Brock University

Department of Computer Science

Rough set data representation using binary decision diagrams

A. Muir, I. Düntsch, and G. Gediga
Technical Report # CS-04-03
February 2004

Brock University
Department of Computer Science
St. Catharines, Ontario
Canada L2S 3A1
www.cosc.brocku.ca

Rough set data representation using binary decision diagrams

Alex Muir, Ivo Düntsch, Günther Gediga
Department of Computer Science
Brock University
St Catharines, ON, Canada, L2S 3A1
{am98ap|duentsch|gediga@cosc.brocku.ca}

February 3, 2004

Abstract

A new information system representation, which inherently represents indiscernibility is presented. The basic structure of this representation is a Binary Decision Diagram. We offer testing results for converting large data sets into a *Binary Decision Diagram Information System* representation, and show how indiscernibility can be efficiently determined. Furthermore, a Binary Decision Diagram is used in place of a relative discernibility matrix to allow for more efficient determination of the discernibility function than previous methods. The current focus is to build an implementation that aids in understanding how binary decision diagrams can improve Rough Set Data Analysis methods.

Keywords: Rough Set Data Analysis, Binary Decision Diagrams, Indiscernibility, Discernibility Function

1 Introduction

As one of the most referenced data structures in Computer Science the uses of Binary Decision Diagrams (BDD) [5, 15] are too numerous to mention. R.E. Bryant [13] popularized BDD when he observed that *reduced ordered BDD* (ROBDD) are a canonical representation of Boolean functions. More recently, the work of Berghammer et al. [3] has shown "how relations and their operations can efficiently be implemented by means of BDD ... by demonstrating how they ... can be applied to attack computationally hard problems".

Rough Set Data Analysis (RSDA), developed by Z. Pawlak [11] and his co–workers in the early 1980s has become a recognized and widely researched method with over 2500 publications to date[1]. As part of the RSDA process, indiscernibility among objects and attributes within an information system are determined in order to perform rule dependency generation. Previously methods of RSDA were not feasible with large data tables (e.g. with over 10,000 objects)even on powerful workstations. Nguyen and Nguyen [9]. Demonstrating more efficient reduct determination than previous methods, we offer testing results using UCI data sets with more than 10,000 objects.

[1] We thank A. Skowron for this estimate

2 Rough set data analysis

2.1 Basics

The basic idea of RSDA is that objects are only discernable up to a certain granularity. For example, if we only know that a hit and run car was a red BMW, then, within this framework, all red BMWs are the same to us. It is the aim of the police to increase the granularity of the information so that, finally, the culprit's car is distinguishable from all others. The mathematical tool of RSDA are *indiscernibility relations*, which are simply equivalence relations, i.e. they are reflexive, antisymmetric and transitive. Data representation in RSDA is done via *information systems*, which can be described as tables of

$$\text{Object} \mapsto \text{Feature–vector}$$

relationships. More formally, an information system is a structure $I = \langle U, \Omega, \{V_a : a \in \Omega\}\rangle$, where

- U is a finite non–empty set of objects.
- Ω is a finite set of mappings $a : U \to V_a$, called *attributes*.

Throughout the rest of the paper we will use $I = \langle U, \Omega, \{V_a : a \in \Omega\}\rangle$ as a generic information system with $|U| = n$ and $|\Omega| = k$. We can think of $\langle a, a(x)\rangle$ as a descriptor which assigns value $a(x)$ to object x for attribute a. Since both U and Ω are finite, we can picture an information system as a table where the rows are labeled by the objects and the columns are labeled by the attributes. An example is shown in Table 1. In terms of measurement

Table 1: A simple information system

U	a	b	c	d
x_1	15	26	36	27
x_2	15	26	18	19
x_3	14	37	18	48
x_4	14	37	18	48

theory, RSDA operates on a nominal scale, where only (in-)equality is recognized. Each set Q of attributes determines an indiscernibility relation θ_Q on U by setting

$$x\theta_Q y \iff (\forall a \in Q)[a(x) = a(y)]. \tag{2.1}$$

If $Q = \{a\}$, we usually just write θ_a. Observe that for $P, Q \subseteq \Omega$,

$$P \subseteq Q \Rightarrow \theta_Q \subseteq \theta_P, \tag{2.2}$$

$$\theta_P = \bigcap\{\theta_a : a \in P\}. \tag{2.3}$$

The complexity of determining indiscernibility relations is given by

Lemma 1. *[9] If $Q \subseteq \Omega$, and $|Q| = k$, then θ_Q can be found in $O(k \cdot n\log(n))$ time and $O(n)$ space.*

One aim of RSDA is to eliminate features which are superfluous for a reclassification of the data; in other words, one looks for attribute sets Q which give the same classification of the objects as the full set Ω, and which are minimal with respect to this property; such sets are called *reducts*. Formally, a reduct Q is a set of attributes such that

1. $\theta_Q = \theta_\Omega$.
2. If $R \subsetneq \Omega$, then $\theta_\Omega \subsetneq \theta_R$.

Reducts correspond to keys of a relational database; consequently; as was pointed out in [12] the problem of finding a reduct of minimal cardinality is, in general, NP-hard, and finding all reducts has exponential complexity [17]. Clearly, each I has a reduct; the intersection of all reducts is called the *core*.

Let d be a new attribute with a set of values V_d and information function $d : U \to V_d$, and suppose that $\emptyset \neq Q \subseteq \Omega$. The aim is to relate the values an object has with respect to the attributes of Q to its value with respect to d. The new attribute is called the *dependent attribute* or *decision attribute*; the elements of Q are called *independent* or *condition* attributes. The structure $\mathbf{D} = \langle I, d \rangle$ is called a *decision system*; it is called *consistent* if

$$(\forall x, y \in U)[(\forall a \in \Omega) a(x) = a(y) \text{ implies } d(x) = d(y)]. \tag{2.4}$$

An example of a decision system is given in Table 2. There we interpret the decision variable as "Demand".

