International Journal of Algebra, Vol. 4, 2010, no. 17, 827 - 842

Genetic Algorithm for Object Oriented Reducts Using Rough Set Theory

N. Ravi Shankar[1], T. Srikanth[2], B. Ravi Kumar [2] and G. Ananda Rao[1]

[1]Dept. of Applied Mathematics, GIS, GITAM University, Visakhapatnam, India

[2] Dept. of Computer Science, GIS, GITAM University, Visakhapatnam, India

Abstract

Knowledge reduction is NP-hard problem. Many approaches are proposed to get the minimal reduction, which is mainly based on the significance of the attributes. There are some disadvantages of the reduction algorithms at present. In this paper,. We propose a heuristic algorithm based on C-Tree for object –oriented reducts and also present a genetic algorithm (GA) for object oriented reducts based on C-Tree using rough set theory.

Keywords: Condensing Tree, object oriented reduct, rough set, Genetic algorithm

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.

Knowledge reduction is one of the most important problems in areas of data mining, pattern recognition and machine learning. It can remove redundant properties, and effectively simplify the knowledge and enhance learning efficiency and lower cost of classification. Minimal reduction that has been proved is a kind of NP-hard problem [9], so researchers have to propose some heuristic reduction algorithms which generally make use of attribute's significant degree as a kind of heuristic information to get the better result.

Genetic algorithm (GA) is a kind of effective searching and optimizing technique which has the characteristics of implicit parallelism, robust and global search, and has been applied to many fields. Bjorvand [10] used GA to calculate minimal reduct, but he didn’t make use of any heuristic information. What’s more, it is complicated to determine the fitness. In this paper, we introduced a heuristic strategy into generic algorithm and proposed a heuristic genetic algorithm to find object oriented reducts using rough set theory. We consider relative minimal reduct in this paper.

The rest of this paper is organized as follows. In Section2, 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 discernibility matrix to generate condensing tree (C-Tree). In section 5, we developed a heuristic algorithm based on C-Tree for object oriented reducts using rough set theory. In Section 6, we presented a genetic algorithm(GA) for object oriented reducts based on C-Tree using rough set theory.

2. Rough Sets and Information Systems

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 $a : U \to v_a$ 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:

$$xR_By \Leftrightarrow a(x) = a(y) , \forall \ a \in B \qquad \dots \qquad (1)$$

where a(x) means the value of the object $x \in U$ at the attribute a. $x\ R_B\ y$ means that we can not 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 R_B that contain x as $[x]_{R_B}$. 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:

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

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

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: $\underline{R_B}(X) \subseteq \text{X} \subseteq \overline{R_B}(X)$.

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 $\underline{R_B}(X) \subset \text{X} \subset \overline{R_B}(X)$, *X* is called R_B-rough.

Quality of approximation of *X* by the rough set R$_B$(X) is numerically evaluated as follows:

$$\frac{|\underline{R_B}(X)|}{|\overline{R_B}(X)|} \quad \dots \quad (4)$$

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 :

$$POS_P(Q) = \bigcup_{X \in U/Q} \underline{\sim_P}(X) \quad \dots \quad (5)$$

$$NEG_P(Q) = U - \bigcup_{X \in U/Q} \overline{\sim_P}(X) \quad \dots \quad (6)$$

$$BND_P(Q) = \bigcup_{X \in U/Q} \overline{\sim_P}(X) - \bigcup_{X \in U/Q} \underline{\sim_P}(X) \quad \dots \quad (7)$$

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 information system where C is set of condition attributes and D is 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 POS$_R$(D) = POS$_C$(D).The set of all the condition attributes indispensable in T is denoted by CORE (C) = $\cap$ RED (C) where RED (C) is the set of all reducts of C.

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** is the following triple:

$$(C, R_c, S_c) \quad --- \quad (6)$$

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

$\nexists \; c_1,c_2 ,\ldots c_n \in C$ 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:

$\forall\, c_i, c_j, c_k \in C \;, c_i S_c c_j \;, \; c_j R_c c_k \Rightarrow c_j R_c c_k \quad \ldots.$ (7)

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 [4] and order-sorted logic [5].

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.

