Development of fast algorithms for reduct computation

Vladímir Rodríguez Diez, José Francisco Martínez Trinidad

Reporte Técnico No. CCC-16-005
11 de abril de 2016

Luis Enrique Erro 1.
Sta. Ma. Tonantzintla,
72840, Puebla, México.

Development of fast algorithms for reduct computation

Vladímir Rodríguez Diez, José Francisco Martínez Trinidad

Computer Science Department
National Institute of Astrophysics, Optics and Electronics
Luis Enrique Erro # 1, Santa María Tonantzintla, Puebla, 72840, México

Abstract

Information systems in Rough Set Theory (RST) are tables of objects described by a set of attributes. This type of tables are widely used in different pattern recognition problems, particularly in supervised classification. RST reducts are minimal subsets of attributes preserving the discernibility capacity of the whole set of attributes. Reduct computation has an exponential complexity regarding the number of attributes in the information system. In the literature, several algorithms for reduct computation have been reported, but their high computational cost makes them infeasible in large problems. For this reason, in this research we will develop new fast algorithms in two directions, the computation of all reducts and the computation of globally shortest reducts. The proposed algorithms will be faster than state of the art algorithms, and hence the reduct computation will be viable for larger information systems than it is today. As part of this PhD research proposal, we present some preliminary results, which show that it is possible to develop faster algorithms for computing reducts.

Keywords: Rough Sets, Dimensionality Reduction, Reduct Computation.

Contents

1 Introduction **4**

2 Basic Concepts **4**

2.1 Information System 5

2.2 Positive Region 5

2.3 Reducts and Core 6

2.4 Discernibility Matrix and Discernibility Function 6

2.5 Binary Discernibility Matrix 7

2.6 Simplified Discernibility Matrix 7

3 Related Work **8**

3.1 Algorithms Finding a Single Reduct 8

3.2 Algorithms for all Reducts and all Typical Testors Computation 10

3.3 Parallel Accelerations 11

3.4 Concluding Remarks 12

4 Research Proposal **13**

4.1 Justification and Motivation 13

4.2 Research Questions 14

4.3 Research Objectives 15

4.4 Expected Contributions 15

4.5 Methodology 15

4.6 Schedule 17

5 Preliminary Results **17**

5.1 A Hardware Architecture for Filtering Typical Testors 19

5.1.1 Evaluation and Discussion 20

5.2 A Hardware Architecture based the CT_EXT Algorithm 20
5.2.1 CT_EXT algorithm 21
5.2.2 Evaluation and Discussion 23
5.3 A New Recursive Algorithm for Reduct Computation 25
5.3.1 Concepts for RSDM 25
5.3.2 The RSDM Algorithm 26
5.3.3 Evaluation and Discussion 26
5.4 A New Algorithm for Reduct Computation Based on the Gap Elimination and Contribution . 29
5.4.1 Concepts for GCSDM 29
5.4.2 The GCSDM Algorithm 31
5.4.3 Evaluation and Discussion 32

6 Conclusions **38**

1. Introduction

Rough set theory (RST), proposed by Z. Pawlak in 1982 (Pawlak, 1981a,b, 1982, 1991), is a relatively new mathematical theory to deal with imperfect knowledge, in particular with vague concepts. Into RST, information systems are tables of objects (rows) described by a set of attributes (columns). When data is collected or recorded, every single aspect (attribute) of the object under study is considered to have a complete representation and to ensure that no potentially useful information is lost. As a result, information systems are usually characterized by a large number of attributes, degrading the performance of machine learning tools (Parthaláin *et al.*, 2008). One of the main concepts in RST is the notion of reduct, which is a minimal subset of attributes preserving the discernibility capacity of the whole set of attributes (Pawlak, 1991). However, the main restriction in practical applications of RST is that computing all reducts of an information system has an exponential complexity (Skowron & Rauszer, 1992). Therefore an active research line is the development of fast algorithms for reduct computation.

Several attempts to speed up the reduct computation have been reported. Many of these algorithms are based on some heuristics. The main drawback of this approach is that these algorithms do not necessarily return the complete set of reducts of an information system, and sometimes they may obtain super-reducts (non minimal subsets). Another way to speed up reduct computation is parallelization (Strakowski & Rybiski, 2008). There are also interesting alternatives such as the use of a parallel version of genetic algorithms (Wroblewski, 1998) and the transformation of the reduct computation problem to the well known SAT problem (Jensen *et al.*, 2014).

Recently the RST reducts have been related to the typical testors (TT) from the logical combinatorial approach to pattern recognition (Lazo-Cortés *et al.*, 2015). Testor Theory was created by Cheguis & Yablonskii (1955) as a tool for analysis of problems connected with control and diagnosis of faults in circuits. Testor Theory can be used for feature selection as shown in (Ruiz-Shulcloper, 2008) and (Martínez & Guzmán, 2001). Algorithms for typical testors computation like (Ruiz-Shulcloper *et al.*, 1985), (Santiesteban & Pons, 2003), (Sanchez & Lazo, 2007) and (Lias-Rodríguez & Pons-Porrata, 2009), can be applied to reduct computation due to the similarity between these two concepts. Fast implementations of these algorithms; based on cumulative binary operations (Sánchez-Diaz *et al.*, 2010), genetic algorithms (Sanchez-Díaz *et al.*, 1999) and hardware architectures (Rojas *et al.*, 2012); have been developed to reduce the computation time. One strength of our research is that we will be evaluating, for the first time, these two families of algorithms in the same arena.

Throughout our research, we will conduct a comparative study between the most relevant algorithms for reduct computation. Although our main focus will be on algorithms for computing all reducts and globally shortest reducts, experiences from heuristics approaches will be considered as well. We will be exploring the relationship between algorithms' performance and characteristics of the information system. Based on this relationship, we will propose fast algorithms for reduct computation. Finally, the proposed algorithms will be redesigned and implemented in a hardware fashion in order to improve, even more, their efficiency.

2. Basic Concepts

RST is based on the assumption that every object in the universe of discourse is described, through a set of attributes, by some information associated to it. This information constitutes the basis for the classification of unseen objects. RST motto is *Let the data speak for themselves* (Tiwari, 2014).

From the RST point of view, two objects are indistinguishable (indiscernible) if they have an equivalent value for each attribute in their description. Indiscernibility relations arising this way constitute the mathematical foundations of RST. Some basic concepts of RST are presented bellow. Although we will be following the explanation in (Polkowski *et al.*, 2000), some modifications in the notation are introduced to provide clarity in the rest of the document.

2.1. Information System

The basic representation of data in RST is an *Information System* (IS). An IS is a table with rows representing objects while columns specify attributes or features. Formally, an IS is defined as a pair $IS = (U, A)$ where U is a finite non-empty set of objects $U = \{x_1, x_2, ..., x_n\}$, and A is a finite non-empty set of attributes (features, variables). Every attribute in A is a map: $a : U \rightarrow V_a$. The set V_a is called the *value set* of A. Attributes in A are further divided into condition attributes C and decision attributes D such that $A = C \cup D$ and $C \cap D = \emptyset$. Table 1 shows an example of an IS.

Table 1: An Information System.

	c_1	c_2	c_3	d
x_1	1	3	0	0
x_2	1	0	0	0
x_3	3	1	1	1
x_4	3	3	2	1
x_5	4	2	3	1
x_6	4	3	1	0
x_7	4	2	5	1

Decision attributes determine to which class an object belongs. In the IS of table 1, d is the decision attribute; this is a two-class system. *Condition attributes* do not absolutely determine the class but help to decide to which class an object belongs to. In supervised classification, condition attributes are the only information available for classifying new objects; while, decision attributes are only available for objects in the training set. An IS with decision and condition attributes is called a decision table. In table 1, c_1, c_2 and c_3 are condition attributes.

2.2. Positive Region

Decision attributes induce a partition of the universe U into equivalence classes (*decision classes*). Since we will be trying to associate a decision class to an object, based on the attributes belonging to $B \subseteq C$, we are interested in those $B - classes$ (classes induced by B) which correspond to classes induced by d. This idea leads to the notion of the *positive region of the decision*. The set $POS_B(d)$, called the *B-positive region of d*, is defined as the set of all objects in U such that all their indistinguishable objects (under the knowledge in B) belong to its same class induced by d.

Taking for example the IS in table 1, we can see that

$$\begin{aligned} POS_{\{c_1\}}(d) &= \{x_1, x_2, x_3, x_4\} \\ POS_{\{c_2\}}(d) &= \{x_2, x_3, x_5, x_7\} \\ POS_{\{c_1,c_2\}}(d) &= U \end{aligned}$$

2.3. Reducts and Core

Given an information system $IS = (U, A)$ with condition attribute set C and decision attribute set D such that $A = C \cup D$ and $C \cap D = \emptyset$. A subset $B \subseteq C$ is a *reduct* of IS relative to D if

1. $POS_B(D) = POS_C(D)$.
2. B is a minimal subset (with respect to inclusion) satisfying condition 1.

We call super-reduct to any subset $B \subseteq C$ satisfying condition 1 whether it satisfies condition 2 or not.

The intersection of all reducts of an IS is called the *core*.

2.4. Discernibility Matrix and Discernibility Function

The discernibility knowledge of an information system is commonly stored in a symmetric $|U| \times |U|$ matrix called the *discernibility matrix*. Each element m_{ij} in the discernibility matrix M_{IS} is defined as

$$m_{ij} = \begin{cases} \{c \in C : c(x_i) \neq c(x_j)\} & \text{for } D(x_i) \neq D(x_j) \\ \emptyset & \text{otherwise} \end{cases} \tag{1}$$

Here, $c(x_i)$ represents the value of the condition attribute c in the object x_i, and

$$D(x_i) \neq D(x_j) \Rightarrow \exists d \in D \mid d(x_i) \neq d(x_j)$$

where $d(x_i)$ represents the value of the decision attribute d in the object x_i.

Table 2 shows the discernibility matrix for the IS in table 1 as a lower triangular matrix ($\emptyset$'s are omitted for clarity).

Table 2: Discernibility Matrix Example.

$x \in U$	1	2	3	4	5	6	7
1							
2							
3	$\{c_1, c_2, c_3\}$	$\{c_1, c_2, c_3\}$					
4	$\{c_1, c_3\}$	$\{c_1, c_2, c_3\}$					
5	$\{c_1, c_2, c_3\}$	$\{c_1, c_2, c_3\}$					
6			$\{c_1, c_2\}$	$\{c_1, c_3\}$	$\{c_2, c_3\}$		
7	$\{c_1, c_2, c_3\}$	$\{c_1, c_2, c_3\}$				$\{c_2, c_3\}$	

Once the discernibility matrix M_{IS} is found, we can define the *discernibility function* f_{IS}. This is a Boolean function of n Boolean variables $c_1^*, c_2^*, ..., c_n^*$, representing the presence of the corresponding attribute (True) or its absence (False) in M_{IS}. Here, the disjunction ($\vee$) and conjunction ($\wedge$) operators have their common meaning. Their evaluation over a set of Boolean variables $X = \{x_1^*, x_2^*, ..., x_n^*\}$ should be denoted as $\wedge X = x_1^* \wedge x_2^* \wedge ... \wedge x_n^*$.

$$f_{IS}(c_1^*, c_2^*, ..., c_n^*) = \wedge\{\vee c_{ij}^* : 1 \leq j \leq i \leq |U|, m_{ij} \neq \emptyset\} \tag{2}$$

where $c^*_{ij} = \{c^* : c \in m_{ij}\}$. Only the lower triangular matrix from M_{IS} is taken into consideration since M_{IS} is symmetric. An equivalence between the prime implicants of f_{IS} and all reducts of IS has been found and reported in (Pawlak & Skowron, 2007).

The discernibility function for the discernibility matrix in table 2, after simplifying by deleting repeated clauses, is

$$f_{IS}(c^*_1, c^*_2, c^*_3) = (c^*_1 \vee c^*_2 \vee c^*_3) \wedge (c^*_1 \vee c^*_2) \wedge (c^*_1 \vee c^*_3) \wedge (c^*_2 \vee c^*_3)$$

2.5. Binary Discernibility Matrix

The *Binary Discernibility Matrix* is a binary table representing the discernibility sets between pairs of objects. This is another representation of the information in M_{IS}. In the binary discernibility matrix, columns are single condition attributes and rows represent pairs of objects belonging to different classes. The discernibility element $m(i, j, c)$ for two objects x_i and x_j and a single condition attribute $c \in C$ is given in a binary representation, such that:

$$m(i, j, c) = \begin{cases} 1 & \text{for } c(x_i) \neq c(x_j), D(x_i) \neq D(x_j) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Table 3 shows the binary discernibility matrix for the information system of Table 1.

Table 3: Binary Discernibility Matrix Example.

	c_1	c_2	c_3
x_1, x_3	1	1	1
x_1, x_4	1	0	1
x_1, x_5	1	1	1
x_1, x_7	1	1	1
x_2, x_3	1	1	1
x_2, x_4	1	1	1
x_2, x_5	1	1	1
x_2, x_7	1	1	1
x_3, x_6	1	1	0
x_4, x_6	1	0	1
x_5, x_6	0	1	1
x_6, x_7	0	1	1

2.6. Simplified Discernibility Matrix

The *Simplified Discernibility Matrix* is a reduced version of the discernibility matrix after eliminating supersets and repeated rows in M_{IS}. This new discernibility matrix has the same reducts as the original one (Yao & Zhao, 2009). An equivalent concept exists in Testor Theory, called *Basic Matrix*. The basic matrix was proven to have the same TT as the original information system (Lazo-Cortés *et al.*, 2001). Table 4 shows the basic matrix for the discernibility matrix from Table 3.

Table 4: Simplified Discernibility Matrix.

c_1	c_2	c_3
1	0	1
1	1	0
0	1	1

3. Related Work

In this section, we will be first discussing heuristic algorithms for reduct computation. Some of these algorithms are capable of finding several reducts and others are intended to obtain a single *shortest* reduct. Then, two kinds of algorithms for computing all reducts will be exposed: those from RST and those from Testor Theory. Finally, we will make a review of parallel accelerations reported in the literature.

In Figure 1, we propose a taxonomy of the reported algorithms for computing a single reduct and, in Figure 2, a taxonomy of the reported algorithms for computing all reducts. This classification corresponds to the sequence that we will be following throughout our review of the state of the art.

