriented information system using rough set theory. We have proposed object-oriented information system into rough set models. Moreover, we have introduced indiscernibility relations on the set of objects, lower and upper approximations, and object-oriented rough sets in the object-oriented information systems and we have introduced the concept of object oriented reducts, object oriented reducts using indiscenibility matrix.

We can develop techniques for object oriented dynamic reducts using probability and we can also develop decision and dynamic rules for object oriented dynamic reducts.

**References :**