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, $c \not R_c c'$ 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:

$$AT = \{c \in C / c \not R_c c' , \forall\, c' \in C \} \quad \ldots \quad (8)$$

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

C={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.

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:

$$(O, R_o, S_o) \quad \dots \quad (9)$$

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:

$\forall\ o_i, o_j, o_k \in O,\ o_i\ S_o\ o_j,\ o_j\ R_o\ o_k \Rightarrow o_i\ R_o\ o_k$ … (10)

We intend that every object o ∈ O is an instance of some class c ∈ 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. $\forall\ o_i, o_j \in O,\ o_i\ R_o\ o_j \Rightarrow id_C(o_i)\ R_c\ id_c(o_j)$ … (11)
2. $\forall o_i \in O, \forall c_j\ c\ C, id_C(o_i)\ R_c\ c_j \Rightarrow \exists\ o_j \in O s.t. o_i\ R_o\ o_j$ and $id_c(o_j) = c_j$ … (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:

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

where N is a finite non-empty set such that $|C| \le |N|$, 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 :

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

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

$$H_n^{f_n}(c/n_i) = H_n^{f_n}(c/n_j) \quad (17)$$

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,ndept, nstudents} and the following relationships:

Has- a relation: : university S_N college,
college S_N department,
university S_N Department,
………….

Offers-relation :
College R_N Department
Department R_N Courses.

Moreover, suppose we have a naming function f_n : N $\rightarrow$ C such that
f_n (university) = University,
f_n (college) = f_n (college2) = College,
f_n(department) = Department,
f_n (faculty) = Faculty,
f_n(student) = Student,
f_n(course) = Course,
f_n(ncollege) = Ncollege,
f_n(ndepartment) = Ndepartment,
f_n(nstudent) = Nstudent.
Note that we have H_N (College/university) = {college, college2}, and
H_N (Ndepartment/ college) = H_N (Ndepartment/ college2) = {ndepartment}.
Here, to illustrate connection between the classes and names, we use class diagrams of UML[7] authorized by OMG[8] 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.

University	
College	college
College	college2
Student	student

College	
Department	department
Student	student
Faculty	faculty
Ncollege	ncollege

Department	
Course	course
Faculty	faculty
Student	student

Faculty	
Ndepartment	ndepartment

Course	
Ndepartment	ndepartment

Student	
Nstudent	nstudent

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 → 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 ∈ O, if $H_O(x) \neq \phi$, the restriction of a_n into $H_O(x)$:

$$a_n / H_O(x) = H_O(x) \to N \quad \ldots \qquad (18)$$

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 → 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 → C and a name assignment a_n: O → N such that

$$id_C = f_n \circ a_n \ldots \qquad (19)$$

that is, $id_C(x) = f_n(a_n(x))$ for all x ∈ O .

Definition 10 : Let **C, N** and **O** be well defined structures. Suppose we have $o_1,o_2,\ldots o_k \in O$, $n_1, n_2,\ldots n_k \in N$, and $c_1,c_2,\ldots c_k \in C$ 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 $o_1.n_2\ldots n_i$ 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-1}…as a part of o_1".

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.
x S_O x , $\forall$ x $\in$ O and
university3 S_O university1 , university3 S_O university2.
Moreover, let a_n :O $\rightarrow$ N be the following name assignment :

a_n(university1) = a_n(university2) = college,

a_n(university3) = college2,

a_n(c1) = a_n(c2)= a_n(c3) = college,

a_n(c4) = college2,

a_n(s1) = a_n(s2) = a_n(s3) = student,

a_n(f1) = a_n(f2) = a_n(f3) = faculty,

a_n(24) = a_n(16) = ncollege,

a_n(150) = a_n(120) = ndepartment,

a_n(2400) = a_n(1200) = nstudent.

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.