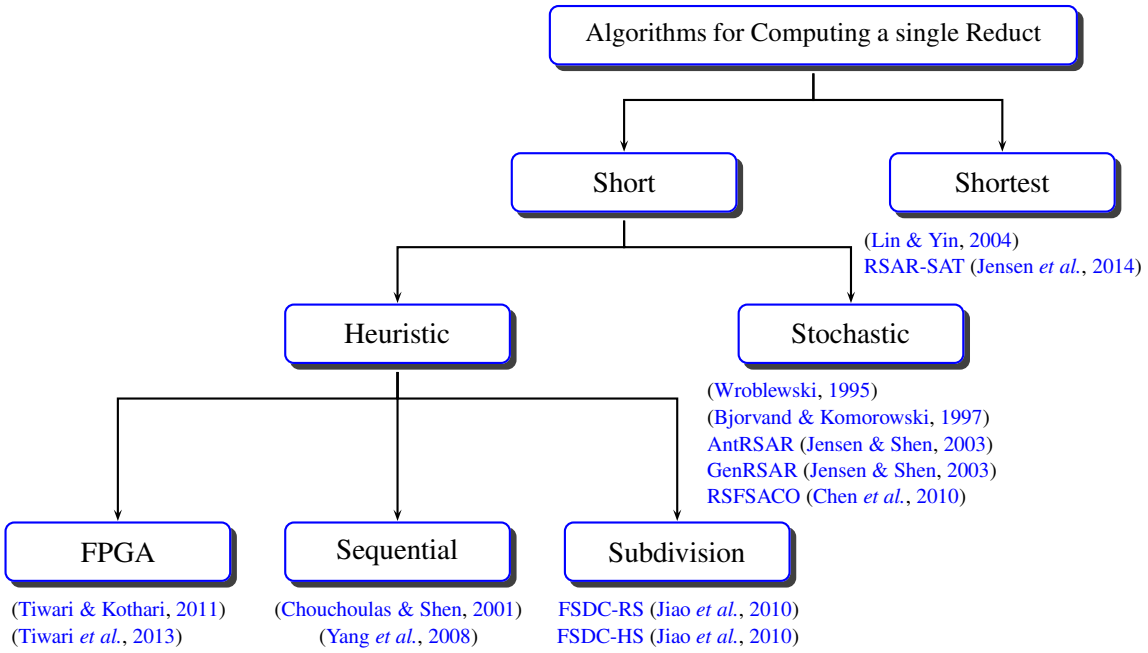

Figure 1: Taxonomy of algorithms for computing a single reduct.

3.1. Algorithms Finding a Single Reduct

The algorithm presented in (Chouchoulas & Shen, 2001) QUICKREDUCT starts with an empty set of attributes and adds, one at a time, the attribute having the highest significance. This greedy algorithm evaluates the significance of an attribute as the number of objects added to the positive region after its inclusion. A similar approach is the Johnson Reducer (Johnson, 1973), first introduced in RST by Ø hrn (2000). This simple greedy algorithm begins with an empty set of attributes evaluating each conditional attribute in the discernibility function according to a heuristic measure. In the simplest case, those attributes with highest appearance frequency within the logical clauses of the discernibility function, are considered to be more

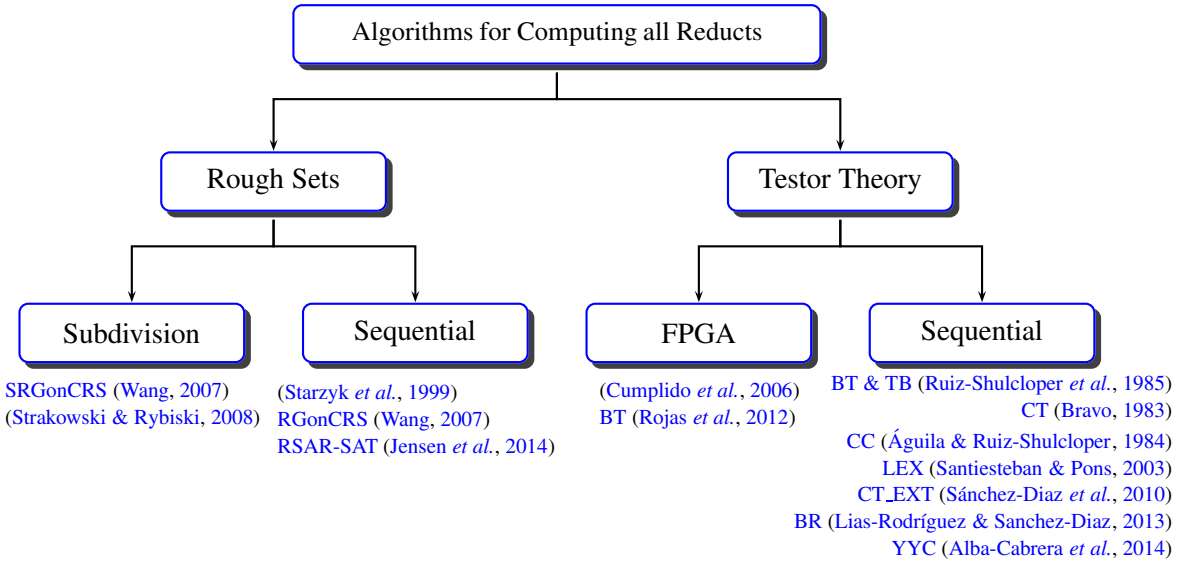

Figure 2: Taxonomy of algorithms for computing all reducts.

relevant. The algorithms in (Nguyen & Skowron, 1997) and (Wang & Wang, 2001) use alternative heuristic functions for guiding the search; while (Yang *et al.*, 2008) use the discernibility matrix instead of the discernibility function. The algorithm presented in (Zhong *et al.*, 2001) starts from the core (since it must be contained in every reduct) and follows a similar procedure adding selected attributes. This optimization may be impractical for large datasets (Jensen *et al.*, 2014) since the core must be computed a priori.

The method presented in (Jiao *et al.*, 2010) improves the efficiency of computing reducts by means of subdivisions of the dataset. The original dataset is broken down into a master-table and several sub-tables both, simpler and more manageable. Two algorithms are proposed (FSDC-RS and FSDC-HS) using different decomposition strategies. Results are then joined together in order to find reducts of the original dataset.

Special attention deserves the approaches using genetic algorithms to discover locally shortest reducts. Although these algorithms do not guarantee finding globally shortest reducts, many reducts may be found in a determined time. A good point in this approach is the use of a fitness function to guide the search down to a set of reducts with the desired properties. The algorithm reported in (Wroblewski, 1995) encodes candidates as bit strings with a positional representation of attributes in candidate sets. The fitness function depends on the number of attributes in the subset, penalizing strings with a large number of attributes. The second optimization parameter is the number of objects that can be distinguished by the given candidate. The reduct should discern as many objects as possible. Jensen & Shen (2003) also introduced a simple algorithm (GenRSAR), which uses a genetic search strategy in order to find reducts.

Other bio-inspired approaches to reduct computation include Ant Colony Optimization (Jensen & Shen, 2003) (AntRSAR) and (Chen *et al.*, 2010) (RSFSACO); and Particle Swarm Optimization (Wang *et al.*, 2007).

In (Lin & Yin, 2004), a heuristic is followed to find a short reduct. This first reduct is used to limit the search space, in order to only consider those attribute combinations with lower cardinality. The main drawback of this algorithm is that the second step searches for reducts by checking all possible s-subtables of the whole database. An s-subtable means a subtable whose conditional attribute set have size s. In other words, it is a decision table with a conditional attribute subset of size s plus the decision attributes of the

original table. This final process uses no pruning strategy and explores all combinatorial possibilities of attribute combinations, which is unfeasible in most cases.

Although originally intended for computing a single minimal reduct, the algorithm proposed in (Jensen *et al.*, 2014) (RSAR-SAT) may be modified in order to obtain all reducts in an Information System. The method introduced in this work reduces the problem of finding a reduct from the discernibility function to the SAT problem (Davis *et al.*, 1962). The boolean function generated in this way is always satisfied since the complete set of attributes is a trivial solution.

3.2. Algorithms for all Reducts and all Typical Testors Computation

One of the first algorithms designed to overcome the exponential complexity (regarding the number of features) of the problem of finding all TT, was proposed by Ruiz-Shulcloper *et al.* (1985). This algorithm, called BT, codifies a subset of features as a binary word with as many bits as features in the dataset. A 0 represents the absence of the corresponding feature in the current subset while a 1 represents its inclusion. This way, candidates subsets are evaluated in the natural order induced by binary numbers. The pruning process in the search space is based on the minimal condition of TT and a convenient sorting of the basic matrix associated to the dataset. Finally, testors found by BT algorithm must be filtered in order to remove any non-TT. In (Ruiz-Shulcloper, J., Alba-Cabrera, E., Lazo-Cortés, 1995) a new algorithm (REC) is presented. The main drawback of REC is that it works directly over the dataset (instead of using the basic matrix), handling a huge amount of superfluous information. Ayaquica (1997) presented the algorithm CER directed to solve this problem by using a different traversing order.

Then, Santiesteban & Pons (2003) proposed a new algorithm called LEX. The main ideas behind LEX are a new traversing order of candidates (which resembles the lexicographical order in which string characters are compared) and the concept of gap. In LEX, the typical condition is verified first and only for those potentially TT, the testor condition is checked. The concept of gap allows to avoid the evaluation of any subset of a candidate, given that it is a TT (or a not testor) which includes the last feature in the dataset.

Sanchez & Lazo (2007) proposed the CT_EXT algorithm for computing all TT. Following a traversing order similar to that in LEX, this algorithm searches for testors without verifying the typical condition. This way, a larger number of candidates are evaluated, in comparison to LEX; but the cost of each evaluation is lower. Results from experiments show that CT_EXT is faster than the previous existing algorithms for most datasets. Then, Lias-Rodríguez & Pons-Porrata (2009) presented the BR algorithm, a Recursive algorithm based on Binary operations. BR is very similar to LEX in its bones but its recursive nature encloses a great improvement. Given a candidate subset, the remaining features are tested a priori and those being rejected are excluded from subsequent evaluations. Sánchez-Diaz *et al.* (2010) presented a cumulative procedure for the CT_EXT algorithm. This fast-CT_EXT implementation drastically reduces the runtime for most datasets at no extra cost. In (Lias-Rodríguez & Sanchez-Diaz, 2013) the gap elimination and column reduction are added to BR. This fast-BR algorithm is, no doubt the one evaluating the minimum number of candidates in the state of the art. The main drawback of fast-BR and BR is, as in LEX, the high cost of evaluating the typical condition for each candidate.

Recently, a new internal scale typical testor–finding algorithm (YYC) was proposed by Alba-Cabrera *et al.* (2014). Although they claim that this algorithm verify less candidates than previous alternatives, two weak points should be addressed. First, BR is not included in their comparisons; and second, the evaluation cost for a candidate in YYC is high compared to that of previous algorithms. YYC verifications involve calculations of the Hamming weight.

A method for the computation of all reducts in an Information System is proposed in (Starzyk *et al.*, 1999, 2000). This is a divide and conquer approach. On each step, the absorption laws are applied over the incoming discernibility matrix to obtain a basic matrix. Then, the strong equivalent attributes are compressed (which is a local reduction of columns). The most discerning attribute is selected (in the same way as Johnson's reducer does) and the problem is divided into two sub-problems:

- Finding reducts containing the selected attribute. Thus a recursive function is called with a new basic matrix, having only those rows where the selected attribute does not appear.
- Finding reducts that do not contain the selected attribute. Thus a recursive function is called with a new discernibility matrix, removing the column corresponding to the selected attribute.

The base case in the recursion is reached when each attribute in the incoming discernibility matrix appears in a single clause. In this way, a set of super-reducts is obtained and supersets must be removed in order to obtain the final reduct set. Notice that this algorithm is oriented to the binary discernibility function, then terms such as discernibility and basic matrix, rows and columns are not used in the paper. The algorithm is presented in an iterative fashion and its recursive nature is not clearly expressed.

Wang (2007) proposed a new algorithm for computing all reducts RGonCRS. Even though this algorithm was developed independently and reported two years before to the one reported in (Lias-Rodríguez & Pons-Porrata, 2009), it is very similar to BR. Notice that this is a rough set approach to the problem and the nomenclature is totally different from that of BR. Essentially, every proposition supporting the pruning process in (Wang, 2007) have an equivalent proposition in (Lias-Rodríguez & Pons-Porrata, 2009). The main differences of RGonCRS with BR are:

- It works directly over the dataset instead of the basic matrix.
- It starts searching the core and looks for reducts as supersets of the core.
- A recursive implementation, instead of the iterative solution used in BR, is proposed.
- During the algorithm execution, contributing attributes are sorted as in the Johson reducer.
- A second algorithm SRGonCRS is proposed for subdividing the dataset and the reducts are incrementally found.

Different variants (DT, DISC FUNCTION and CANDIDATE REDUCTS) for decomposition of a reduct computation problem are discussed and proposed in (Strakowski & Rybiski, 2008).

3.3. Parallel Accelerations

A parallel acceleration of the algorithm presented in (Yang *et al.*, 2008), for reduct generation from a binary discernibility matrix, was developed in (Tiwari & Kothari, 2011; Tiwari *et al.*, 2012). This FPGA implementation computes a single reduct. A real application for object identification into an intelligent robot is presented. In (Tiwari *et al.*, 2013) a *quick reduct* algorithm, similar to that presented in (Chouchoulas & Shen, 2001), is proposed and implemented in a hardware fashion. A recent work from these authors (Tiwari, 2014), shows a thorough survey of FPGA applications in rough set reduct computation.

In (Wroblewski, 1998), a parallel variant of the algorithm proposed in (Wroblewski, 1995) is presented. Developments in parallel implementations of genetic algorithms are exploited to provide a speedup for the problem of finding reducts.

In (Grze, 2013; Kopczynski *et al.*, 2014), an FPGA application for a single reduct computation is presented. Although authors claim that a huge acceleration is achieved, some weak points have to be mentioned. Experiments presented in (Kopczynski *et al.*, 2014) to validate their results are performed over a small dataset which in our experience does not imply its applicability to larger cases where such acceleration is needed. On the other hand, runtime estimations for the FPGA component executions are made by means of an oscilloscope without taking into account communication overhead.

From the Testor Theory, several attempts to overcome the complexity problem, by means of FPGA implementations of algorithms, have been done. In a first work (Cumplido *et al.*, 2006), an FPGA-based brute force approach for computing testors was proposed. This first approach did not take advantage of dataset characteristics to reduce the number of candidates to be tested; thus all 2^n combinations of n features have to be tested. Then, in (Rojas & Cumplido, 2007) a hardware architecture of the BT algorithm for computing typical testors was implemented. This algorithm uses a candidate pruning process for avoiding many unnecessary candidate evaluations, reducing the number of verifications of the typical testor condition. These two previous works compute a set of testors on the FPGA device whilst the typical condition was evaluated afterwards by the software component in the hosting PC. Thus, in (Rojas *et al.*, 2012) a hardware-software platform for computing typical testors that implements the BT algorithm, similar to (Rojas & Cumplido, 2007), was proposed; but it also included a new module that eliminates most of the non typical testors before transferring them to a host software application for final filtering.

3.4. Concluding Remarks

From our literature review, we found that algorithms for finding typical testors can be used for computing reducts and vice-versa (Lazo-Cortés *et al.*, 2015). We also noticed that the development of algorithms from Testor Theory is biased to finding all the typical testors of an information system. Algorithms from Rough Set Theory, on the other hand, are mainly divided into three categories:

- Algorithms for computing a pseudo-optimal reduct according to a criterion (which is, most of the time, the cardinality of the obtained reduct).
- Algorithms for computing one shortest reduct.
- Algorithms for computing all reducts.

The most proliferative research area in Rough Set Theory is the development of algorithms for computing a pseudo-optimal reduct.