Table 2: A decision system

Type	Price	Guarantee	Sound	Screen	d
1	high	24 months	Stereo	76	high
2	low	6 months	Mono	66	low
3	low	12 months	Stereo	36	low
4	medium	12 months	Stereo	51	high
5	medium	18 months	Stereo	51	high
6	high	12 months	Stereo	51	low

A *local rule of* $\mathbf{D}$ is a pair $\langle K_Q, K_d \rangle$ where K_Q is a class of θ_Q, K_d a class of θ_d and $K_Q \subseteq K_d$. This is the case just when all elements that have the same attribute vector determined by K_Q, have the same decision value. In Table 2, we have, for example, the local rule $\langle \{2,3\}_Q, \{\text{low}\}_d \rangle$, where $Q = \{\text{Price}\}$. This rule can be read as

If Price = low, then Demand = low.

Therefore, we usually write $K_Q \to K_d$ instead of $\langle K_Q, K_d \rangle$. A class K_Q of θ_Q is called *d-deterministic* (or just *deterministic*, if d is understood), if there is some class K_d such that $K_Q \to K_d$. The union of all deterministic classes of θ_Q is called the Q-positive region of d, denoted by $\text{pos}(Q,d)$. The characteristic function of $\text{pos}(Q,d)$ is denoted by $\chi(Q,d)$.

If each class of θ_Q is deterministic, i.e. if $\text{pos}(Q,d) = U$, then we say that *d is dependent on Q*, and write $Q \Rightarrow d$. In this case, we call $Q \Rightarrow d$ a *global rule* or just a *rule*. A d-reduct now is a set of attributes which is minimal with respect to $Q \Rightarrow d$. If it is clear that we are considering decision systems, we will just speak of reducts.

2.2 Discernibility matrices and Boolean reasoning

Using the fact that RSDA operates on a nominal scale, an alternative way of representing discernibility is to cross-classify objects by assigning to each pair $\langle x, y \rangle \in U^2$ the set $\delta(x,y)$ of all those attributes a for which $a(x) \neq a(y)$ [17]; the result is called a *discernibility matrix*. The discernibility matrix for our example of Table 1 is shown in Table 3. It may be noted that time complexity of finding the indiscernibility matrix from a given information system is $O(k \cdot n^2)$ and that the space required to store the matrix is $O(k \cdot n^2)$ as well.

Table 3: A discernibility matrix

U	x_1	x_2	x_3	x_4
x_1	$\emptyset$	$\{c,d\}$	Ω	Ω
x_2	$\{c,d\}$	$\emptyset$	$\{a,b,d\}$	$\{a,b,d\}$
x_3	Ω	$\{a,b,d\}$	$\{a,b,d\}$	$\emptyset$
x_4	Ω	$\{a,b,d\}$	$\emptyset$	$\emptyset$

Associated with a discernibility matrix is a *discernibility function*, which is a frequently used tool to handle reducts [17]. First, we need some preparation from Boolean reasoning: Suppose that $\mathbf{2} = \langle \{0,1\}, \wedge, \vee, -, 0, 1\rangle$ is the two element Boolean algebra. A *Boolean function* is a mapping $f : \mathbf{2}^n \to \mathbf{2}$, where $1 \leq n$, and $\mathbf{2}^n = \overbrace{\mathbf{2} \times \mathbf{2} \times \cdots \times \mathbf{2}}^{n-\text{times}}$. If $\vec{x}, \vec{y} \in \mathbf{2}^n$ we say that $\vec{x} \leq \vec{y}$ if $x_i \leq y_i$ for all $1 \leq i \leq n$. A Boolean function $f : \mathbf{2}^n \to \mathbf{2}$ is called *monotone*, if $\vec{x} \leq \vec{y}$ implies $f(\vec{x}) \leq f(\vec{y})$.

If $V = \{y_i : 1 \leq i \leq n\}$ is a set of variables, and $T \subseteq V$, we call T an *implicant of* f, if for any valuation of $\vec{y} \in \mathbf{2}^n$

$$y_i = 1 \text{ for all } y_i \in T \text{ implies } f(\vec{y}) = 1. \tag{2.5}$$

Observe that we can regard the left hand side of (2.5) as a conjunction, and we can equivalently write

$$\bigwedge T = 1 \Rightarrow f(\vec{y}) = 1. \tag{2.6}$$

Thus, an implicant gives us a sufficient condition for $f(\vec{y}) = 1$. A *prime implicant of* f is a subset T of V such that T is an implicant, but no proper subset of T has this property. Suppose that $\Omega = \{a_1, \ldots, a_n\}$, and $U = \{x_1, \ldots, x_n\}$. For each $a_i \in \Omega$, we let a_i^* be a variable, and, for $\delta(x_i, x_j) \neq \emptyset$, we define $\delta^*(x_i, x_j) = \bigvee\{a_r^* : a_r \in \delta(x_i, x_j)\}$. Now, the *discernibility function of* I is the formal expression

$$\Delta_I(a_1^*, \ldots, a_n^*) = \bigwedge\{\delta^*(x_i, x_j) : 1 \leq i \lneq j \leq m,\ \delta(x_i, x_j) \neq \emptyset\}. \tag{2.7}$$

We usually just write Δ if I is understood. The discernibility function of the matrix of Table 3 is therefore

$$\Delta(a^*, b^*, c^*, d^*) = (c^* \vee d^*) \wedge (a^* \vee b^* \vee d^*) \wedge (a^* \vee b^* \vee c^* \vee d^*).$$

The connection between reducts and the discernibility has been shown by Skowron and Rauszer [17]; for completeness, we provide a proof.