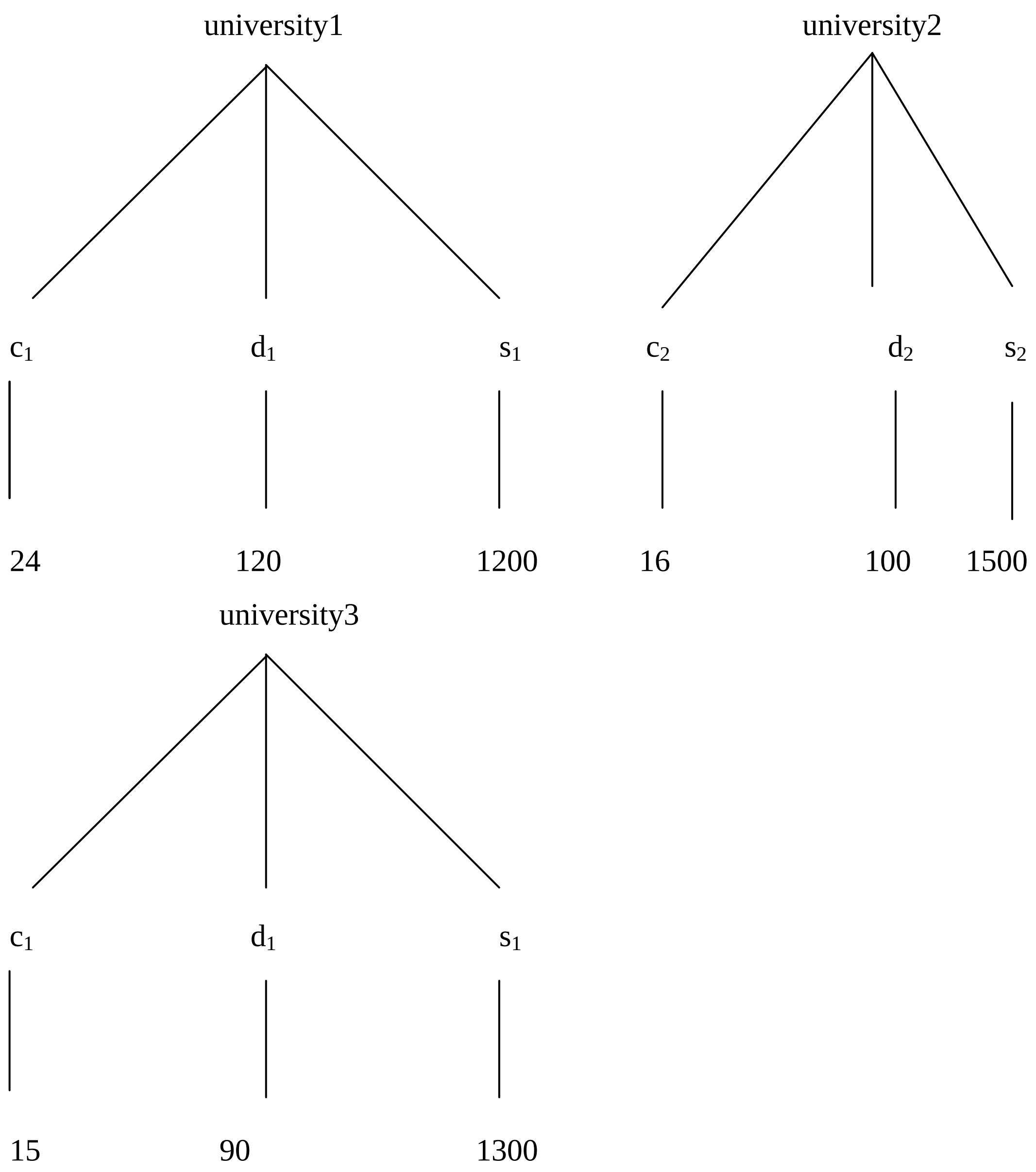

Fig 2. Offers- a relation on objects in example3

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 OOIS(**O,C,N**) is the following structure:

OOIS(**O,C,N**)= (O,C,N,o,id_c) … (20)

where $id_c = f_n \circ a_n$

The object oriented information system can be illustrate "traditional" information system as special case. In particular ,for any information system IS(U,A) ,we can construct an object oriented information system OOIS($\mathbf{O}_{IS}$, $\mathbf{C}_{IS}$,$\mathbf{N}_{IS}$) that corresponds to IS: First, using the information system IS = (U,A), we construct a name structure.