We found two algorithms for finding all reducts (Starzyk *et al.*, 2000; Wang, 2007) and two algorithms for finding shortest reducts (Lin & Yin, 2004; Jensen *et al.*, 2014) from Rough Set Theory. These algorithms have several disadvantages since they do not work over the simplified discernibility matrix and use complex data representations. Most of the ideas for pruning the search space in these algorithms can be found in Testor Theory as well, however, we found interesting ideas, which are unexplored in Testor Theory. Hardware accelerations of algorithms for computing reducts are focused on computing a single pseudo-optimal reduct (Tiwari & Kothari, 2011; Tiwari *et al.*, 2012, 2013; Grze, 2013; Kopczynski *et al.*,

2014; Tiwari, 2014). Since this problem is not an exponentially complex task, we found these hardware platforms less useful.

Algorithms for finding typical testors operate over the simplified discernibility matrix (basic matrix). Notice that computing the basic matrix from the original dataset has quadratic complexity regarding the number of objects (rows) in the dataset, while computing all the typical testors has exponential complexity regarding the number of attributes (columns). In most computationally expensive datasets, it is better to work over the simplified discernibility matrix. Properties used by these algorithms are supported by boolean operations and bit manipulations, which lead to faster implementations. We identified fast-CT_EXT (Sánchez-Diaz *et al.*, 2010) and fast-BR (Lias-Rodríguez & Sanchez-Diaz, 2013) as the fastest algorithms reported in the literature for finding typical testors. Hardware accelerations reported into Testor Theory (Cumplido *et al.*, 2006; Rojas & Cumplido, 2007; Rojas *et al.*, 2012) are focused on computing all the typical testors, and their main disadvantage is that the size of the basic matrix to be solved is limited by the hardware resources available in the FPGA device.

In the search for algorithms to overcome the exponential complexity of the problem of computing all reducts or shortest reducts, the last word has not been said. In our preliminary results, we propose two new algorithms to show that improvements can be obtained by exploring new pruning rules.

4. Research Proposal

In this section we present the justification and motivation, the research questions, the objectives and the expected contributions of this PhD research proposal. We also include a detailed methodology and the schedule for reaching our objectives.

4.1. Justification and Motivation

RST can be used to reduce the number of attributes in a dataset without relevant information loss. Therefore, there has been a lot of research on finding reducts, particularly, shortest reducts (Jensen *et al.*, 2014). Zheng *et al.* (2014) highlighted the relevance of feature selection through rough set reducts and illustrated the current research activity on this topic. Recently, Jiang & Yu (2015) said that attribute reduction is one of the most important tasks in rough sets and, as a consequence, many strategies for finding reducts have been investigated. In general, we find consensus in the literature about both, the relevance and the actuality of research on rough set reducts.

Heuristic methods such as (Chouchoulas & Shen, 2001; Jensen & Shen, 2004; Zhong *et al.*, 2001) are fast alternatives for finding reducts but they do not guarantee to find a shortest reduct. Stochastic approaches such as (Wroblewski, 1995; Jensen & Shen, 2003; Chen *et al.*, 2010; Wang *et al.*, 2007) still do not guarantee finding a shortest reduct, as we have seen in Section 3. Techniques for finding all reducts (Starzyk *et al.*, 1999; Wang, 2007) can, of course, find the shortest reducts but with a high computational cost.

The motivation of this research is the development of algorithms for computing all reducts and globally shortest reducts in information systems. Both problems have exponential complexity, which makes every attempt of reducing execution time a challenging task. Our proposals for computing all reducts and globally shortest reducts must be competitive with the state of the art algorithms in the general case and faster in some determined cases. The main arena for comparison will be a large set of synthetic, randomly generated

datasets and benchmarking datasets from (Bache & Lichman, 2013). Practical applications of reducts in supervised classification or feature selection, are beyond our goals.

The products of this research will impact feature selection methods specially in large datasets. Nowadays, data is automatically collected, thus the generation of huge databases appears in almost every field. The current growth of the size of data, and the number of existing databases, is another justification for our research on fast algorithms for dimensionality reduction without losing discriminative power.

4.2. Research Questions

Throughout our state of the art review, we noticed that there is not a fastest algorithm for finding reducts on any dataset. Algorithms reported in the literature use different strategies for traversing and pruning the whole search space. Consequently, some strategies are better suited for some datasets while they are time consuming for some others. This leads us to our first research question:

> *Is there a relationship between some properties of the basic matrix and the runtime of traversing strategies for finding reducts in information systems?*

We will be considering those properties that can be extracted from the basic matrix by traversing their cells just once. Lets take for instance, the minimum and maximum number of attributes in a cell, the core or the mean number of attributes per cell. Other properties that require more complex operations to be extracted such as the number of reducts, the cardinality of the shortest and largest reducts, etc; will not be considered.

From this research question we formulate the following hypothesis:

> *There is a relationship between the properties of the basic matrix and the runtime of traversing strategies for finding reducts in information systems*

The more sophisticated a traversing strategy is, the less number of candidate attributes sets are evaluated to verify whether they are reducts or not. Unfortunately, a more sophisticated traversing strategy has usually a higher computational cost. This trade-off between the number of evaluated candidates and their evaluation cost, leads us to our second research question:

> *Can the runtime for computing reducts in information systems be reduced by dynamically changing the traversing strategy?*

By dynamically changing we mean the change of the traversing strategy during the reduct computation. From this research question we formulate the following hypothesis:

> *The runtime for computing reducts in information systems can be reduced by dynamically changing the traversing strategy*

Based on these two scientific questions we can formulate the main hypothesis for our research:

Using some properties of the basic matrix, and dynamically changing the traversing strategy, we can design new algorithms for computing reducts in information systems; which are faster than the state of the art alternatives in a certain kind of datasets

4.3. Research Objectives

The main objective in our research is the *development of new algorithms for computing reducts in information systems; which will be comparable to state of the art algorithms in most datasets, and faster in some specific kinds of datasets*.

These algorithms will use some properties of the basic matrix to conveniently select the traversing strategy for the search space. We will explore two variants of this problem, the problem of computing all reducts and the problem of computing globally shortest reducts. The problem of finding shortest reducts has also exponential complexity (Lin & Yin, 2004) but different pruning rules could be used.

Our specific objectives are:

1. Finding a relationship between some properties of the basic matrix and the fastest traversing strategy for computing all reducts.
2. Developing a new algorithm for computing all reducts.
3. Finding a relationship between some properties of the basic matrix and the fastest traversing strategy for computing globally shortest reducts.
4. Developing a new algorithm for computing globally shortest reducts.

4.4. Expected Contributions

- A new algorithm for computing all reducts, which will be comparable to state of the art algorithms in most datasets, and faster in some specific kinds of datasets.
- A new algorithm for computing shortest reducts, which will be comparable to state of the art algorithms in most datasets, and faster in some specific kind of datasets.
- A meta-characterization of algorithms' efficiency in relation to some properties of the basic matrix associated to a dataset.
- Software and hardware implementations for computing both, all reducts and shortest reducts.

4.5. Methodology

1. Critical study of the most recent and fastest algorithms for computing reducts.
2. Finding a relationship between some properties of the basic matrix and the fastest traversing strategy for computing all reducts.
    - Generating a set of random datasets, systematically covering the space of possible combinations of properties of basic matrices (factorial design). These properties are, for instance, the density of ones in the basic matrix, the minimum and maximum number of ones in a row, the standard deviation of the density of ones in rows and columns, etc.
    - Implementing the main traversing strategies reported in the literature for the computation of all reducts.

- Generating an information system with the properties of each basic matrix, using the fastest strategy as decision attribute.
- Extracting a relevant subset of properties for determining a priori the appropriate traversing strategy for a given dataset; and the rules governing this relation. We will use Rough Set Theory for this purpose.
- Proposing a new algorithm for computing all reducts using the rules found in the previous step.
- Evaluating the proposed algorithm over synthetic and benchmarking datasets (Bache & Lichman, 2013).

3. Finding a relationship between the traversed space and the expected cost of traversing strategies.
   - Implementing the main traversing strategies reported in the literature for the computation of all reducts in such a way that we can collect statistics for every execution stage.
   - Making a statistical description of strategies' runtime cost over synthetic and benchmarking datasets.
   - Finding a correlation between the traversed space and the expected cost of traversing strategies.
4. Developing a new algorithm for computing all reducts in information systems.
   - Proposing a new algorithm for computing all reducts based on the relations found in steps 2 and 3.
   - Evaluating the proposed algorithm over synthetic and benchmarking datasets.
5. Finding a relationship between some properties of the basic matrix and the fastest traversing strategy for computing globally shortest reducts.
   - Implementing the main traversing strategies reported in the literature for the computation of minimal length reducts.
   - Generating an information system with the properties of each basic matrix, using the fastest strategy as decision attribute.
   - Extracting the relevant subset of properties for determining a priori the appropriate traversing strategy for a given dataset; and the rules governing this relation.
   - Proposing a new algorithm for computing shrotest reducts using the rules found in the previous step.
   - Evaluating the proposed algorithm over synthetic and benchmarking datasets.
6. Finding a relationship between the traversed space and the expected cost of traversing strategies.
   - Implementing the main traversing strategies reported for the computation of globally shortest reducts in such a way that we can collect statistics for every execution stage.
   - Making a statistical description of strategies' runtime cost over synthetic and benchmarking.
   - Finding a correlation between the traversed space and the expected cost of traversing strategies.
7. Developing a new algorithm for computing shortest reducts in information systems.
   - Proposing a new algorithm for computing shortest reducts based on the relations found in steps 5 and 6.
   - Evaluating the proposed algorithm over synthetic and benchmarking datasets.
8. Finally, the proposed algorithms will be redesigned and implemented in a hardware fashion in order to evaluate the speed up that can be obtained.

4.6. Schedule

Table 5 shows the schedule of the main tasks that will be carried out throughout this research.

Table 5: Research schedule (quarterly[1]).

Task	Quarters											
	2014	2015			2016			2017			2018	
	1	2	3	4	5	6	7	8	9	10	11	12
Literature review												
Writing the research proposal												
Critical study of algorithms for computing all reducts in information systems												
Implementation of algorithms for computing all reducts in information systems												
Development of a new algorithm for computing all reducts in information systems												
Critical study of algorithms for computing shortest reducts in information systems												
Implementation of algorithms for computing shortest reducts in information systems												
Development of a new algorithm for computing shortest reducts in information systems												
Critical study of hardware accelerations of algorithms for computing reducts in information systems												
Designing and implementing in hardware the proposed algorithms												
Experimental set-up												
Experiments run												
Writing papers												
Writing the PhD dissertation												
Submit final draft of the PhD dissertation to supervisors												
Submit final version of the PhD dissertation to the PhD committee												

5. Preliminary Results

In this section, we expose the preliminary results of this PhD research. Our first studies were directed to algorithms for computing typical testors. Throughout our literature review, we found that there are two kinds of algorithms for computing typical testors: *internal scale* algorithms, such as CT (Bravo, 1983), CC

[1]Quarters are: [January-April], [May-August] and [September-December]. Schedule starts in September 2014, according to the admission of the student in the PhD program.

(Águila & Ruiz-Shulcloper, 1984) and YYC (Alba-Cabrera *et al.*, 2014); and *external scale* algorithms, such as BT and TB (Ruiz-Shulcloper *et al.*, 1985), LEX (Santiesteban & Pons, 2003), CT_EXT (Sanchez & Lazo, 2007) and BR (Lias-Rodríguez & Pons-Porrata, 2009). The former analyze the matrix to find out some conditions that guarantee that a subset of attributes is a typical testor. The latter search typical testors over the whole power set of attributes, avoiding unnecessary evaluations by pruning some attribute subsets. Internal scale algorithms usually evaluate less candidates than external scale algorithms but each candidate evaluation has a higher computational cost. Therefore, the search of fast algorithms for computing typical testors has been biased to external scale algorithms (Alba-Cabrera *et al.*, 2014).

Rojas & Cumplido (2007) presented a hardware implementation of the external scale algorithm BT, and a comparative study including a brute force hardware approach and software implementations of BT and CT (Bravo, 1983) algorithms. This hardware implementation of BT was faster than other existing algorithms for finding all typical testors (although the hardware architecture was only capable of finding testors; which should be filtered afterwards by a software component in a hosting PC, to remove the non typical testors). In (Rojas *et al.*, 2012) a new component was introduced to this platform in order to filter most of the non typical testors in the FPGA device before transferring them to the hosting PC. We identified the final filtering stage, in the hosting PC, as the main drawback of both platforms (Rojas & Cumplido, 2007; Rojas *et al.*, 2012). Thus, in the subsection 5.1 we present a new architecture for computing typical testors, which is capable of finding all the typical testors without needing a final filtering stage. The runtime performance of our proposal is shown using some benchmarking datasets (Bache & Lichman, 2013), as we propose in our methodology. This result has been reported in (Rodríguez *et al.*, 2014).

As a next step in our preliminary work, we included in our study the CT_EXT algorithm (Sanchez & Lazo, 2007), which uses one of the fastest strategies for traversing the search space. CT_EXT evaluates less candidates than BT (used in our previous preliminary result) in most cases, without including more complicated operations. As a result, CT_EXT is faster than BT in most datasets. Following the idea of Rojas & Cumplido (2007), we introduce a hardware platform inspired in CT_EXT. Then, in the subsection 5.2 we describe the new platform for computing typical testors, based on the CT_EXT algorithm. This new hardware implementation outperforms existing implementations of algorithms for computing typical testors. A runtime comparison, including algorithms reported in (Rodríguez *et al.*, 2014; Sanchez & Lazo, 2007), is carried out over several synthetically generated basic matrices, as suggested in the second step of our methodology. This result was accepted in the journal of *Expert Systems with Applications*[2].

As part of the preliminary results of this PhD proposal, we propose and evaluate two new algorithms for finding all the reducts from an information system. Throughout the critical study proposed in the first step of our methodology, we identified some properties of reduct computation that we used to develop the algorithms presented in subsections 5.3 and 5.4. These results, partially cover the second point of our methodology, and some elements of these algorithms will be used for the algorithm to be proposed in the fourth point.

Additionally, in subsections 5.3 and 5.4, we also implemented several traversing strategies and evaluated them over synthetically generated matrices. These experiments aim to look for rules that would allow selecting the fastest strategy, based on properties of the simplified discernibility matrix.

[2]http://www.sciencedirect.com/science/article/pii/S0957417415004972

5.1. A Hardware Architecture for Filtering Typical Testors

The first step of our methodology includes the critical study of the most recent and fastest algorithms for computing reducts. In this direction, we started studying the hardware implementations of algorithms for computing typical testors. Rojas *et al.* (2012) presented a hardware-software platform for computing typical testors, based on the BT algorithm, that included a module for eliminating most of the non typical testors before transferring them to a host software application for final filtering. The main disadvantages of this approach are the huge amount of data that must be transferred to the PC and the extra cost of the final filtering stage in the software component. For this reason, we developed a new hardware module for eliminating all non typical testors on the hardware component; reducing the amount of data that must be transferred to the PC and eliminating the final filtering stage. This new module can be used together with any algorithm for computing typical testors implemented on an FPGA device, as we will show in the subsection 5.2.