Proposition 1. *Q is a reduct of I if and only if Q is a prime implicant of Δ.*

PROOF. "$\Rightarrow$": Suppose that Q is a reduct of I, and let $\vec{y} \in \mathbf{2}^n$ be a valuation of $\{a_1^*, \ldots, a_n^*\}$ such that $y_i = 1$ for all $a_i \in Q$. We first show that Q is an implicant of Δ. Assume that $\Delta(\vec{y}) = 0$. Then, by definition (2.7), there are $1 \leq i \lneq j \leq n$ such that $x_i, x_j \in U$, $\delta(x_i, x_j) \neq \emptyset$ and $x_t = 0$ for all $a_t \in \delta x_i, x_j$. It follows that $\delta(x_i, x_j) \cap Q = \emptyset$, and therefore, Q does not distinguish between x_i and x_j. Since Q is a reduct, we have, in particular, $\theta_Q = \theta_\Omega$, so that, in fact, x_i and x_j cannot be distinguished by any attribute in Ω. Hence, $\delta(x_i, x_j) = \emptyset$, contrary to our assumption.

To show that Q is prime, suppose that $P \subseteq Q$ is an implicant of Δ, and assume there are $x_i, x_j \in U$ such that $x_i \theta_P x_j$, and Ω distinguishes x_i and x_j, i.e. $\delta(x_i, x_j) \neq \emptyset$. It follows from $x_i \theta_P x_j$ that $P \cap \delta(x_i, x_j) = \emptyset$. Let $\vec{y}$ be a valuation such that

$$y_i = \begin{cases} 1, & \text{if } a_i \in P, \\ 0, & \text{otherwise.} \end{cases} \tag{2.8}$$

Then, $\bigwedge\{y_i : a_i \in P\} = 1$, while $\Delta(\vec{y}) = 0$, contradicting the assumption that P is an implicant of Δ. It follows that $\theta_P \subseteq \theta_\Omega$, and the fact that Q is a reduct implies $P = Q$.

"$\Leftarrow$": Suppose that P is a prime implicant of Δ, and let $x_i \theta_P x_j$; then, $P \cap \delta(x_i, x_j) = \emptyset$. Assume that Ω distinguishes x_i and x_j, and choose a valuation $\vec{y}$ as in (2.8). By the same argument as above, we arrive at a contradiction. Since $\theta_P = \theta_\Omega$, P contains a reduct Q, and it is straightforward to see that Q is an implicant. It now follows from the fact that P is prime, that $Q = P$. ■

Corollary 1. *$P \subseteq \Omega$ is a reduct of I if and only if P is minimal with respect to the property*

$$P \cap \delta(x,y) \neq \emptyset \tag{2.9}$$

for all $x, y \in U,\ \delta(x,y) \neq \emptyset$.

We can define a *relative discernibility matrix* for a decision system $\mathbf{D} = \langle I, d \rangle$ in a slightly different way than for I: First, construct the discernibility matrix $\Delta_{\mathbf{D}}$ for the extended attribute set $\Omega \cup \{d\}$. In case of our TV example, the matrix is given in Table 4.

Table 4: Discernibility matrix $M_{\mathbf{D}}$

	2	3	4	5	6
1	Pr,Gu,So,Sc,d	Pr,Gu,Sc,d	Pr,Gu,Sc	Pr,Gu,Sc	Gu,Sc,d
2		Gu,So,Sc	Pr,Gu,So,Sc,d	Pr,Gu,So,Sc,d	Pr,Gu,So,Sc
3			Pr,Sc,d	Pr,Gu,Sc,d	Pr,Sc
4				Gu	Pr,d
5					Pr,Gu,d

Next we define a *relative discernibility matrix* $\overline{M}_{\mathbf{D}}$ by

$$\overline{\delta}(x_i, x_j) = \begin{cases} \delta(x_i, x_j) \setminus \{d\}, & \text{if } d \in \delta(x_i, x_j) \text{ and } x_i, x_j \in \text{pos}(\Omega, d) \\ & \text{or } \chi(\Omega, d)(x_i) \neq \chi(\Omega, d)(x_j), \\ \emptyset & \text{otherwise.} \end{cases} \tag{2.10}$$

The result for the TV example is shown in Table 5; note that $\text{pos}(\Omega, d) = U$.

Table 5: Relative discernibility matrix $\overline{M}_{\mathbf{D}}$

	2	3	4	5	6
1	Pr,Gu,So,Sc	Pr,Gu,Sc			Gu,Sc
2			Pr,Gu,So,Sc	Pr,Gu,So,Sc	
3			Pr,Sc	Pr,Gu,Sc	
4					Pr
5					Pr, Gu

Let $\Delta_{\overline{M}_{\mathbf{D}}}$ be the Boolean function belonging to $\overline{M}_{\mathbf{D}}$. Then,

Proposition 2. *[17] $\{a_{i_1}, \ldots, a_{i_t}\}$ is a reduct of $\mathbf{D}$ if and only if $a_{i_1} \wedge \ldots \wedge a_{i_t}$ is a prime implicant of $\Delta_{\overline{M}_{\mathbf{D}}}$.*

For our example, after absorption laws, we see that the relative discernibility function has the form

$$\Delta_{\overline{M}_{\mathbf{D}}}(\mathrm{Pr}^*, \mathrm{Gu}^*, \mathrm{So}^*, \mathrm{Sc}^*) = \mathrm{Pr}^* \wedge (\mathrm{Gu}^* \vee \mathrm{Sc}^*). \tag{2.11}$$