$\mathbf{N}_{IS} = (N_{IS}, R_{N_{IS}}, S_{N_{IS}})$ as follows :

$N_{IS} = A \cup \{IS\}$

$R_{N_{IS}} = \{(s,a) / a \in A\}$

$S_{N_{IS}} = \{(n,n) / n \in N_{IS}\}$

Where s is a symbol that doesnot appear in A. We also construct and object structure

$\mathbf{O}_{IS} = (O_{IS}, R_{O_{IS}}, S_{O_{IS}})$ as follows :

$\mathbf{O}_{IS} = U \cup (\bigcup_{a\in A} \{v_a^x / \exists a, \exists x, v \in v_a, a(x) = v\})$

$R_{O_{IS}} = \{ (x, v_a^x) / x \in U\}$

$S_{O_{IS}} = \{(o,o) / o \in O_{IS}\}$

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 $\mathbf{C}_{IS} = (C_{IS}, R_{C_{IS}}, S_{C_{IS}})$ as $C_{IS} = N_{IS}$, $R_{C_{IS}} = R_{N_{IS}}$ and $S_{C_{IS}} = S_{N_{IS}}$.

Finally, we construct a name assignment $(a_n)_{N_{IS}}$, a naming function $(f_n)_{C_{IS}}$, and a class identifier $id_{C_{IS}}$, respectively. Suppose a function $(a_n)_{N_{IS}} : O_{IS} \to N_{IS}$ by

$$(a_n)_{N_{IS}}(o) = \begin{cases} s \text{ if } o \in U \\ a \text{ if } o \in v_a^x, \exists x \in U \end{cases} \qquad \cdots \qquad (21)$$

The function $(a_n)_{N_{IS}}$ becomes a name assignment : if $o \in U$,then we have $H_O(o) = \{ v_a^o / a \in A\}$, that is, the set of value objects about o, and by the construction of value objects v_a^o, each v_a^o and $a = (a_n)_{N_{IS}} v_a^o \in N_S$ corresponds one to one. Otherwise, we have $o = v_a^x$, and therefore $H_O(o) = \phi$. We define the naming function $(f_n)_{C_{IS}} : N_{IS} \to C_{IS}$ by $(f_n)_{C_{IS}}(n) = n \in C_{IS}$ for all $n \in N_{IS}$.

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

OOIS ($\mathbf{O}_{IS}$, $\mathbf{C}_{IS}$,$\mathbf{N}_{IS}$) satisfies the following property : $a(x) = v \Leftrightarrow val(x,a) = v$, $\forall x \in U$,
$\forall a \in A$.

Definition 11 : 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 :

$\wedge c_i \Rightarrow d_i$ where $c_i \in AT$ and $d_i \in D$ (i= 1, 2,…|AT|) … (22)

4. Generation of condensing tree (C-Tree)

4.1 Discernibility Matrix

A decision table is denoted by $DT = \langle O, AT \cup D, V, f \rangle$ where $O = \{o_1, o_2, \dots, o_n\}$ is a non-empty finite set of objects or cases called Universe, where AT is the set of conditional attributes and D is the decision attributes, $AT \cap D = \phi$. In this paper, D = {d} is a singleton set, where D is the class attribute that denotes classes of objects.

$f : O \times (AT \cup D) \rightarrow V$ is a total function such that $f(o_i, a) = v_a$ for each $a \in (AT \cup D)$,

$o_i \in O$, where v_a is domain of the attribute a. Throughout this paper, ϕ denotes empty set, and |X| denotes the function that returns the cardinality of the argument set X. Given a decision table DT, a discernibility matrix DM [ ] is defined as an $n \times n$ matrix of DT with the $(i,j)^{th}$ entry m_{ij} is given by

$$m_{ij} = \{a \in AT : f(o_i, a) \neq f(o_j, a)\} \text{ for } f(o_i, D) \neq f(o_j, D)$$

$$= \phi \text{ otherwise} \qquad \dots \qquad (23)$$

An attribute subset R of C is an attribute reduction iff $R \cap m_{ij} \neq \phi$ holds for each $m_{ij} \in DM$ ($m_{ij} \neq \phi$) , and for every $S \subset R$, $\exists\, m_{ij} \in DM$ ($m_{ij} \neq \phi$) such that $S \cap m_{ij} = \phi$. An attribute subset R of C is an approximate attribute reduction if and only if $|\{m_{ij} / R \cap m_{ij} \neq \phi\}| \geq \delta * N, |\{m_{ij} / S \cap m_{ij} \neq \phi\}| < \delta * N, \forall S \subset R,$ where N is the number of non-empty entries in DM, $\delta \in [0.8,1]$.