In the hardware platform (Rojas *et al.*, 2012), a feature subset is handled as an n-tuple, using a positional representation for the n attributes of a basic matrix (BM). Given a subset T, its n-tuple representation has a 1 in the corresponding position j for each $c_j \in T$ and 0 anywhere else. The information of BM is hold in the BM hardware module. This module handles the process of deciding whether an n-tuple is a testor of BM, by comparing the candidate against each one of the BM's rows.

In our proposed architecture, an *N to N Decoder* is introduced into each row of the basic matrix. This new component receives as input the result of the AND operation between the current candidate and the corresponding BM row. The output from the *N to N Decoder* repeats the input when there is only one bit set to 1, and returns the null n-tuple $(0, ..., 0)$ otherwise. For those rows with only one bit having a 1 after ANDed with the candidate, the attribute in the position of that bit is indispensable if the candidate is a testor.

Two operations are added to the BM module in order to verify the typical condition of testors. First, a bitwise OR operation is performed among the output of the *N to N Decoder* of every row. The result of this operation has a 1 in those positions corresponding to each indispensable attribute in the current candidate. This value is then compared to the current candidate, and if this comparison holds equality and the current candidate is a testor, then the current candidate is a typical testor.

Lets us take for example the basic matrix shown in Table 4. We are going to illustrate the operation of the proposed architecture using two typical testor candidates for this basic matrix. First, we will evaluate the candidate $\{c_1, c_2\}$ which is a typical testor. Secondly, the candidate $\{c_1, c_2, c_3\}$ will be evaluated. This last attribute set is a superset of $\{c_1, c_2\}$ and thus, it is not a typical testor.

Table 6: An example of typical testor

Cand. $\{c_1, c_2\}$			Decoder output		
c_1	c_2	c_3	c_1	c_2	c_3
1	0	0	1	0	0
1	1	0	0	0	0
0	1	0	0	1	0
Candidate $=$			1	1	0

Table 7: An example of a non typical testor

Cand. $\{c_1, c_2, c_3\}$			Decoder output		
c_1	c_2	c_3	c_1	c_2	c_3
1	0	1	0	0	0
1	1	0	0	0	0
0	1	1	0	0	0
Candidate $\neq$			0	0	0

Left rows of Table 6 and Table 7 show the result of the AND operation between each row of BM and the corresponding candidate. Rows in the right side show the decoder output taking as input its corresponding left row. In the last row, the result of an OR operation over all above n-tuples is shown. According to our previous explanation, the candidate $\{c_1, c_2\}$ is a typical testor given that the result of the OR operation is equal to the candidate itself; while the candidate $\{c_1, c_2, c_3\}$ is not.

5.1.1. Evaluation and Discussion

This proposed architecture sends only typical testors from the FPGA device to the host PC. This modification eliminates the final filtering stage in the software component of the hardware-software platform (Rojas *et al.*, 2012). On the other hand, the final filtering stage in the software component needed in the original platform (Rojas *et al.*, 2012) checks every testor received from the FPGA. Testors which are a superset of any other testor are eliminated because they do not satisfy the typical condition. We can establish a lower boundary of the computational complexity for this process as $N(N-1)/2$, where N is the number of typical testors in the basic matrix. This is the complexity of verifying that the received testors are typical given that they all are typical testors (best case of the final filtering process).

Table 8 shows the runtime, including testor computation and final filtering, for some basic matrices obtained from real data. For this purpose eight standard datasets from the UCI Repository of Machine Learning (Bache & Lichman, 2013) were used. Columns in Table 8 show the dataset name, the number of candidates tested by the BT algorithm, the number of typical testors found, the runtime for the BT algorithm execution on the FPGA device and the final filtering stage on the host PC, which only is needed by the original architecture. For these runtime calculations, a core i5 processor at 3.6GHz was used for the software component and 50MHz for the FPGA architecture.

Table 8: Algorithm execution and typical testors filtering stage runtime for benchmarking datasets

Dataset	Tested Candidates	Typical Testors	FPGA runtime (μs)	PC runtime (μs)
liverdisorder	16	9	0.32	0.04
zoo	20	7	0.40	0.02
krvskp	36	4	0.72	0.01
shuttle	38	19	0.76	0.17
tic-tac-toe	44	9	0.88	0.04
australian	330	44	6.60	0.95
lymphography	802	75	16.04	2.77
german	16921	846	338.42	357.44

For most datasets shown in Table 8 the FPGA and PC runtime are of the same order of magnitude. For those basic matrices with a large number of typical testors, the final filtering stage could be even more expensive than the BT algorithm, as is the case for the *german* dataset. The total runtime for the original architecture is the sum of the FPGA and PC runtimes. Our proposed platform only requires the FPGA runtime, since the proposed modifications do not increase the runtime. Although finding all typical testors for these datasets does not constitute a complex computational problem, they serve to show the benefit of our proposal.

A drawback of our proposed modification is its dependence on the basic matrix dimensions which could lead to a larger hardware architecture (a greater percentage of the FPGA device) when the number of rows is much bigger than the number of columns in the basic matrix.

5.2. A Hardware Architecture based the CT_EXT Algorithm

Following with the critical study of our research proposal, we studied the hardware implementation of the CT_EXT algorithm which is one of the most recent and fastest algorithms reported in the literature, after studying the work in (Rojas *et al.*, 2012) we designed and implemented a new hardware architecture that traverses the search space in a different order than that presented in (Rojas & Cumplido, 2007; Rojas *et al.*,

2012; Rodríguez *et al.*, 2014). This strategy evaluates less candidate subsets than previous architectures, which results in shorter runtime. Moreover, unlike software versions of CT_EXT (Sanchez & Lazo, 2007; Sánchez-Diaz *et al.*, 2010), our proposal evaluates a candidate every clock cycle, which leads to a faster execution. The runtime gain of our new hardware software platform is shown throughout experiments over synthetic datasets.

5.2.1. CT_EXT algorithm

CT_EXT is one of the fastest algorithms for computing all typical testors reported in the literature (Sanchez & Lazo, 2007; Sánchez-Diaz *et al.*, 2010; Piza-Davila *et al.*, 2014). In order to describe this algorithm we introduce some definitions and notations.

Let T be a subset of attributes, T is a testor of a basic matrix BM if the attributes in T do not form a zero row in BM. It means that every row in BM has at least a 1 in those columns corresponding to attributes belonging to T. We say that a testor T is a typical testor if all its proper subsets are not testors.

We can interpret a typical testor as a subset of attributes being jointly sufficient and individually necessary to differentiate every pair of objects belonging to different classes.

During the search, CT_EXT follows the idea that an attribute contributes to a subset T (candidate to be a typical testor) if after adding this attribute to T, the attributes in T form less zero rows in BM than the amount of zero rows before adding the attribute. This idea is used for pruning the search space.

The following proposition, introduced and proved in (Sanchez & Lazo, 2007), constitutes the basis for the CT_EXT algorithm.

Proposition 1. *Given $T \subseteq R$ and $c_j \in R$ such that $c_j \notin T$. If c_j does not contribute to T, then $T \cup \{c_j\}$ cannot be a subset of any typical testor.*

Algorithm 1 shows the pseudocode of CT_EXT, a detailed explanation of this algorithm can be seen in (Sanchez & Lazo, 2007). The function SortBM(BM) sorts the basic matrix as follows. First, it randomly selects one of the rows of BM with the fewest number of 1's. Then, the selected row is moved to the first position and all columns in which it has a 1 are moved to the left.

The function Evaluate(BM, T) returns three values: $testor$, $typical$ and $zero\_rows$. $testor$ is TRUE if the set T is a testor of BM and FALSE otherwise. $typical$ is TRUE if the set T is a typical testor and FALSE otherwise. $zero\_rows$ is the amount of zero rows of T. The function LastOne(T) returns the position of the rightmost 1 of the binary codification of the set T. Notice that for this hardware implementation, an attribute candidate is coded into an n-tuple as we described in the subsection 5.1.

In the proposed hardware platform, we replaced the candidate generator module previously proposed in (Rodríguez *et al.*, 2014), which follows the BT traversing order, by a completely new module following the lexicographical traversing order of CT_EXT. Some design changes were also introduced in the rest of the platform in order to integrate this new module. Moreover, the module for eliminating all non typical testors on the hardware component, proposed in the subsection 5.1, was also included.

Algorithm 1 CT_EXT algorithm

Input: BM - basic matrix with m rows and n columns.
Output: TT - set of typical testors.
$TT \leftarrow \{\}$
$j \leftarrow 0$ ▷ first attribute from BM to be analyzed
$BM \leftarrow$ SortBM(BM)
while $BM[0, j] \neq 0$ **do**
    $T \leftarrow \{c_j\}$ ▷ current attribute subset
    $testor, typical, zero\_rows \leftarrow$ Evaluate(BM, T)
    if $testor = TRUE$ **then**
        if $typical = TRUE$ **then** ▷ T is a typical testor
            $TT \leftarrow TT \cup T$
    else
        $i \leftarrow j + 1$
        while $i < n$ **do**
            $T \leftarrow T \cup \{c_i\}$
            $zero\_rows\_last \leftarrow zero\_rows$
            $testor, typical, zero\_rows \leftarrow$ Evaluate(BM, T)
            if $zero\_rows = zero\_rows\_last$ **then**
                $T \leftarrow T \setminus \{c_i\}$ ▷ attribute c_i does not contribute
            else
                if $testor = TRUE$ **then** ▷ attribute c_i contributes
                    if $typical = TRUE$ **then**
                        $TT \leftarrow TT \cup T$
                    $T \leftarrow T \setminus \{c_i\}$
                    $zero\_rows \leftarrow zero\_rows\_last$
            if $i = n - 1$ **then**
                $k \leftarrow$ LastOne(T)
                if $k = i$ **then**
                    $T \leftarrow T \setminus \{c_k\}$
                    $k \leftarrow$ LastOne(T)
                if $k \neq j$ **then**
                    $T \leftarrow T \setminus \{c_k\}$
                    $testor, typical, zero\_rows \leftarrow$ Evaluate(BM, T)
                    $i \leftarrow k + 1$
                else
                    $i \leftarrow i + 1$
            else
                $i \leftarrow i + 1$
    $j \leftarrow j + 1$

5.2.2. Evaluation and Discussion

In order to show the performance of the proposed platform, it was compared against a software implementation of the CT_EXT algorithm (Sanchez & Lazo, 2007) and the BT hardware platform presented in the subsection 5.1 and reported in (Rodríguez *et al.*, 2014); which is the most recent and fastest hardware implementation for computing typical testors reported in the literature.

Either CT_EXT or BT hardware implementations are capable of evaluating a candidate per clock cycle. If both architectures are running at the same frequency, as it will be the case in our experiments, there are two reasons for runtime differences. The first one is the time taken for reorganizing the basic matrix, which is a more expensive process in BT, although it can be neglected as shown in (Rojas *et al.*, 2012). The second and the most relevant, is the amount of candidates to be evaluated.

Regarding the software implementation, the CT_EXT hardware platform has two disadvantages. First, VHDL code is generated for each BM data, and a synthesis process must be executed previously to executing the algorithm; while this is unnecessary in the software version of CT_EXT. Secondly, the software will be running in a PC at a frequency of 3.10GHz while our FPGA architecture will run at 50MHz.

These disadvantages make our hardware approach useful (faster) under two conditions. First, the number of candidates to be evaluated is big enough to overcome the synthesis overhead. Second, the dimensions of the BM are big enough to provide a considerable speed up of the candidate evaluation process. Although the hardware architecture could be designed for a fixed maximum matrix size and receive the BM through the USB port, by doing this, the size of the problem that can be solved would be significantly reduced. The synthesis process comprehend an optimization of the design, taking advantage of the BM data distribution for reducing the generated hardware configuration. The number of operations for the evaluation of a single candidate, in the software approach, is proportional to the number of rows and it is directly related to the number of columns in the BM. Using this approach, it is possible to achieve a significant runtime reduction, even operating at a much lower clock frequency, by evaluating a candidate on each clock cycle.

With these points in mind, and in order to show the advantages of the proposed platform, three kinds of basic matrices were randomly generated. Each type containing a different percentage of 1's:

1. Very-low density matrices: approximately 8%.
2. Low density matrices: approximately 33%.
3. Medium density matrices: approximately 45%.

Higher density matrices were discarded because they do not constitute a computationally expensive problem, as stated by Rojas *et al.* (2012). Here after, we will be referring to these three sets of matrices by its approximate density of 1's.

For our experiments, 30 basic matrices of different sizes were randomly generated. A random number generator was used to generate rows, which are filtered for the minimum and maximum number of 1's allowed. In this way the desired density was controlled. If accepted, a row is verified as basic against the saved rows. Basic rows are saved until the desired number of rows is reached.

For the hardware platforms, we measure the runtimes including the time for the following stages: BM input parsing and VHDL code generation, synthesis process, and typical testor computation (with the hardware component running at 50MHz). The number of rows for each type of matrices (very-low: 400, low: 225, medium: 255) is conditioned by the dimensions of the biggest matrix that can be synthesized at the desired

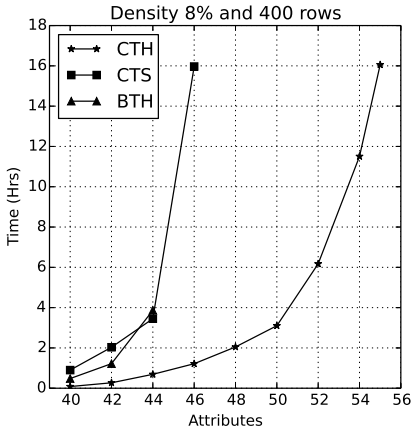

Figure 3: Total runtime for density 8%.

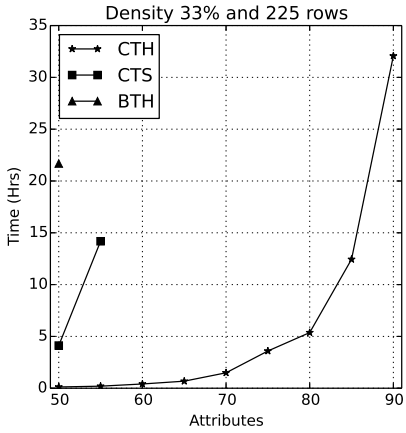

Figure 4: Total runtime for density 33%.

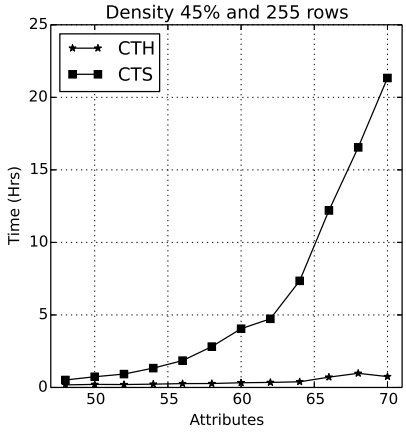

Figure 5: Total runtime for density 45%.

running frequency. All experiments were performed using an Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz for software and an Atlys board, powered by a Spartan-6 LX45 FPGA device, for the hardware. Figures 3, 4, and 5 show graphics of the runtime (in hours) for the three types of basic matrices.