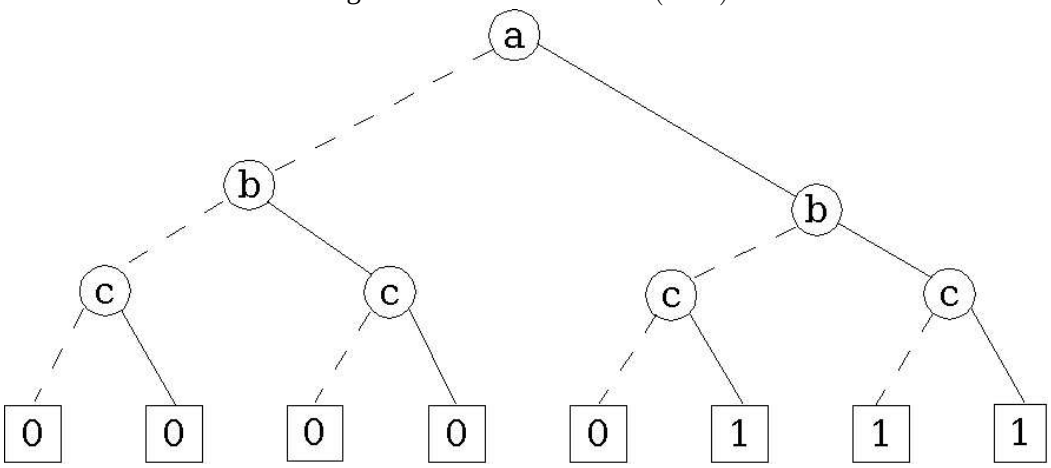
The prime implicants are {Pr,Gu} and {Pr,Sc}.

Much work has been done on employing methods of Boolean reasoning for reduct determination and rule finding, and we refer the reader to [7–9, 16, 17].

3 Binary Decision Diagrams

Given a n-ary Boolean function $f(x_1, \ldots, x_n)$, an *ordered binary decision diagram* (OBDD) [13] is a finite directed acyclic graph with one root, $n+1$ levels, and exactly two branches at each non-terminal node. One of these is the 0 case, denoted by $lo(x)$ and shown as a dashed line, the other the 1 case, denoted by $hi(x)$ and drawn as a solid line. The levels are determined by the (fixed) ordering of the variables $x_1 \gtrdot x_2 \gtrdot \ldots \gtrdot x_n$. Each traversal through the tree corresponds to an assignment to the variables, and the nodes at level $n+1$ give the evaluation of f corresponding through this traversal. Figure 1 shows an OBDD for the function $f(a,b,c) = a \wedge (b \vee c)$.

Figure 1: An OBDD for $a \wedge (b \vee c)$



The following reduction rules do not change the value of the function:

1. Use only one terminal label for 0 and one terminal label for 1 and redirect all lines from level n to the respective node.
2. If two non-terminal nodes x,y are on the same level and $lo(x) = lo(y),\ hi(x) = hi(y)$, then eliminate one of them, say, x and redirect all lines into x to y.
3. If $lo(x) = hi(x)$ for some non–terminal node x, then remove x and redirect all lines into x to $lo(x)$.

The result of applying these rules to an OBDD until none of them can be applied any more is called a *reduced ordered binary decision diagram* (ROBDD). Figure 2 shows the ROBDD of the function $f(a,b,c) = a \wedge (b \vee c)$.

A Shared Binary Decision Diagram Figure 3, is a multirooted directed acyclic graph that represents multiple Boolean functions [14]. Reuse of nodes by more than one Boolean function in the shared BDD allows for reduced memory use. A quasi-reduced BDD [10] is one that does not apply Rule 3 above resulting in each node having paths to one or more nodes one level below.

Figure 2: An ROBDD for $a \wedge (b \vee c)$

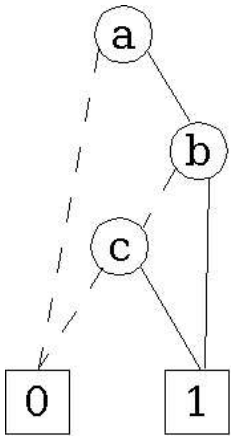
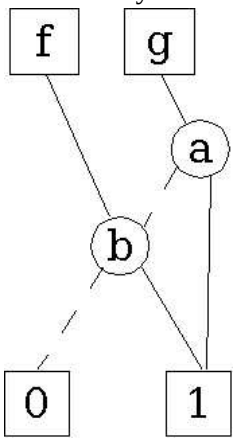


Figure 3: Shared Binary Decision Diagram



4 Binary Decision Diagram Information System

4.1 Representation

A *Binary Decision Diagram Information System*(BDDIS) is a quasi-reduced shared BDD which represents both data and indiscernibility within an information system. A BDDIS has one or more root nodes. Each unique root node has a subtree that may share nodes with other objects and represents one or more indiscernible objects. This level containing root nodes is defined as the *object-top level*; a top level element is referenced by $T(u)$, $u \in U$. The shared nature of the structure provides the mechanism for which indiscernibility in inherent. Every unique attribute value is represented by a unique node on one level of the roots subtrees defined as the *attribute-top level*. The current implementation has as leaf nodes the unique characters present within the information system. An alternative more efficient representation is discussed in Section 7

From the information system of Table 1 the BDDISs of Figures 4 and 5 are derived. Figure 4 is shaded to visualize the attribute-top level. Similarly, 5 is shaded to visualize the object-top level.

Each column within the information system is represented by a path composed of a unique series of lo and/or hi branches arriving at an attribute-top level node. In Figures 4 and 5, the first column is represented by the path lo-lo, the second a path lo-hi, the third a path hi-lo... Each node is labeled with Ln_x for level number *n* and node number *x*.

4.2 Indiscernibility

Two objects $j,k \in U$ are in the same equivalence class if $T(j)$ and $T(k)$ reference the same object-top level node. From Figure 5 we can derive the following object-indiscernible sets: $\{x_1\},\{x_2\},\{x_3,x_4\}$. Objects are indiscernible with respect to one attribute if by traversing a unique series of branches from each object root node the same attribute-top level node is attained. From Figure 4 it is clear that objects 2, 3, 4 have a common path hi-lo to node L2_5 representing the value 18 for attribute *c*. Thus the object 2,3,4 are indiscernible with respect to attribute *c*.