4.2 Construction of Condensing Tree (C-Tree)

For a given disecernibility matrix(DM) , to efficiently compress and not lost information of DM, the so called C-Tree structure, a compact data structure was introduced, the information of DM can also be compressed but not lost by C-Tree structure.

A Condensing tree is a tree structure defined as given below.

1. It consists of one root labeled as “null” a set of attribute (or attribute index) prefix subtrees as the children of the root, and an attribute (or attribute index) header table.
2. Each node in the attribute prefix subtree consists of six fields : attribute-name, count, stcount, parent, childhead, and node-link, where attribute name registers, which attribute this node represents, count registers the number of cells of a discernibility matrix represented by the portion of the path reaching this node,

stcount registers the sum of all count fields of its child nodes. Parent points to its parent node, child head points to the head child of all children and node-link to the next node in the C-Tree carrying the same attribute name, or null if there is none.
3. Each entry in the attribute header table consists of three fields : attribute-name, frequency, and head of the node-link, where frequency registers the number of entries including the attribute represented by attribute-name appears in corresponding discernibility matrix, head of node-link points to the first node in the C-Tree carrying the attribute-name.

According to this definition, the C-Tree generation algorithm is described as follows.

Algorithm 1 Generating C-Tree (AT,D,O)

Input : AT : Conditional attributes, D: decision attributes O : objects
Output : Its condensing tree, C-Tree T.

```
1. Set a proper order of attributes R;
2. Create the root of an C-Tree T, and label it as "null";
3. Create the header table HT[1…|AT|], according to order R, get every attribute f
in turn and set its attribute-name, frequency and head of node-link be f-attribute-
name, O and NULL, respectively.
4. for (i =1;i<=|O|;i++)
   for ( j =1 ; j <= i-1;j++)
4.1 generate an element $m_{ij}$ of and DM by (1);
4.2 if $m_{ij} \neq \phi$ then
4.2.1 sort the attributes $m_{i,j}$ according to the order of R. Let the sorted attribute
test in $m_{ij}$ be [f/F] , where f is the first attribute and F is the remaining list.
4.2.2 call insert_tree ([f/F] , T);
5   return T.
```

In algorithm 1, the function insert_tree ([f/F],T) is performed as follows. If T has a child N such that N.attribute_name=f.attribute_name, then increment N's count by 1; else create a new node N, and set its count be 1, its parent link be linked to T, and its node link be linked to the nodes with the same attribute_name via the nod-link structure. If F is non-empty, call insert_tree (F,N) recursively. The order of attributes, R, is usually set the order obtained by choosing attributes from left to right in a decision table.

5. A heuristic algorithm based on C-Tree for object –oriented reducts.

Algorithm 2 : OReductBtree (AT,D,O)

Input : AT : Conditional attributes, D: Decision attributes , O : objects.

Output R : Object oriented reducts $R \subseteq AT$.

```
1. R ← ϕ , A ← AT;
2. T ← Generating C-Tree (AT,D,O);
3. do
4. a ← select highest frequency attribute (T);
5. R← R  ∪ {a}
6. I ← locate (a,HT);
/* the position of attribute a in header table HT, that is, HT[i] attribute_name
is the attribute a */
7. p ← HT [i]. head of node-link;
8. HT[i]. frequency ← 0;
9. while p ≠ ϕ do
update count and stcount of each node along the parent and childhead;
update the frequency of the attribute in HT;
delete each node that its count equals zero;
p← p → node-link ;
10. A ← A- {a};
11. until T = ϕ;
12. return R.
```

OReductBtree can only obtain one attribute subset. To get more useful approximate object oriented reducts will be introduced in section 4.