The results of the proposed CT_EXT hardware platform (CTH) were taken as reference for axis limits in Figures 3, 4, and 5. Slowest executions of the CT_EXT software implementation (CTS) are not shown in order to keep clarity in the figures. The hardware platform for BT (BTH) was not able to meet the constrain of 50MHz clock frequency for some matrices.

Experiment results show that the proposed platform beats the software implementation of the CT_EXT algorithm, with ratios of around one order of magnitude. However, for large enough datasets this improvement could be significantly higher, as it can be inferred from Figure 5.

5.3. A New Recursive Algorithm for Reduct Computation

In this subsection we propose a new **R**ecursive algorithm using the **S**implified **D**iscernibility **M**atrix (RSDM) for computing all the reducts in a dataset. This new algorithm is based on the concept of contribution like CT_EXT; but non contributing attributes are discarded a priori, avoiding even more unnecessary evaluations. Compared to CT_EXT, the number of evaluated candidates in RSDM is smaller; but each evaluation has a higher cost.

5.3.1. Concepts for RSDM

In this subsection, we will be working with the binary representation of the simplified discernibility matrix (SDM). The following definition of super-reduct is equivalent to the condition 1 stated in Subsection 2.3 (Lazo-Cortés *et al.*, 2015).

Definition 1. *Let $T \subseteq R$ be a subset of attributes from a dataset. T is a super-reduct iff in the sub-matrix of SDM formed by the columns corresponding to attributes in T, there is not any zero row (row having only zeros).*

The following definition constitutes the key concept in CT_EXT and RSDM, and it is also an important component of the algorithms BR and RGonCRS.

Definition 2. *Let $T \subseteq R$ and $c_i \in R$ such that $c_i \notin T$. We say that c_i contributes to T iff the number of zero rows, in the sub-matrix of SDM formed by the columns corresponding to attributes in $T \cup \{c_i\}$, is lower than in T.*

For a fast implementation of the RSDM algorithm, columns in SDM are coded as binary words with as many bits as rows in the SDM. The cumulative mask for an attribute c_i, denoted as cm_{c_i}, is defined as the binary word representing the ith column in SDM. The cumulative mask for a subset of attributes $T = \{c_{i1}, c_{i2}, ..., c_{ik}\}$ is defined as $cm_T = cm_{c_{i1}} \vee cm_{c_{i2}} \vee ... \vee cm_{c_{ik}}$ where $\vee$ represents the binary OR operator. It is not hard to see that the number of 0's in cm_T is the same as the number of zero rows in the sub-matrix of SDM formed by the columns corresponding to attributes in T. According to the definition 2, c_i contributes to T iff $cm_{T \cup c_i}$ has more 1's than cm_T. The incremental nature of RSDM is given by the associative property of the OR operation, such that $cm_{T \cup c_i} = cm_T \vee cm_{c_i}$. Notice that, from this last formulation, c_i contributes to T iff $cm_{T \cup c_i} \neq cm_T$ since $cm_{T \cup c_i}$ cannot have less 1's than cm_T. It is easy to see, from the definition 1 that $T \subseteq R$ is a super-reduct iff $cm_T = (1, ..., 1)$ (has a 1 in every bit).

The RSDM algorithm is supported by the proposition 2.

Proposition 2. *Given $T \subseteq Z \subseteq R$ and $c_i \in R$ such that $c_i \notin Z$. If c_i does not contribute to T or form a super-reduct with T, then $Z \cup \{c_i\}$ cannot be a subset of any reduct.*

Proof 1. *From the proposition 1, if c_j does not contribute to T, then $T \cup \{c_j\}$ cannot be a subset of any reduct. Since $\{T \cup \{c_i\}\} \subseteq \{Z \cup \{c_i\}\}$, if $Z \cup \{c_j\}$ is a subset of a reduct, then $T \cup \{c_j\}$ must be a subset of a reduct; which contradicts the proposition 1. If $T \cup \{c_i\}$ is a super-reduct, $Z \cup \{c_i\}$ cannot be a*

reduct, since it is a superset of a super-reduct; which contradicts the condition 2 of the definition of reduct (Subsection 2.3)[3] .

5.3.2. The RSDM Algorithm

The first step in the RSDM algorithm consist in sorting the SDM in order to reduce the search space. The sorting process is the same that we presented in the subsection 5.2 for CT_EXT. A SDM, sorted in this way, can be seen in Table 9. The pseudocode for RSDM is shown in Algorithm 2.

Table 9: Sorted Simplified Discernibility Matrix

c_0	c_1	c_2	c_3	c_4
1	1	0	0	0
0	1	0	0	1
1	0	0	1	0
0	0	1	0	1

The algorithm execution starts with the first element (column) in the sorted SDM and repeats until an attribute with a 0 in the first row is reached (line 2). Notice that for the rest of the candidates, the first row will always be an empty row. If this single attribute is a super-reduct (line 3) then it is saved in the super-reduct set SR; else, the recursive evaluator (line 8) is called with the current attribute as the base set B and the remaining attributes to its right as the tail set C (line 6). In the recursive procedure, every attribute in the tail is tested whether contributes or not to the base set (line 10). The attributes are removed from the tail set for furthers evaluations, if they form a super-reduct (line 13) or they do not contribute to B (line 16). This a priori evaluation and rejection of attributes constitutes the key of our proposed algorithm, and it is supported by proposition 2. Finally, the remaining attributes in the tail set are concatenated with B, one at a time, and used as base set for subsequent recursive evaluations with the remaining attributes in C as tail (line 19).

Lets take for example the simplified discernibility matrix from Table 9. The candidates evaluated by CT_EXT and RSDM are shown in Table 10. Notice that while CT_EXT evaluates 17 candidates, in RSDM, only 13 candidates are evaluated.

Table 10: Candidates evaluated by CT_EXT and RSDM for the matrix of Table 9

CT_EXT					RSDM			
$\{c_0\}$	$\{c_0, c_1, c_4\}$	$\{c_0, c_3\}$	$\{c_1, c_2, c_3\}$	$\{c_1, c_4\}$	$\{c_0\}$	$\{c_0, c_4\}$	$\{c_1, c_3\}$	$\{c_1, c_3, c_4\}$
$\{c_0, c_1\}$	$\{c_0, c_2\}$	$\{c_0, c_4\}$	$\{c_1, c_2, c_4\}$		$\{c_0, c_1\}$	$\{c_0, c_1, c_2\}$	$\{c_1, c_4\}$	
$\{c_0, c_1, c_2\}$	$\{c_0, c_2, c_3\}$	$\{c_1\}$	$\{c_1, c_3\}$		$\{c_0, c_2\}$	$\{c_1\}$	$\{c_1, c_2, c_3\}$	
$\{c_0, c_1, c_3\}$	$\{c_0, c_2, c_4\}$	$\{c_1, c_2\}$	$\{c_1, c_3, c_4\}$		$\{c_0, c_3\}$	$\{c_1, c_2\}$	$\{c_1, c_2, c_4\}$	

5.3.3. Evaluation and Discussion

In order to evaluate the performance of RSDM, we present a comparative execution over synthetically generated simplified discernibility matrices. For this experiment, we selected CT_EXT as reference algorithm because it is one of the fastest algorithms reported in the literature. Comparing RSDM with CT_EXT

[3]Herein after denoted as irreducible condition

Algorithm 2 Recursively calculate super-reducts in SDM

Input: Sorted SDM

Output: SR - set of super-reducts containing all reducts

```
1:  i ⇐ 0, SR ⇐ ∅
2:  while SDM(0, i) ≠ 0 do
3:      if cm_{c_i} = (1, ..., 1) then
4:          SR ⇐ SR ∪ {c_i}
5:      else
6:          eval({c_i}, cm_{c_i}, {c_{i+1}, ..., c_m})
7:      i ⇐ i + 1
8:  eval(B,cm_B,C)
9:  for all c ∈ C do
10:     cm_{B∪{c}} = cm_B ∨ cm_c
11:     if cm_{B∪{c}} ≠ cm_B then
12:         if cm_{B∪{c}} = (1, ..., 1) then
13:             C ⇐ C \ c
14:             SR ⇐ SR ∪ {B ∪ {c}}
15:     else
16:         C ⇐ C \ c
17: for all c ∈ C do
18:     C ⇐ C \ c
19:     eval(B ∪ {c},cm_{B∪{c}},C)
```

makes possible to evaluate the effect of the difference in the traversing order between these two algorithms; and its relation to some SDM properties (the number of rows and the density of 1's).

For our experiments we used 380 $SDMs$ divided into five groups regarding their number of rows (200, 1100, 2000, 2900 and 3800 rows). Each matrix has 30 attributes and was randomly generated using the procedure exposed in the subsection 5.2. Our controlled variables are the number of rows in the SDM and the density of 1's. Matrices were divided into three levels of density:

1. Very-low density: approximately 20% (75 matrices).
2. Low density: approximately 30% (125 matrices).
3. Medium density: approximately 47% (180 matrices).

For this experiment, CT_EXT and RSDM were applied over every SDM. Each combination was executed three times and the runtime was estimated using the fastest execution. This process ensures a 95% confidence interval according to Haveraaen *et al.* (2001). The execution order was randomized to control the runtime bias due to the operating system load. All experiments were run on a G1620 Intel processor at 2.7GHz with 2GB in RAM over a GNU/Linux operating system.

Figures 6, 7 and 8 show the runtime of CT_EXT and RSDM for $SDMs$ with very-low, low and medium densities respectively. Each point, in these figures, represents the average runtime for all the matrices having the same number of rows. Figure 9 shows the average runtime of CT_EXT and RSDM for all the $SDMs$, without taking their density into account.

From Figures 6-8 we notice a positive correlation between the performance improvement of RSDM over

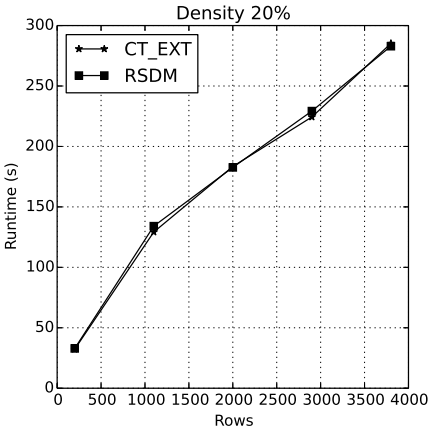

Figure 6: Runtime for matrices with densities around 20%.

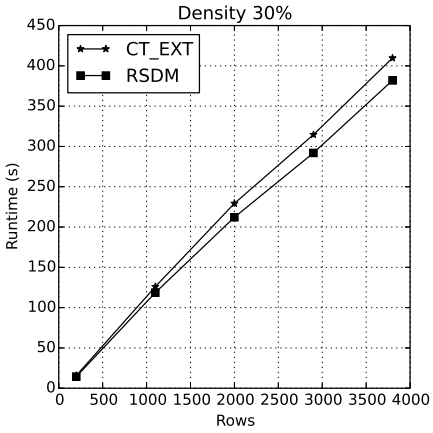

Figure 7: Runtime for matrices with densities around 30%.

CT_EXT, and the density of 1's in the SDM. Indeed, there is no apparent difference between RSDM and CT_EXT runtime for very-low density matrices. In RSDM, every remaining attribute (according to the lexicographical traversing order) is evaluated for contribution with the current candidate subset. When some of the remaining attributes do not contribute to the current candidate subset, they are excluded from subsequent evaluations over supersets of the current candidate. CT_EXT, on the other hand, strictly follows the lexicographical order, and may evaluate a non contributing attribute several times. However, this advantage is reduced for matrices with a very low density of 1's, where the probability of finding non contributing attributes is lower.

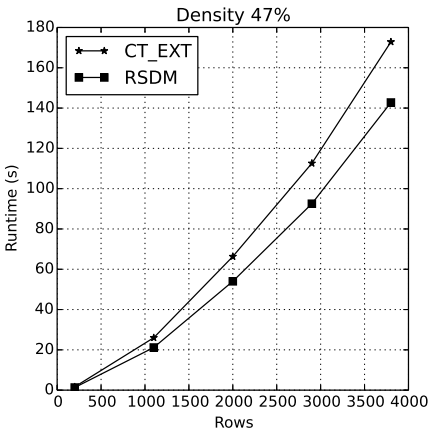

Figure 8: Runtime for matrices with densities around 47%.

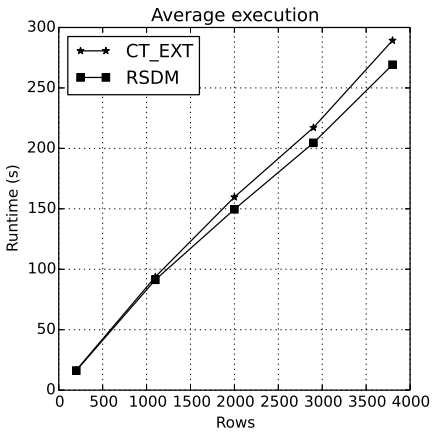

Figure 9: Average runtime.

Another important peculiarity that we notice from Figures 6-9 is an increasing runtime reduction of RSDM over CT_EXT, regarding the number of rows in the SDM. This trend is because the recursive nature of RSDM requires a complex memory handling to keep the cumulative mask of evaluated candidates. CT_EXT, on the other hand, uses a simple table to this end; which is indexed by the current evaluated attribute. As a result, RSDM requires a larger number of operations for each candidate evaluation, but these extra operations are independent of the SDM dimensions. With the increase of the number of rows, the cost of evaluating a single candidate is greater. Thus, for matrices with a larger number of rows, the improvement of evaluating less candidates becomes significant in relation to the cost of the extra operations.

Paired one-sided t-test

data: CT_EXT runtime and RSDM runtime
t = 25.5347, df = 251, p-value $<$ 2.2e-16
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
16512.34ms Inf
sample estimates:
mean of the differences
17653.75ms

Figure 10: R output for the t-test of mean CT_EXT and RSDM runtime.

From Figures 7 and 8, we propose to apply a one-sided t-test to assess the relative performance of RSDM over CT_EXT in low and medium density matrices with more than 1000 rows. We selected from our original 380 matrices, the 252 $SDMs$ having these characteristics. Our **null hypothesis** is: *there is no difference between the RSDM and CT_EXT runtime for $SDMs$ with density between 0.3 and 0.5, and more than 1000 rows*; and our **alternative hypothesis**: *the runtime of CT_EXT is higher than the runtime of RSDM for $SDMs$ with density between 0.3 and 0.5, and more than 1000 rows*. The output from the R[4] software (Figure 10) supports the alternative hypothesis beyond a 95% confidence interval.

Based on this result, we can conclude that RSDM is faster than CT_EXT for matrices with more than 1000 rows and density between 0.3 and 0.5.

5.4. A New Algorithm for Reduct Computation Based on the Gap Elimination and Contribution

In this subsection we propose a new algorithm based on the **G**ap elimination and **C**ontribution, using the **S**implified **D**iscernibility **M**atrix (GCSDM), for computing all the reducts in a dataset. The main difference between the algorithm introduced in this section and CT_EXT is the gap elimination; which leads to a great runtime reduction as we will show afterwards. LEX and fast-BR algorithms also use gap elimination but they are based on the concept of exclusion (Lias-Rodríguez & Sanchez-Diaz, 2013); which implies several iterations for evaluating each candidate. GCSDM, on the other hand, is based on the concept of contribution (as CT_EXT), which allows a simpler candidate evaluation. The concept of exclusion is used in GCSDM, over the identified super-reducts, to verify the irreducible condition.