For a given data set, the size of the BDDIS decreases as the level of indiscernibility increases. The worst-case data that causes the BDDIS to grow as large as possible – relative to the size of the information system – is data containing all unique attribute values. The quasi-reduced structure is chosen to reduce the complexity associated

Figure 4: BDDIS: Shaded Attribute Top Level

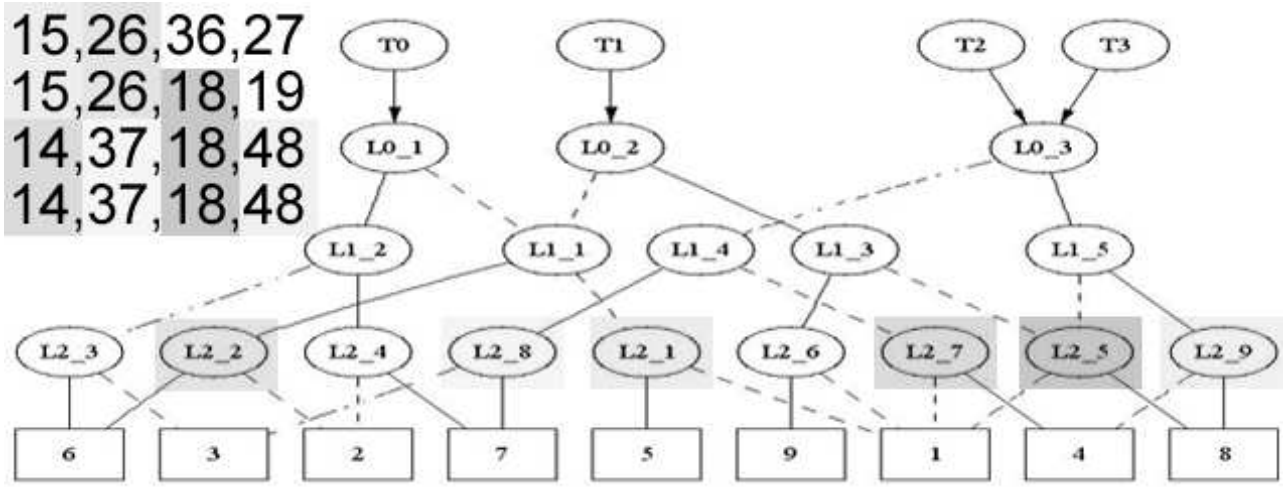
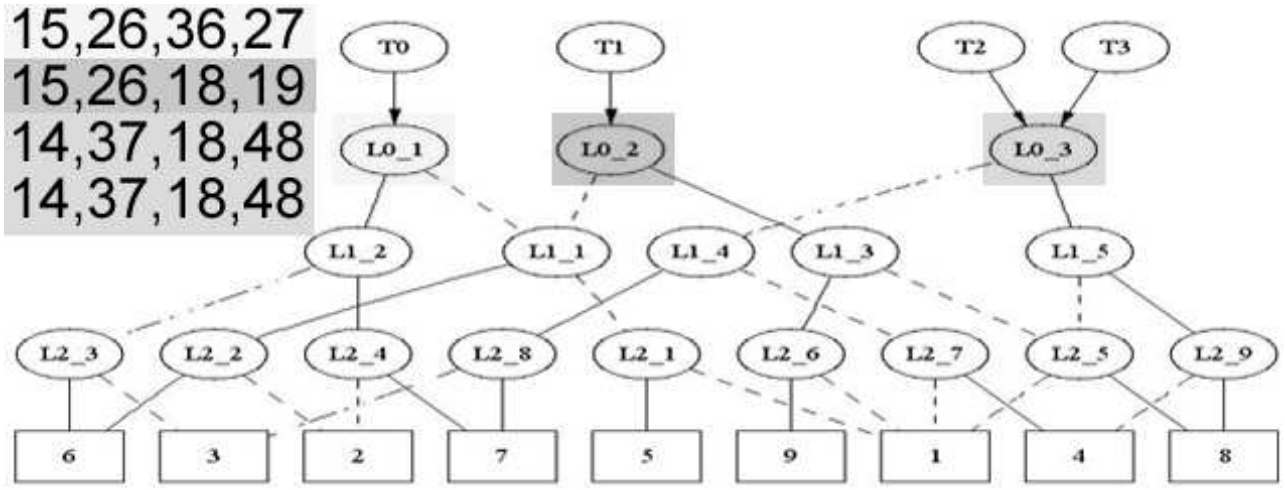


Figure 5: BDDIS: Shaded Object Top Level



with determining indiscernibility for the initial implementation. Methods which reduce BDDIS size are discussed in Section 7.

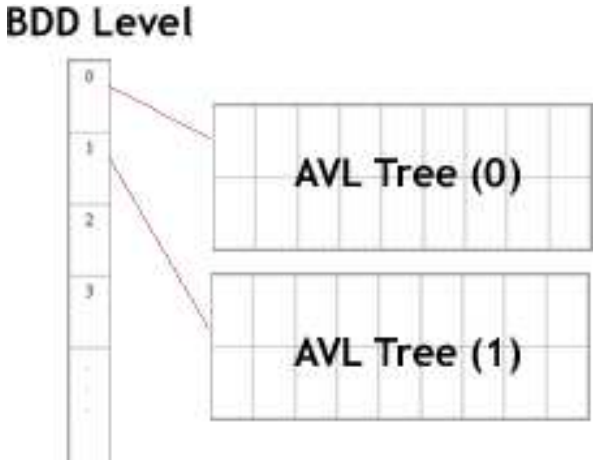
4.3 Construction

A *build tree* is a two-dimensional array used to convert one row of the information system at a time into one root node and one subtree within the BDDIS. The bottom row consists of null-padded attribute values as characters of which each character pair j and j+1 is stored as lo and hi keys within the nodes one level above the leaf nodes in the BDDIS. The leaf nodes only exist logically within these nodes. The attribute values are padded to the left with null characters to create attributes values of equal 2^k character length, allowing each attribute value to form a subtree of equal height in the BDDIS.