6. A genetic algorithm (GA) for object oriented reducts based on C-Tree.

Generating algorithms deal with a population of individuals by using selection, crossover and mutation operators. A population of individuals is repeatedly evolved over generations by optimizing a fitness function, which provides a quantitative measure of the fitness of individuals in the pool. Selection operator chooses better individuals to participate into crossover, i.e., those individuals with high fitness values. Cross-over operator is responsible for creating new individuals from the old ones. Mutation also generate new individuals, but only in the vicinity of old individuals.

Generally, solutions are represented by binary strings of length m, where m is the number of conditional attributes. In the bit representation ‘1’ means that attribute is present and ‘0’ means that it is not. Here a new fitness function based on discernibility matrix, for attribute reduction is defined as follows,

$$F(v) = (m-L_v)/m + |AT_v| /n \quad \dots \qquad (24)$$

Where v is an individual, i.e., an object oriented reduct candidate, m is the number of conditional attributes, L_v is the number of 1’s in V, AT_v is the set of non-empty entries in which some attributes hidden in individuals appear, N is the number of non-empty entries in DM.

To obtain potential and more useful individuals, the so-called "Roulette Wheel' strategy is employed, this strategy makes those attributes with relatively low frequencies can also be chosen. Moreover Eq.2 should be divided into two parts F_1 and F_2.

$$F_1 = \frac{(m - L_v)}{m} \quad \dots \qquad (25)$$

$$F_2 = \frac{|AT_v|}{n} \quad \dots \qquad (26)$$

A genetic algorithm for object oriented reducts based on C-tree is summarized as follows :

Algorithm 3. CTBGAOOR(AT,D,O,δ)

```
/* A Novel Condensing tree based genetic algorithm for object oriented reducts*/
Input : AT : Conditional attributes, D: Decision attribute, O: objects, $\delta$ : pre-set thresholds:
Output : More useful approximate object oriented reducts.
1. parameter initialization :
   $p_c \leftarrow$ crossover probability ;
   $p_m \leftarrow$ Mutation probability ;
   T $\leftarrow$ Maximum number of iterations;
   K $\leftarrow$ 0;
2. T$\leftarrow$ Generating C-Tree (AT,D,O);
3. getting the frequency of each attribute that appears in T;
4. p $\leftarrow$ A random population of size pop_size is generated by using the principle of 'Roulette Wheel' according to the frequency of each attribute;
5. computing three fitness values of each individuals: t $\leftarrow$ all individuals that their fitness values $F_2 \geq \delta$ ;
6. while (k <T) and |t| < 0.9 * |P|) do
6.1 select individuals using 'Roulette wheel' strategy;
6.2 crossover with $p_c$ ;
6.3 Mutation with $p_m$ ;
6.4 some uninteresting individuals are replaced and offspring individuals p are created;
6.5 The two fitness values for each individuals is repeated by calculate d;
6.6 k$\leftarrow$k+1;
7 Choose some sub optimal individuals.
```

Conclusion

We have proposed a heuristic algorithm based on C-Tree for object –oriented reducts. We have presented a genetic algorithm (GA) for object oriented reducts based on C-Tree using rough set theory.

References

[1] Z.Pawlak, “Rough sets”, International Journal of Computer and Information Sciences 11,1982, 341-356.

[2] Z.Pawlak, Rough sets : Theoretical Aspects of Reasoning about Data, vol.9, kluwer Academic Publishers, Dorderecht, The Netherlands, 1991.

[3] Z.Pawlak, A. Skowron, “Rudiments of rough sets”, Information Sciences ,1, 2007,3- 27.

[4] R. Socher-Ambrosius and P. Johann, “Deduction Systems,” Springer, 1996.

[5] K. Kaneiwa, “Order-Sorted Logic Programming with Predicate Hierarchy,”Artificial Intelligence, Vol.158, pp. 155-188, 2004.

[6] S. Popkorn, “First Steps in Modal Logic,” Cambridge University Press, 1994.

[7] http://www.uml.org/

[8] http://www.omg.org/

[9] Wong S K M, Ziarko W. "On optimal decision rules in decision tables", Bulletin of the Polish Academy of Sciences: Mathematics, 1985, 33(11-22): 693-696.

[10] Bojrvand, And ers Torvill. Times Series and Rough Sets : [Master’s Thesis]. The Norwegian Institute of Technology, Department of Computer Systems, Trondheim, Norway, 1996.

Received: March, 2010