5.4.1. Concepts for GCSDM

The concept of gap was first introduced by Santiesteban & Pons (2003) for the LEX algorithm, as shown in the definition 3.

Definition 3. *Let $L \subseteq R$ be a subset of attributes from a dataset, such that $L = \{c_{j_0}, ..., c_{j_s}\}$ and $c_{j_0} < \cdots < c_{j_s}$ according to their order in the dataset. The gap of L is the attribute $c_p \in L$ with $j_0 \leq p < j_s$ such that $p = \max(j_q | c_{j_q}, c_{j_{q+1}} \in L \wedge j_{q+1} \neq j_q + 1)$.*

[4]http://www.r-project.org

In other words, the gap is the attribute in L with the highest index such that its consecutive attribute in L is not its consecutive attribute in SDM.

Table 11: A sorted Simplified Discernibility Matrix

c_0	c_1	c_2	c_3	c_4	c_5	c_6
1	0	0	0	0	0	0
0	1	0	1	0	0	0
0	0	0	0	0	0	1
0	0	1	0	1	0	0
0	0	0	0	0	1	0

Lets take for example the simplified discernibility matrix from Table 11:

$$\begin{array}{ll} \{c_0, c_1, c_2, c_3\} & \text{there is no gap} \\ \{c_0, c_1, c_2, c_5, c_6\} & \text{the gap is } c_2 \\ \{c_0, c_1, c_2, c_4, c_6\} & \text{the gap is } c_4 \\ \{c_0, c_1, c_4, c_5, c_6\} & \text{the gap is } c_1 \end{array}$$

The gap elimination is supported by the proposition 3 (Santiesteban & Pons, 2003). In order to clarify the basis of the gap elimination we show the lexicographical traversing order for a SDM with three attributes (columns):

$\{c_0\}, \{c_0, c_1\}, \{c_0, c_1, c_2\}, \{c_0, c_2\}, \{c_1\}, \{c_1, c_2\}, \{c_2\}$

Proposition 3. *Let $L \subseteq R$ be a reduct, such that $L = \{c_{j_0}, ..., c_{j_s}\}$, $c_{j_0} < \cdots < c_{j_s}$ and c_{j_s} is the last attribute in SDM. If there is a gap c_p in L, and $L' = \{c_{j_0}, ..., c_{j_k}, c_{p+1}\}$, $j_0 < \cdots < j_k <= p-1$; then, no attribute subset between L and L' (following the lexicographical order) is a reduct.*

From the proposition 3 we have the following corollary; which is relevant to GCSDM in order to avoid other unnecessary evaluations.

Corollary 1. *Let $L \subseteq R$ be a non super-reduct, such that $L = \{c_{j_0}, ..., c_{j_s}\}$, $c_{j_0} < ... < c_{j_s}$ and c_{j_s} is the last attribute in SDM. If there is a gap c_p in L, and $L' = \{c_{j_0}, ..., c_{j_k}, c_{p+1}\}$, $j_0 < ... < j_k <= p-1$; then, no attribute subset between L and L' (following the lexicographical order) is a reduct.*

In Table 12 we show a partial execution of CT_EXT over the SDM of Table 11 to illustrate the gap elimination. The first 17 evaluated candidates out of 41, are presented along with their evaluation. For this particular case, GCSDM evaluates only 23 candidates for a runtime reduction above the 40%. Candidates of iterations 7 and 12 are reducts, both including the last attribute in SDM (c_6). Candidates of iterations 8, 13, 14, 15 and 16 are proper subsets of reducts, as indicated by the proposition 3, and they cannot be a reduct. Thus, the evaluation of shadowed candidates in Table 12 is unnecessary.

In order to determine whether a super-reduct is a reduct (verifying the irreducible condition) the exclusion mask, introduced by Lias-Rodríguez & Pons-Porrata (2009), plays a fundamental role.

Table 12: Partial execution of CT_EXT over the SDM of Table 11. The evaluation of a candidate is labelled as C - the added attribute contributes, NC - the added attribute does not contribute, R - the candidate is a reduct, or NSR - the last attribute is reached and the candidate is not a super-reduct.

Iter	Candidate	Evaluation	Iter	Candidate	Evaluation
1	$\{c_0\}$	C	10	$\{c_0, c_1, c_4\}$	C
2	$\{c_0, c_1\}$	C	11	$\{c_0, c_1, c_4, c_5\}$	C
3	$\{c_0, c_1, c_2\}$	C	12	$\{c_0, c_1, c_4, c_5, c_6\}$*	R
4	$\{c_0, c_1, c_2, c_3\}$	NC	13	$\{c_0, c_1, c_4, c_6\}$	NSR
5	$\{c_0, c_1, c_2, c_4\}$	NC	14	$\{c_0, c_1, c_5\}$	C
6	$\{c_0, c_1, c_2, c_5\}$	C	15	$\{c_0, c_1, c_5, c_6\}$	NSR
7	$\{c_0, c_1, c_2, c_5, c_6\}$*	R	16	$\{c_0, c_1, c_6\}$	NSR
8	$\{c_0, c_1, c_2, c_6\}$	NSR	17	$\{c_0, c_2\}$	C
9	$\{c_0, c_1, c_3\}$	NC	...	...	...

Definition 4. *Let $L \subseteq R$ be a subset of attributes from a dataset. We call exclusion mask of L, denoted as em_L, to the binary word in which the i^{th} bit is 1 if the i^{th} row in SDM has a 1 in only one column of those columns corresponding to attributes in L, and it is 0 otherwise.*

For instance, from the SDM in Table 11 we have:

$$\begin{aligned} em_{\{c_0,c_1,c_2\}} &= (1,1,0,1,0) \\ em_{\{c_0,c_1,c_2,c_3\}} &= (1,0,0,1,0) \\ em_{\{c_0,c_1,c_2,c_3,c_4\}} &= (1,0,0,0,0) \end{aligned}$$

Lias-Rodríguez & Sanchez-Diaz (2013) introduced the following proposition to support the cumulative computation of the exclusion mask.

Proposition 4. *Let $L \subseteq R$ be a subset of attributes from a dataset and $c \notin L$ an attribute of SDM. The exclusion mask of $L \cup \{c\}$ is calculated as follows:*

$$em_{L\cup\{c\}} = (em_L \wedge \neg cm_c) \vee (\neg cm_L \wedge cm_c)$$

where cm refers to the cumulative mask.

Finally, they stated and proved the following proposition.

Proposition 5. *Let $L \subseteq R$ be a subset of attributes from a dataset and $c \notin L$ an attribute of SDM. If $\exists c_i \in L$ such that $em_{L\cup\{c\}} \wedge cm_{c_i} = (0, ..., 0)$. Then, c does not form a reduct with L.*

5.4.2. The GCSDM Algorithm

The first step in the GCSDM algorithm consist in sorting the SDM in the same way as we did for RSDM. The pseudocode for GCSDM is shown in Algorithm 3. Unlike RSDM, this algorithm returns only the set of all reducts.

We start with an empty base subset B and the position of the first attribute in c[5]. Then we assign a null vector to the cumulative mask (cm_B) for $B = \emptyset$ or retrieve its calculated value for $B \neq \emptyset$ (lines 4-7). CM stores the calculated cumulative masks, indexed by the last attribute in B. The function getLast(B) returns the last attribute in B. In the line 8, we update the cumulative mask and in the line 10 we evaluate the contribution of c to B. The cumulative mask of an attribute cm_c has the same meaning as in RSDM. If the current attribute c contributes, we evaluate the super-reduct condition on $B \cup \{c\}$ (line 12). For super-reducts, we use the proposition 4 to compute the exclusion mask $em_{B\cup\{c\}}$ (lines 15-17) and we verify the irreducible condition (lines 18- 22) by means of the proposition 5. At this point, the candidate evaluation is finished.

From line 25 to the end, the next candidate subset ($B \cup \{c\}$) is generated. LastAttribute holds the position of the last attribute in SDM. If the last attribute is reached, we check if the current candidate is a reduct or it is not a super-reduct, to eliminate the gap (lines 26-32). If the last attribute is reached but the current candidate is a super-reduct, the last attribute in B is removed (line 34). If the last attribute is not reached yet, there are two possibilities. The current candidate is a super-reduct or the current attribute c does not contribute to B; then we must replace c by the next attribute in SDM (line 37). The attribute c contributes to B and the current candidate is not a super-reduct; then the current attribute is added to B (line 39) and the next attribute in SDM is loaded to c (line 40). The algorithm finishes when the column corresponding to the first attribute in the current candidate has a 0 in the first row.

In Table 13 we show an execution example of GCSDM over the SDM from Table 11. The columns labelled C, SR and R represent the result of the candidate evaluation on **C**ontribution, **S**uper-**R**educt and **R**educt conditions, respectively. Notice that the gap elimination occurs after candidates of iterations 7, 11, 16, 23 (reducts) and 19 (not super-reduct).

Alba-Cabrera *et al.* (2014) pointed out that CT_EXT has a surprisingly low performance over identity matrices. In fact; CT_EXT, for this kind of matrices, traverses the complete power set of attribute combinations. The introduction of the gap elimination solves this drawback in such a way that a minimum number of candidate verifications is required for this kind of matrices.

5.4.3. Evaluation and Discussion

In order to evaluate the performance of GCSDM, we present a comparative analysis of different algorithms over synthetically generated simplified discernibility matrices ($SDMs$). For this experiment, we selected CT_EXT and fastBR, reported in the literature as the fastest algorithms for typical testor computation, in addition to our two proposals. Algorithms for computing all reducts such as RGonCRS (Wang, 2007) and that proposed in (Starzyk *et al.*, 2000) were not included since they showed a lower performance in our preliminary experiments.

This experiment was done over 57 randomly generated $SDMs$. The density of 1's in the SDM and the standard deviation of the density of 1's in the rows of the SDM are shown in the Figures 11 and 12 respectively. The selected algorithms (CT_EXT, fastBR, RSDM and GCSDM) were applied over every matrix. Each combination was executed three times and the runtime is estimated using the fastest execution. The execution order was randomized to control the runtime bias due to the operating system load, as we did in the subsection 5.3. All experiments were run on a G1620 Intel processor at 2.7GHz with 2GB in RAM over a GNU/Linux operating system.

[5]Here we abuse of notation by using c for both an attribute and a number denoting its position in SDM

Algorithm 3 GCSDM algorithm for computing all reducts

Input: Sorted SDM
Output: RR - set of all reducts

1: $B \Leftarrow \emptyset$, $RR \Leftarrow \emptyset$, $c \Leftarrow 0$
2: **while not** done **do**
3: $reduct \Leftarrow$ False, $superReduct \Leftarrow$ False, $contributes \Leftarrow$ False
4: **if** $B = \emptyset$ **then**
5: $cm_B \Leftarrow (0, ..., 0)$
6: **else**
7: $cm_B \Leftarrow CM[\text{getLast}(B)]$
8: $cm_{B \cup \{c\}} \Leftarrow cm_B \vee cm_c$
9: $CM[c] \Leftarrow cm_{B \cup \{c\}}$
10: **if** $cm_{B \cup \{c\}} \neq cm_B$ **then**
11: $contributes \Leftarrow$ True
12: **if** $cm_{B \cup \{c\}} = (1, ..., 1)$ **then**
13: $superReduct \Leftarrow$ True
14: $em_{B \cup \{c\}} \Leftarrow (0, ..., 0)$, $cm \Leftarrow (0, ..., 0)$
15: **for all** $x \in B \cup \{c\}$ **do**
16: $em_{B \cup \{c\}} \Leftarrow (em_{B \cup \{c\}} \wedge \neg cm_x) \vee (\neg cm \wedge cm_x)$
17: $cm \Leftarrow CM[x]$
18: $reduct \Leftarrow$ True
19: **for all** $x \in B$ **do**
20: **if** $em_{B \cup \{c\}} \wedge cm_x = (0, ..., 0)$ **then**
21: $reduct \Leftarrow$ False
22: **break**
23: **if** $reduct$ **then**
24: $RR \Leftarrow RR \cup \{B \cup \{c\}\}$
25: **if** $c =$LastAttribute **then** ▷ Reached the last column of the binary SDM
26: **if** $reduct$ **or not** $superReduct$ **then** ▷ Eliminate existing gap
27: $last \Leftarrow c$
28: **while** $\text{getLast}(B) = (last - 1)$ **do**
29: $last \Leftarrow \text{getLast}(B)$
30: $B \Leftarrow B \setminus last$
31: **if** $|B| = 1$ **then**
32: **break**
33: $c \Leftarrow \text{getLast}(B) + 1$
34: $B \Leftarrow B \setminus \text{getLast}(B)$
35: **else**
36: **if not** $contributes$ **or** $superReduct$ **then**
37: $c \Leftarrow c + 1$
38: **if** $contributes$ **and not** $superReduct$ **then**
39: $B \Leftarrow B \cup c$
40: $c \Leftarrow c + 1$
41: **if** $B = \emptyset$ **and** $cm_c[0] = 0$ **then** ▷ First attribute has a 0 in the first row
42: $done \Leftarrow$ True

Table 13: Sample Execution of GCSDM.