One *AVL tree* [1] for each BDDIS level ensures nodes are unique. Each node in the AVL trees is also a node within the BDDIS as well so that each AVL tree collectively forms the BDDIS. Each node contains keys, which

represent the location of the lo and hi children. The AVL trees use the *lo-key* as primary key and the *hi-key* as secondary key for insertions and rebalancing. Each AVL tree is stored in a two-dimensional array. The array index is used as the key of each node.

Figure 6: BDDIS as AVL Trees



For each level of the build tree, a probing/ insertion operation is performed into the corresponding AVL tree that returns the node key of either an existing node or a newly inserted node. The returned key is placed within build tree in the element corresponding to the parent of the lo and hi keys: $[\text{next level}][\frac{j+2}{2}-1]$. Upon each level of the build tree the same process is performed bottom up resulting with the top element in the build tree representing an object root node and the completion of the addition of an object into the BDDIS.

4.4 Order of operation

The conversion process involves AVL probing/insertion operations. For an object with 32 attributes having 16 characters each, there are a total of 31×15 probing operations. Only the last object probes the AVL trees holding nodes created from the insertion operations of all other objects before it. e.g the third object converted, probes the AVL trees holding nodes created by the insertion operations for the first and the second objects.

Figure 7 displays the relationship between the size of the AVL trees representing each level and the number of probe/insertion operations with worst-case data. Note that the number of probe/insertion operations decreases as the maximum size of the AVL trees increases, and that the size of the levels below the attribute-top level is much smaller as compared to the number of probe operations.

It is observed that for each level $i : (i \geq 1) : O(\text{num\_probe\_operations}(i)) \times log(\text{max\_AVL\_size}(i))$. With n objects, a attributes per object, c characters per attribute, z unique characters, the max AVL size below the attribute top level is restricted by either the number of possible combinations of digits z^{2^i} or largest number of nodes $na(c/2^i)$. It is clear that for some large value of n and a, the BDDIS will grow larger than available memory with worst case data. It is expected that c will, in general, be small in most data sets.

5 BDDIS Testing Results

The success of the representation is currently defined in terms of its speed, and demonstrated using data sets from the UCI data repository [4] with initial results given in Table 6. Furthermore, tests with random data, shown in

Figure 7: Exploration of order developed from AVL size and the number of probe operations

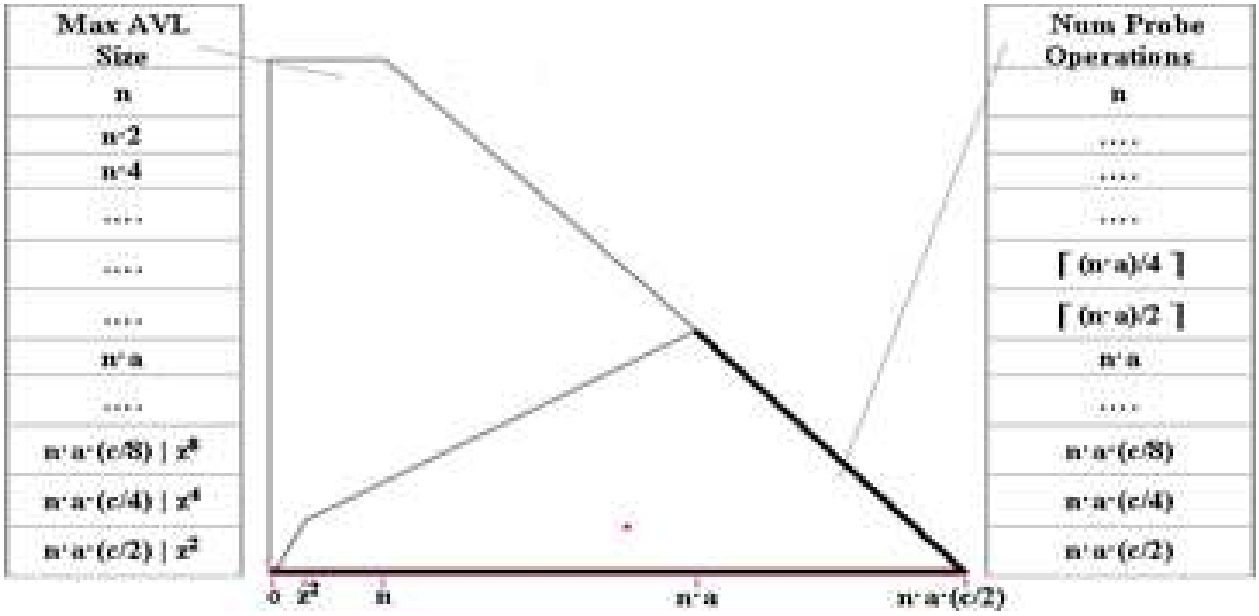


Table 7, illustrate the ability of the method to handle larger data sets.

Table 6: Conversion to BDDIS; UCI Data Sets; Pentium 4 1.8GHz

	Objects		Attributes		Size (KB)		
Data set	number	discernible	number	unique	Text	BDDIS	Time [*s*]
Iris	150	147	5	77	5	13	0.00001
Nursery	12960	12960	9	28	1047	440	1
Adult	32561	32537	15	22124	3913	3385	7
Cover Type	30000	30000	55	6399	3811	5626	13
Connect-4 Opening	67557	67557	42	6	5895	2120	5

In Table 6, the size of UCI data as text, and the time required to convert the data into the BDDIS, are listed. In addition, in order to gain a better understanding of the results, the size of the data along with the number of indiscernible objects and unique attributes within the information system is listed; in this way, we can represent the relationship of the level of indiscernibility and the size of the data with the BDDIS size and conversion time required. As the level of indiscernibility increases or the number of unique attributes increases, the AVL trees grows larger, and thus, the conversion time required and the memory used by the BDDIS are increased.

Table 7 displays results of the conversion into BDDIS representation on a Pentium 4 1.8GHz system using random 10-digit data sets of eight attributes and eight characters each. Random data cause the quasi-reduced BDDIS to grow unreasonably large as there is little opportunity for the BDDIS to reuse nodes because, in general, classes are small, and the large majority of attribute values are unique.