Iter	B	c	$B \cup \{c\}$	C	SR	R	Comments
1	$\{\}$	0	$\{c_0\}$	True	False		Add a new attribute.
2	$\{c_0\}$	1	$\{c_0, c_1\}$	True	False		Add a new attribute.
3	$\{c_0, c_1\}$	2	$\{c_0, c_1, c_2\}$	True	False		Add a new attribute.
4	$\{c_0, c_1, c_2\}$	3	$\{c_0, c_1, c_2, c_3\}$	False			Remove c_3 (proposition 2).
5	$\{c_0, c_1, c_2\}$	4	$\{c_0, c_1, c_2, c_4\}$	False			Remove c_4 (proposition 2).
6	$\{c_0, c_1, c_2\}$	5	$\{c_0, c_1, c_2, c_5\}$	True	False		Add a new attribute.
7	$\{c_0, c_1, c_2, c_5\}$	6	$\{c_0, c_1, c_2, c_5, c_6\}$	True	True	True	Eliminate the gap (c_2).
8	$\{c_0, c_1\}$	3	$\{c_0, c_1, c_3\}$	False			Remove c_3 (proposition 2).
9	$\{c_0, c_1\}$	4	$\{c_0, c_1, c_4\}$	True	False		Add a new attribute.
10	$\{c_0, c_1, c_4\}$	5	$\{c_0, c_1, c_4, c_5\}$	True	False		Add a new attribute.
11	$\{c_0, c_1, c_4, c_5\}$	6	$\{c_0, c_1, c_4, c_5, c_6\}$	True	True	True	Eliminate the gap (c_1).
12	$\{c_0\}$	2	$\{c_0, c_2\}$	True	False		Add a new attribute.
13	$\{c_0, c_2\}$	3	$\{c_0, c_2, c_3\}$	True	False		Add a new attribute.
14	$\{c_0, c_2, c_3\}$	4	$\{c_0, c_2, c_3, c_4\}$	False			Remove c_4 (proposition 2).
15	$\{c_0, c_2, c_3\}$	5	$\{c_0, c_2, c_3, c_5\}$	True	False		Add a new attribute.
16	$\{c_0, c_2, c_3, c_5\}$	6	$\{c_0, c_2, c_3, c_5, c_6\}$	True	True	True	Eliminate the gap (c_3).
17	$\{c_0, c_2\}$	4	$\{c_0, c_2, c_4\}$	False			Remove c_4 (proposition 2).
18	$\{c_0, c_2\}$	5	$\{c_0, c_2, c_5\}$	True	False		Add a new attribute.
19	$\{c_0, c_2, c_5\}$	6	$\{c_0, c_2, c_5, c_6\}$	True	False		Eliminate the gap (c_2).
20	$\{c_0\}$	3	$\{c_0, c_3\}$	True	False		Add a new attribute.
21	$\{c_0, c_3\}$	4	$\{c_0, c_3, c_4\}$	True	False		Add a new attribute.
22	$\{c_0, c_3, c_4\}$	5	$\{c_0, c_3, c_4, c_5\}$	True	False		Add a new attribute.
23	$\{c_0, c_3, c_4, c_5\}$	6	$\{c_0, c_3, c_4, c_5, c_6\}$	True	True	True	Eliminate the gap (c_0).
24	$\{\}$	1	$\{c_1\}$	Algorithm finishes because c_1 has a 0 in the first row of SDM (line 41)			

In order to obtain the desired density and standard deviation of density in rows, we used the matrix generator described in subsection 5.2.2. We control the maximum and minimum number of 1's in a randomly generated row. In this way, the desired density is computed as the mean number of 1's in a row, divided by the number of attributes (columns). Keeping constant this central value, we can modify at the same time the maximum and minimum number of allowed 1's in a row to control the standard deviation. For each desired pair of values (density and standard deviation), three different matrices were generated. A total of 57 matrices with 30 columns and 2000 rows were generated.

Figure 13 shows a scatter graph of the runtime as a function of the density of 1's in the SDM. The runtime is taken from the fastest algorithm for each case. The dots are grouped by the best performing (fastest) algorithm for that particular matrix as shown in the legend of the figure. There can be identified three distinct groups of matrices by their density:

- Low density matrices: density < 0.3.
- Medium density matrices: $0.3 <$ density < 0.7.
- High density matrices: density > 0.7.

The fastest algorithm for low density matrices is GCSDM, fastBR was the fastest for medium density matrices and RSDM outperforms the other algorithms in most of the high density matrices. The fact that high

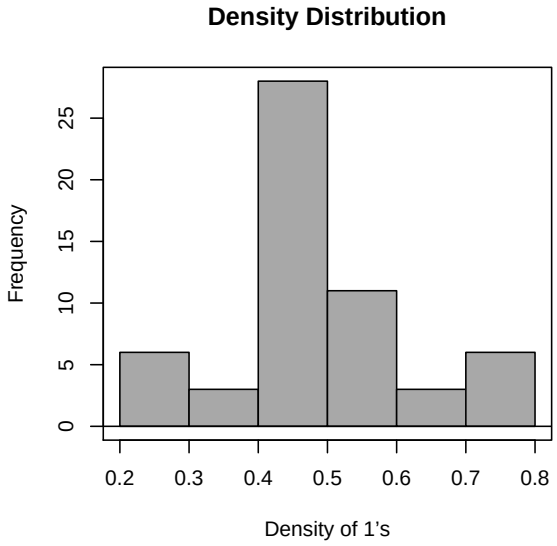

Figure 11: Distribution of densities in the $SDMs$ under study.

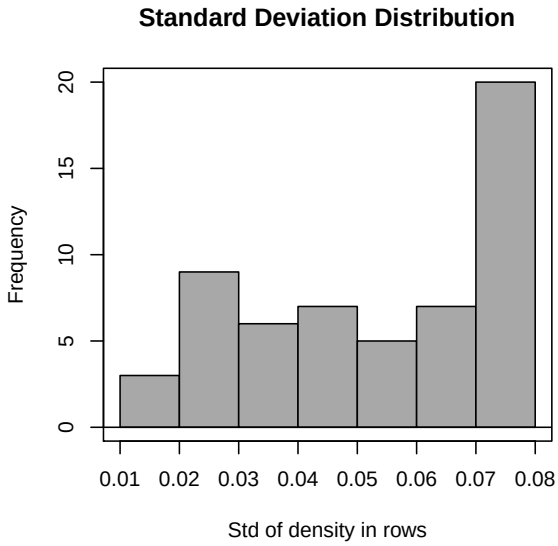

Figure 12: Distribution of standard deviation of density in rows for all $SDMs$ under study.

density $SDMs$ do not constitute a complex computational task for reduct computation is clearly visible in Figure 13. Notice that CT_EXT was never the best performing algorithm while fastBR was the fastest in most matrices.

Since all matrices for this experiment have the same dimensions, we can say that low density matrices showed a relatively high complexity. For this reason, the apparent fact that GCSDM outperforms the other algorithms for this kind of matrices, deserves special attention. On the other hand, the result of RSDM on high density matrices seems less important.

Figure 14 shows a scatter graph of the runtime as a function of the standard deviation of the density of 1's in the rows of the SDM. As it can be seen, there is no clear relationship between the runtime of the fastest algorithm and the standard deviation of density. Notice (combining information from Figures 13 and 14) that both, low and high density matrices are related to low values of the standard deviation. In fact, it is not possible to increase the standard deviation in the extreme values of density.

Figure 15 shows a scatter graph of the runtime as a function of the number of reducts in the SDM. There can be seen a positive correlation between these two variables. This is a natural trend, since the time complexity is at least as high as the space complexity; which exceeds the size of the solution. For high density matrices, there are a low number of reducts which partially explains their lower computational cost (see Figure 15).

Figure 16 shows a scatter graph of the runtime of CT_EXT vs. GCSDM for the $SDMs$ used in this experiment. Although GCSDM outperforms CT_EXT in most cases. Based on the evidence of Figure 16, we proposed a one-sided t-test to evaluate the overall performance of GCSDM over CT_EXT. Our **null hypothesis** is: *there is no difference between the GCSDM and CT_EXT runtime*. As **alternative hypothesis** we have: *the runtime of CT_EXT is higher than the runtime of GCSDM*. The output from the R software (Figure 17) supports the alternative hypothesis beyond a 95% confidence interval.

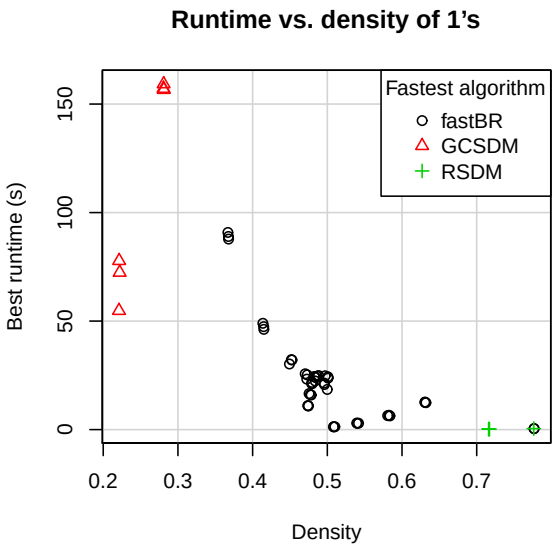

Figure 13: Fastest algorithm runtime vs. density of 1's for all $SDMs$ under study.

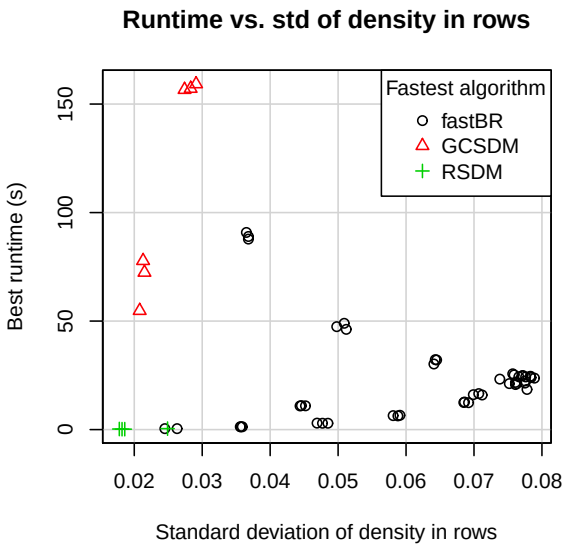

Figure 14: Fastest algorithm runtime vs. standard deviation of density in rows for all $SDMs$ under study.

In order to evaluate the performance improvement of GCSDM over fastBR, which is the fastest algorithm reported in the literature, we present a new experiment. A total of 83 low density matrices, with 30 columns, was generated using the same procedure above described. This time, we generated $SDMs$ with 1000 rows. We evaluated the runtime of reduct computation over all matrices using fastBR and GCSDM. Again we ran three execution of each possible combination in a randomized experiment.

Figure 18 shows the runtime as a function of the density of 1's in the SDM. There can be seen two interesting facts. First, there is a well defined line delimiting those $SDMs$ for which GCSDM is faster than fastBR (densities under 0.3). Second, the runtime of GCSDM increases monotonically with the increase of the density of 1's in the SDM.

In Figure 19 we show the runtime rate between fastBR and GCSDM; where values above 1 mean a runtime improvement of GCSDM over fastBR. For lower densities we have up to two orders of magnitude rate. We found an exponential relationship between this rate and the density of the SDM. Notice that the vertical axis in Figure 19 is logarithmic.

In order to explain this behaviour we must go deeply into the main difference between fastBR and GCSDM. In GCSDM we compute the exclusion mask and evaluate the proposition 5 only for those candidates proven as super-reducts, in order to verify them as reducts. In fastBR, on the other hand, the proposition 5 is evaluated for each contributing candidate. As a result, fastBR evaluates less candidates than GCSDM but at a higher cost per candidate. The attribute exclusion occurs when there is at least one column, in the sub-matrix of SDM formed by those attributes in the current candidate, that can be removed without increasing the number of zero rows in this sub-matrix. The exclusion is more frequent in matrices with a higher density of 1's, as it can be inferred from Figure 19.

Lets take for instance the extreme case of the identity matrix where there is no exclusion at all, since every attribute is indispensable to form a reduct. For this SDM, GCSDM needs to evaluate as many candidates as

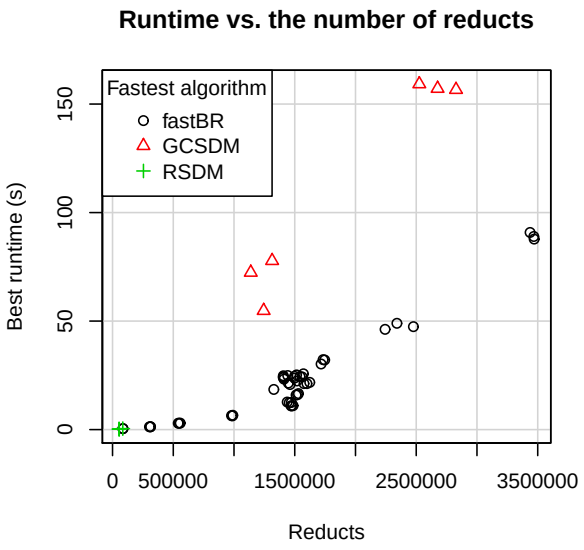

Figure 15: Fastest algorithm runtime vs. the number of reducts.

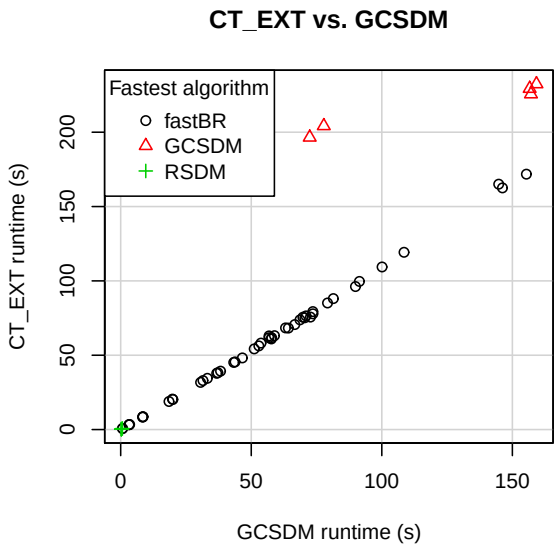

Figure 16: CT_EXT runtime vs. GCSDM runtime.

Paired one-sided t-test

data: CT_EXT runtime and GCSDM runtime
t = 3.3, df = 56, p-value = 0.0008428
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
6976.147ms Inf
sample estimates:
mean of the differences
14145.54ms

Figure 17: R output for the t-test of mean CT_EXT and GCSDM runtime.

fastBR but makes a single verification for exclusion with the set of all attributes. On the other hand, fastBR verifies the exclusion for each candidate, which leads to a higher computational cost.

We used a one-sided t-test to evaluate the relative performance of GCSDM over fastBR in low density matrices. We selected the 62 $SDMs$ having a density of 1's lower than 0.3, from our original 83 matrices. As **null hypothesis** we have: *there is no difference between the GCSDM and fastBR runtime for $SDMs$ with density of 1's lower than 0.3.* As **alternative hypothesis** we have: *the runtime of fastBR is higher than the runtime of GCSDM for $SDMs$ with density of 1's lower than 0.3.* The following output from the R software (Figure 20) supports the alternative hypothesis beyond a 95% confidence interval.

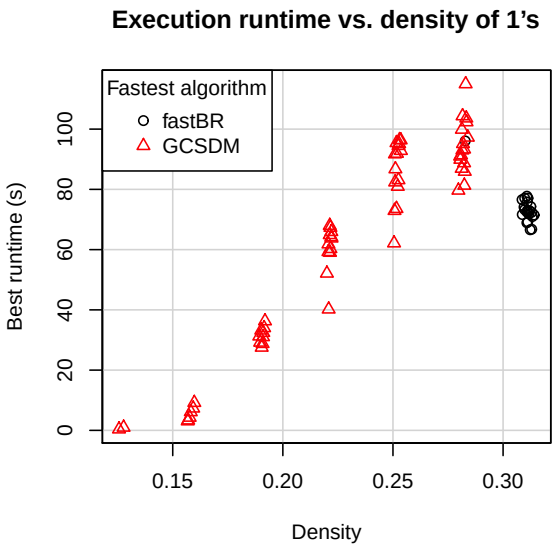

Figure 18: Fastest algorithm runtime vs. density of 1's for all $SDMs$ under study.

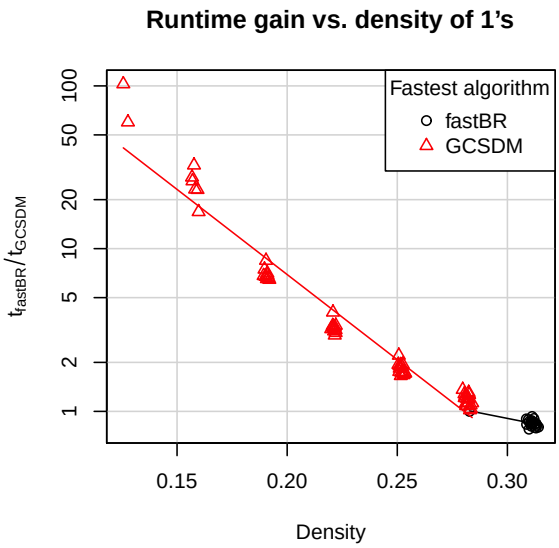

Figure 19: fastBR runtime over GCSDM runtime vs. density of 1's for all $SDMs$ under study.

Paired one-sided t-test

data: fastBR runtime and GCSDM runtime
t = 11.0838, df = 61, p-value $<$ 2.2e-16
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
75124.33ms Inf
sample estimates:
mean of the differences
88453.32ms

Figure 20: R output for the t-test of mean fastBR and GCSDM runtime.

6. Conclusions

This PhD research proposal is focused on the problem of computing all the reducts and its related problem of computing shortest reducts of an information system. These are problems with exponential complexity which are actively studied. The theoretical bases were introduced to provide a unique nomenclature for the document and ensure that it is self contained. We present a revision of the related work to show the most relevant approaches to the problem solution and based on this review we highlight the need for further research in this area. Then, the PhD research proposal is introduced; including justification and motivation, research questions, our research objectives, and the methodology that will guide our research.

As preliminary results, we we developed a hardware module for eliminating all non typical testors on the hardware component; reducing the amount of data that must be transferred to the PC and eliminating the final filtering. This hardware module is applicable to any algorithm for computing typical testors implemented on FPGA devices. This result was published in the memories of an international conference on reconfigurable

computing. Our second preliminary result is a new hardware architecture of the CT EXT algorithm (which is one of the most recent and fastest algorithms reported in the literature) for computing all the typical testors. Our proposal traverses the search space in a different order than previous works, which evaluates less candidate subsets than previous architectures, resulting in shorter runtime. This result has been reported in the journal of Expert Systems With Applications. Additionally, we proposed two new algorithms for computing all the reducts, which are faster than existing algorithms for some kinds of datasets. RSDM was the fastest algorithm for $SDMs$ with density above 0.7 while GCSDM outperforms the rest of the evaluated algorithms in $SDMs$ with density under 0.3. These results will be reported in a congress or a journal specialized in Rough Set Theory.

Throughout our preliminary work, we covered most of the first four points in our proposed methodology, related to algorithms for computing all reducts of an information system. Finally, based on our preliminary results, we conclude that our objectives are reachable, in the scheduled time, following the proposed methodology.

References

Águila, L., & Ruiz-Shulcloper, José. 1984. Algoritmo CC para la elaboración de la información k- valente en problemas de Reconocimiento de Patrones. *Revista Ciencias Matemáticas*, **5**(3).

Alba-Cabrera, Eduardo, Ibarra-Fiallo, Julio, Godoy-Calderon, Salvador, & Cervantes-Alonso, Fernando. 2014. YYC: A Fast Performance Incremental Algorithm for Finding Typical Testors. *Pages 416–423 of: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Springer.

Ayaquica, I O. 1997. Un nuevo algoritmo de escala exterior para el cálculo de testores típicos. *Memorias del II Taller Iberoamericano de Reconocimiento de Patrones*, 141–148.

Bache, Kevin, & Lichman, Moshe. 2013. *UCI machine learning repository*.

Bjorvand, Anders Torvill, & Komorowski, Jan. 1997. Practical applications of genetic algorithms for efficient reduct computation. *Wissenschaft & Technik Verlag*, **4**, 601–606.

Bravo, A. 1983. Algorithm CT for Calculating the Typical Testors of k-valued Matrix. *Revista Ciencias Matematicas*, **4**(2), 123–144.

Cheguis, I. A., & Yablonskii, S. V. 1955. About Testors for Electrical Outlines. *Uspieji Matematicheskij Nauk*, **4**(66), 182–184.

Chen, Yumin, Miao, Duoqian, & Wang, Ruizhi. 2010. A rough set approach to feature selection based on ant colony optimization. *Pattern Recognition Letters*, **31**(3), 226–233.

Chouchoulas, Alexios, & Shen, Qiang. 2001. Rough set-aided keyword reduction for text categorization. *Applied Artificial Intelligence*, **15**(January), 843–873.

Cumplido, René, Carrasco-Ochoa, Jesús Ariel, & Feregrino, Claudia. 2006. On the Design and Implementation of a High Performance Configurable Architecture for Testor Identification LNCS.pdf. *Pages 665–673 of: CIARP 2006*.

Davis, Martin, Logemann, George, & Loveland, Donald. 1962. A machine program for theorem-proving. *Communications of the ACM*, **5**(7), 394–397.

Grze, Tomasz. 2013. FPGA in Rough Set Based Core and Reduct Computation. 263–270.

Haveraaen, Magne, Informatikk, Institutt, & Bergen, Universitetet. 2001. Some Statistical Performance Estimation Techniques for Dynamic Machines. *Norsk Informatikkonferanse*, 176–185.

Jensen, R, & Shen, Q. 2003. Finding rough set reducts with ant colony optimization. *Proceedings of the 2003 UK workshop on*, **1**(2), 15–22.

Jensen, Richard, & Shen, Qiang. 2004. Semantics-preserving dimensionality reduction: Rough and fuzzy-rough-based approaches. *IEEE Transactions on Knowledge and Data Engineering*, **16**(12), 1457–1471.

Jensen, Richard, Tuson, Andrew, & Shen, Qiang. 2014. Finding rough and fuzzy-rough set reducts with SAT. *Information Sciences*, **255**, 100–120.

Jiang, Yu, & Yu, Yang. 2015. Minimal attribute reduction with rough set based on compactness discernibility information tree. *Soft Computing*, 1–11.

Jiao, Na, Miao, Duoqian, & Zhou, Jie. 2010. Two novel feature selection methods based on decomposition and composition. *Expert Systems with Applications*, **37**(12), 7419–7426.

Johnson, David S. 1973. Approximation algorithms for combinatorial problems. *Pages 38–49 of: Proceedings of the fifth annual ACM symposium on Theory of computing*. ACM.

Kopczynski, Maciej, Grze, Tomasz, & Stepaniuk, Jaroslaw. 2014. FPGA in Rough-Granular Computing: Reduct Generation. *Pages 364–370 of: International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*.

Lazo-Cortés, Manuel, Ruiz-Shulcloper, José, & Alba-Cabrera, Eduardo. 2001. An overview of the evolution of the concept of testor. *Pattern Recognition*, **34**(4), 753–762.

Lazo-Cortés, Manuel S, Martínez-Trinidad, José Fco, Carrasco-Ochoa, Jesús A, & Sanchez-Diaz, Guillermo. 2015. On the relation between rough set reducts and typical testors. *Information Sciences*, **294**, 152–163.

Lias-Rodríguez, Alexsey, & Pons-Porrata, Aurora. 2009. BR: A new method for computing all typical testors. *Pages 433–440 of: CIARP 2009*, vol. 5856.

Lias-Rodríguez, Alexsey, & Sanchez-Diaz, Guillermo. 2013. An Algorithm for Computing Typical Testors Based on Elimination of Gaps and Reduction of Columns. *International Journal of Pattern Recognition and Artificial Intelligence*, **27**(08), 1350022.

Lin, Tsau Young, & Yin, Ping. 2004. Heuristically Fast Finding of the Shortest Reducts. *Pages 465–470 of: Rough Sets and Current Trends in Computing*. Springer Berlin Heidelberg.

Martínez, José Francisco, & Guzmán, Adolfo. 2001. The logical combinatorial approach to pattern recognition, an overview through selected works. *Pattern Recognition*, **34**(4), 741–751.

Nguyen, Hung Son, & Skowron, Andrzej. 1997. Boolean Reasoning for Feature Extraction Problems. *Foundations of Intelligent Systems*, 117–126.

Ø hrn, a. 2000. Discernibility and rough sets in medicine: tools and applications. 223.

Parthaláin, Neil Mac, Jensen, Richard, & Shen, Qiang. 2008. Finding fuzzy-rough reducts with fuzzy entropy. *IEEE International Conference on Fuzzy Systems*, 1282–1288.

Pawlak, Z. 1982. Rough sets. *International Journal of Computer and Information Sciences*, 1–51.

Pawlak, Zdislaw. 1981a. *Classification of objects by means of attributes*. Polish Academy of Sciences [PAS]. Institute of Computer Science.

Pawlak, Zdislaw. 1981b. *Rough relations*. Polish Academy of Sciences [PAS]. Institute of Computer Science.

Pawlak, Zdzislaw. 1991. *Rough sets: Theoretical aspects of reasoning about data*. Vol. 9. Springer Science & Business Media.

Pawlak, Zdzisaw, & Skowron, Andrzej. 2007. Rough sets and Boolean reasoning. *Information Sciences*, **177**(1), 41–73.

Piza-Davila, Ivan, Sanchez-Diaz, Guillermo, Aguirre-Salado, Carlos A., & Lazo-Cortes, Manuel S. 2014. A parallel hill-climbing algorithm to generate a subset of irreducible testors. *Applied Intelligence*, **42**(4), 622–641.

Polkowski, Lech, Tsumoto, Shusaku, & Lin, Tsau Y. 2000. *Rough Set Methods and Applications: New Developments in Knowledge Discovery in Information Systems; with 133 Tables;[Adam Mrozek, 1949-1999; this Volume is Dedicated to the Memory of Professor Adam Mrózek]*. Vol. 56. Springer Science & Business Media.

Rodríguez, Vladímir, Martínez, José F, Carrasco, Jesus A, Lazo, Manuel S, Cumplido, René, & Feregrino Uribe, Claudia. 2014. A hardware architecture for filtering irreducible testors. *Pages 1–4 of: ReConFigurable Computing and FPGAs (ReConFig), 2014 International Conference on*. IEEE.

Rojas, A, & Cumplido, R. 2007. FPGA-based architecture for computing testors. *Pages 188–197 of: IDEAL 2007*.

Rojas, Alejandro, Cumplido, René, Ariel Carrasco-Ochoa, J., Feregrino, Claudia, & Francisco Martínez-Trinidad, J. 2012. Hardware-software platform for computing irreducible testors. *Expert Systems with Applications*, **39**(2), 2203–2210.

Ruiz-Shulcloper, José. 2008. Pattern recognition with mixed and incomplete data. *Pattern Recognition and Image Analysis*, **18**(4), 563–576.

Ruiz-Shulcloper, José, Aguila, L., & Bravo, A. 1985. BT and TB algorithms for computing all irreducible testors. *Revista Ciencias Matemáticas*, **2**, 11–18.

Ruiz-Shulcloper, J., Alba-Cabrera, E., Lazo-Cortés, M. 1995. *Introducción a la teoría de Testotes Típicos. Serie Verde No. 50.* México: CINVESTAV-IPN.

Sanchez, Guillermo, & Lazo, Manuel. 2007. CT-EXT: an algorithm for computing typical testor set. *Pages 506–514 of: Progress in Pattern Recognition, Image Analysis and Applications*. Springer.

Sanchez-Díaz, G, Lazo-Cortés, M, & Fuentes-Chávez, O. 1999. Genetic algorithm for calculating typical testors of minimal cost. *Pages 207–213 of: Proc. of the Iberoamerican Symposium on Pattern Recognition (SIARP 1999)*.

Sánchez-Diaz, Guillermo, Piza-Davila, Ivan, Lazo-Cortés, Manuel, Mora-González, Miguel, & Salinas-Luna, Javier. 2010. A fast implementation of the CT-EXT algorithm for the testor property identification. *Pages 92–103 of: LNAI 2010*, vol. 6438.

Santiesteban, Y, & Pons, A. 2003. LEX: a new algorithm for the calculus of typical testors. *Mathematics Sciences Journal*, **21**(1), 85–95.

Skowron, Andrzej, & Rauszer, Cecylia. 1992. The discernibility matrices and functions in information systems. *Pages 331–362 of: Intelligent Decision Support*. Springer.

Starzyk, Janusz, Nelson, D E, & Sturtz, Kirk. 1999. Reduct generation in information systems. *Bulletin of international rough set society*, **3**(May), 19–22.

Starzyk, Janusz a., Nelson, Dale E., & Sturtz, Kirk. 2000. A Mathematical Foundation for Improved Reduct Generation in Information Systems. *Knowledge and Information Systems*, **2**(2), 131–146.

Strakowski, Tomasz, & Rybiski, Henryk. 2008. A New Approach to Distributed Algorithms for Reduct Calculation. *Transactions on Rough Sets IX*, 365–378.

Tiwari, Kanchan, Kothari, Ashwin, & Shah, Riddhi. 2013. FPGA Implementation of a Reduct Generation Algorithm based on Rough Set Theory. *International Journal of Advanced Electrical and Electronics Engineering (IJAEEE)*, **2**(6).

Tiwari, Kanchan Shailendra. 2014. Design and Implementation of Rough Set Algorithms on FPGA : A Survey. **3**(9), 14–23.

Tiwari, K.S., & Kothari, A.G. 2011. Architecture and Implementation of Attribute Reduction Algorithm Using Binary Discernibility Matrix. *Pages 212–216 of: 2011 International Conference on Computational Intelligence and Communication Networks*.

Tiwari, KS, Kothari, AG, & Keskar, AG. 2012. Reduct generation from binary discernibility matrix: an hardware approach. *International Journal of Future Computer and Communication*, **1**(3), 270–272.

Wang, Jue, & Wang, Ju. 2001. Reduction algorithms based on discernibility matrix: the ordered attributes method. *Journal of computer science and technology*, **16**(6), 489–504.

Wang, Pai-Chou. 2007. Highly Scalable Rough Set Reducts Generation. *Journal of Information Science and Engineering*, **4**(23), 1281–1298.

Wang, Xiangyang, Yang, Jie, Teng, Xiaolong, Xia, Weijun, & Jensen, Richard. 2007. Feature selection based on rough sets and particle swarm optimization. *Pattern Recognition Letters*, **28**(4), 459–471.

Wroblewski, Jakub. 1995. Finding minimal reducts using genetic algorithms. *Pages 186–189 of: Proccedings of the second annual join conference on infromation science*.

Wroblewski, Jakub. 1998. A parallel algorithm for knowledge discovery system. *Pages 228–230 of: Proc PARELEC*.

Yang, Ping Yang Ping, Li, Jisheng Li Jisheng, & Huang, Yongxuan Huang Yongxuan. 2008. An Attribute Reduction Algorithm by Rough Set Based on Binary Discernibility Matrix. *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, **2**.

Yao, Yiyu, & Zhao, Yan. 2009. Discernibility matrix simplification for constructing attribute reducts. *Information Sciences*, **179**(7), 867–882.

Zheng, Kai, Hu, Jie, Zhan, Zhenfei, Ma, Jin, & Qi, Jin. 2014. An enhancement for heuristic attribute reduction algorithm in rough set. *Expert Systems with Applications*, **41**(15), 6748–6754.

Zhong, Ning, Dong, Juzhen, & Ohsuga, Setsuo. 2001. Using rough sets with heuristics for feature selection. *Journal of intelligent information systems*, **16**(3), 199–214.