Table 7: Conversion to BDDIS; Random data sets; Pentium 4 1.8GHz

Number of objects	BDDIS (MB)	Time $[s]$
100000	24.4	40
200000	48.6	136
300000	72.6	281

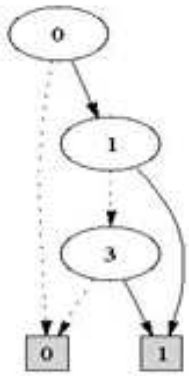
6 Discernibility functions using BDD

In this Section we shall describe a method to determine a Boolean discernibility function without the requirement to create a discernibility matrix. The current implementation uses the Buddy BDD package [6] to represent the Boolean discernibility function as a BDD.

Instead of creating a set of disjunctions for pairs of each object in the information system, the relative discernibility matrix can be created using pairs of *discernible object classes*. Each discernible object class is represented by a top node within the BDDIS. i.e each root node represents an object discernible from all other objects.

A depth first traversal of the BDDIS for every two discernible objects classes that are indiscernible with respect to their decision attributes is performed from top node to attribute top level node in order to create each set of disjunctions in turn. An AND operation is performed with each new set of disjunctions into what will eventually become the final BDD representing the discernibility function; the BDD is always in a reduced form. Figure 8 represents the Boolean function for Table 2. A reduct is derived from each possible set of unique paths leading to the leaf node 1, which for Figure 8 are $\{0,1\}$ and $\{0,3\}$. Buddy codes attributes using integers; in Figure 8, attribute 0 represents price, and 1 represents guarantee.

Figure 8: Discernibility function as a BDD for Table 2



6.1 Order of the operation

In the worst case, the operation requires $\frac{n(n-1)}{2}$ traversals of the BDDIS and AND operations into the BDD discernibility function. The Buddy package uses a hash table to represent the BDD discernibility function. The AND operation of each set of disjunctions into the Boolean discernibility function as well as the creation of the disjunctions is done in constant time [2] . The order of operation is $O(\frac{n^2 \cdot a}{\log(a)})$. As an outlook a method of creating a temporary reverse threaded BDDIS in order to derive the discernibility function more efficiently is explored.

6.2 Time behavior of reduct search in selected UCI data sets

Table 8 displays results for deriving the discernibility function along with the number of reducts attained and the time required. The results show that as the number of object gets large the time required gets unreasonably large.

Table 8: Discernibility function derivation from BDDIS; UCI Data Sets; Pentium 4 1.8GHz

	Objects				
Data set	number	discernible	Attr.	Num Reducts.	Time $[s]$
Iris	150	147	9	4	1
Nursery	12960	12960	9	1	371
Adult	32561	32537	15	2	2489
Cover Type	30000	30000	55	1969	5526
Connect-4 Opening	67557	67557	42	547	24990

It should be mentioned that the adult data set has missing values. Currently, there is no extra missing data treatment in our system, and we just use a code for missing as an extra value. Missing data handling within the BDDIS is one of the next steps after the feasibility of the representation is proven.

7 Outlook

7.1 Reduced BDDIS

Two methods can be taken to reduce the size of the BDDIS. One can reduce the BDDIS by applying the ROBDD rules described in Section 3 which were used to reduce Figure 1to Figure 2. The BDDIS would then be a fully reduced BDD and no longer simply quasi-reduced.

Alternatively, one can create the BDDIS using a coded information system in which every attribute value V_a of the original information system is coded to the values $0,\ldots,k_V-1$; here, each unique coded value represents a unique attribute value. An additional hash table is used to map the codes to the original values. Using the same codings for different entries will reduce the size of the BDDIS, whereas the cost of the hash table is negligible. Table 9 represents a coded decision system for Table 2. A coded decision system can be efficiently derived by creating a temporary BDDIS representing only one attribute. The top nodes will represent the object indiscernibility with respect to one attribute and derive a coding for the column.

From Figure 7, when using worst case data, the attribute top level representing all unique attribute values in the decision system, contains $(n \cdot a)$ nodes. A coded decision system reduces the number of unique attribute values

Table 9: A coded information system

Type	Price	Guarantee	Sound	Screen	d
1	0	0	0	0	0
2	1	1	1	1	1
3	1	2	0	2	1
4	2	2	0	3	0
5	2	3	0	3	0
6	0	2	0	3	1

to n. This can represent a huge reduction is the number of nodes required to represent the decision system when a is large. In a BDDISc created from a coded decision system the attribute top level becomes the leaf nodes level. Leaf nodes are represented logically as lo and hi keys within the leaf parent nodes. A BDDISc removes $(n \cdot a)$ nodes plus all nodes below which formed the attribute top level subtrees. The BDDISc also has the property of allowing further reductions to occur above the attribute top level as the ROBDD rules described in Section 3 are more frequently satisfied. For example: The number of nodes per level for the BDDIS derived by the random data test of 100.000 objects described within Table 7 is displayed in Figure 10. The BDDISc will remove levels 3, 4 and 5 resulting in the reduction of 1,506,819 nodes to 700,000 nodes. As well some number of further reductions will occur as reduction rules will be more likely to apply.

Table 10: BDD nodes per level: 100,000 Objects from random data

Level	Number Nodes
0 - Top	100,000
1	200,000
2	400,000
3	796,719
4	10,000
5	100

A rule is derived from this process in that the larger the number of characters that exist in a string the more nodes that are removed from the BDDISc as compared to the BDDIS. A further random test with 100,000 objects was performed using 16 characters per attribute with again 8 attributes resulting in a total of 3,096,790 nodes within the BDDIS. In this case the BDDISc representation removes levels 3, 4, 5, 6, resulting again in a 700,000 node structure.

It may be worthy of mention that worst-case data becomes near-best case data when using a fully reduced BDDIS derived from a coded decision system. If an decision system contained all unique attribute values in each column, for each row $i, 0 \leq i \leq n-1$ the coded information system would be composed completely of the integer value i.11.

Each row is converted into a subtree into the BDDIS and following the reduction rules, each root node i is also it's only leaf node i. Thus a series of nodes $0, \ldots n-1$ exists, regardless of the number of attributes. Of course a new worst case exists for the fully reduced BDDIS. This would be a coded decision system where each row of attribute values creates as many permutations from coded attribute values as possible which limit as much as possible the reduction rules from reducing the BDDIS. The best case represented by one root node is a coded information system where every attribute value is 0.

Table 11: Worst case coded information system

U	a	b	c	d
x_1	0	0	0	0
x_2	1	1	1	1
x_3	2	2	2	2
x_4	3	3	3	3

7.2 Temporary Reverse-Threaded BDDIS.

The time required to create the discernibility function can be greatly reduced by creating a *temporary reverse-threaded BDDIS* we denote as BDDIS*r*. The BDDIS*r* would contain the contain the same nodes as the BDDIS, but each node would contain two arrays; One for references to parents coming via hi paths, and one to parents coming via low paths within the BDDIS. The BDDIS*r* can be created using one complete traversal of the BDDIS.

The process to create a discernibility matrix then becomes as follows: For each attribute a of each object-top level root node i, traverse down a unique path of the BDDIS to an attribute-top level node. Then traverse up the BDDIS*r* following the reverse unique path to zero or more object-top level root nodes. This finds all objects that are indiscernible with i with respect to a.

Discernibility of i with every other object-top level root node is then implied with respect to a and stored as disjunctions within a temporary array of n length. A BDD representing a discernibility function for i with respect to all other objects is derived using this array.

$n \cdot a$ traversals must be performed down the BDDIS. Some number q traversals up the BDDIS*r*, that is greater or equal to the number of objects indiscernible with respect to a, must also be performed; Some small number of incorrect partial paths up the BDDIS*r* will occur. $O(naq\log(a))$ is derived which is much less that the current $O(n^2a\log(a))$ discussed in 6.1.

8 Concluding remarks

The quasi-reduced non reverse threaded implementation described in Section 4.1 is used in order to reduce complexity of the initial implementation in order to gain an understanding of how BDDs can be used to improve Rough Set Methods. Larger data sets then previously and even very large data with random structure can be treated as a basis for rough set data analysis with the chosen BDDIS representation, the initial results described have proven to be encouraging and will result in a continued focus on using BDD for rough set data analysis.

Acknowledgements

Co-operation for this paper was supported by EU COST Action 274 “Theory and Applications of Relational Structures as Knowledge Instruments” (TARSKI); http://www.tarski.org/. Alex Muir and Ivo Düntsch gratefully acknowledge support from the USRA and PDG programmes of the National Sciences and Engineering Research Council of Canada. Alex Muir was also supported by the President’s Fund, Brock University.

References

[1] Adel’son-Vel’skii, G. M. and Landis, Y. M. (1962). An algorithm for the organization of information. *Soviet Mathematics Doklady*, 3:1259–1262.

[2] Andersen, H. R. (1997). An introduction to binary decision diagrams. Lecture notes, Technical University of Denmark, Lyngby. `http://www.itu.dk/people/hra/bdd97-abstract.html`.

[3] Berghammer, R., Leoniuk, B., and Milanese, U. (2002). Implementation of relational algebra using binary decision diagrams. In de Swart, H., editor, *6th International Conference RelMiCS 2001 and 1st Workshop of COST Action 274 TARSKI, Oisterwijk, The Netherlands, October 16-21*, volume 2561 of *Lecture Notes in Computer Science*, pages 241–257, Berlin-Heidelberg-New York. Springer.

[4] Blake, C. and Merz, C. (1998). UCI repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[5] Lee, C. Y. (1959). Representation of switching circuits by binary-decision programs. *Bell System Technical Journal*, 38:985–999.

[6] Lind-Nielsen, J. (2003). Buddy – a binary decision diagram package – version 2.2. http://www.itu.dk/research/buddy/.

[7] Nguyen, H. S. and Nguyen, S. H. (1998a). Discretization methods in data mining. In Polkowski, L. and Skowron, A., editors, *Rough sets in knowledge discovery, Vol. 1*, pages 451–482. Physica–Verlag, Heidelberg.

[8] Nguyen, H. S. and Nguyen, S. H. (1998b). Pattern extraction from data. *Fundamenta Informaticae*, 34:129–144.

[9] Nguyen, S. and Nguyen, H. (1996). Some efficient algorithms for rough set methods. In *Proc. of the Conference of Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, pages 1451–1456, Granada.

[10] Ochi, H., Yasuoka, K., and Yajima, S. (1993). Breadth-first manipulation of very large binary-decision diagrams. In *Proceedings of the 1993 IEEE/ACM international conference on Computer-aided design*, pages 48–55. IEEE Computer Society Press.

[11] Pawlak, Z. (1982). Rough sets. *Internat. J. Comput. Inform. Sci.*, 11:341–356.

[12] Rauszer, C. (1991). Reducts in information systems. *Fundamenta Informaticae*, 15:1–12.

[13] R.E. Bryant (1986). Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8):677–691.

[14] S. Minato, N. Ishiura, and S. Yajima (1990). Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation. In *Proceedings of the 27th ACM/IEEE design automation conference*, pages 52–57, Los Alamitos, CA. ACM/IEEE, IEEE Society Press.

[15] S.B. Akers (1978). Binary Decision Diagrams. *IEEE Transactions on Computers*, C-27(6).

[16] Skowron, A. (1995). Extracting laws from decision tables - a rough set approach. *Computational Intelligence*, 11:371–388.

[17] Skowron, A. and Rauszer, C. (1992). The discernibility matrices and functions in information systems. In Słowiński, R., editor, *Intelligent decision support: Handbook of applications and advances of rough set theory*, volume 11 of *System Theory, Knowledge Engineering and Problem Solving*, pages 331–362. Kluwer, Dordrecht.