

An Innovative Approach for Attribute Reduction in Rough Set Theory

Alex Sandro Aguiar Pessoa, Stephan Stephany

National Institute for Space Research (INPE), Sao Jose dos Campos, Brazil
Email: asapessoa@gmail.com, stephan.stephany@lac.inpe.br

Received 4 July 2014; revised 3 August 2014; accepted 25 August 2014

Copyright © 2014 by authors and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY).
<http://creativecommons.org/licenses/by/4.0/>



Open Access

Abstract

The Rough Sets Theory is used in data mining with emphasis on the treatment of uncertain or vague information. In the case of classification, this theory implicitly calculates reducts of the full set of attributes, eliminating those that are redundant or meaningless. Such reducts may even serve as input to other classifiers other than Rough Sets. The typical high dimensionality of current databases precludes the use of greedy methods to find optimal or suboptimal reducts in the search space and requires the use of stochastic methods. In this context, the calculation of reducts is typically performed by a genetic algorithm, but other metaheuristics have been proposed with better performance. This work proposes the innovative use of two known metaheuristics for this calculation, the Variable Neighborhood Search, the Variable Neighborhood Descent, besides a third heuristic called Decrescent Cardinality Search. The last one is a new heuristic specifically proposed for reduct calculation. Considering some databases commonly found in the literature of the area, the reducts that have been obtained present lower cardinality, *i.e.*, a lower number of attributes.

Keywords

Rough Set Theory, Reducts, Attribute Reduction, Metaheuristics

1. Introduction

Large amounts of data are generated everyday and the ability to analyze them is normally a challenge. Experts need efficient data mining methods to extract useful information and to perform the analysis of the data. This is the case of the Rough Sets Theory (RST), emerging in the early 80s [1] to deal with uncertain, incomplete or vague information. In addition, RST has a good mathematical formalism, and it is easy to use, since it does not require additional information, such as the probability distribution, *a priori* probability or pertinence degree.

These strong aspects of RST have helped to spread its use over the last years. A weak aspect of RST is the non availability of free RST software, except for limited implementations. On the other hand, there is RST proprietary software.

RST is an extension of the set theory and has the implicit feature of compressing the dataset. Such compression is due to the definition of equivalence classes based on indiscernibility relations and to the elimination of redundant or meaningless attributes. A central concept in RST is attribute reduction, which generates reducts. A reduct is any subset of attributes that preserves the indiscernibility of the elements for the considered classes, allowing performing the same classification that would be obtained with the full set of attributes. This work focuses on the calculation of reducts in RST, which is a NP-hard problem [2]. A brief outline of RST follows in Section 2, while attribute reduction for RST is approached in Section 3.

It is common to employ the RST as a way of preprocessing to obtain reducts for other machine learning techniques. The typical high dimensionality of current databases precludes the use of greedy methods imbedded in RST to find optimal or suboptimal reducts in the search space, requiring the use of stochastic methods. This issue is tackled by the use of metaheuristics, such as genetic algorithms to calculate reducts [3]. In this context, attribute reduction was approached by [4] which proposed some metaheuristics, such as Ant Colony and Simulated Annealing for reduct calculation in RST, besides a Genetic Algorithm. Results are shown for some known databases intended for classification. Tabu Search was also proposed in [5] and applied for the same databases. These former results can be taken as benchmarks. The present work evaluates alternative metaheuristics for reduct calculation in RST, being applied for the classification of the same databases.

The alternative metaheuristics proposed here for reduct calculation are Variable Neighborhood Search (VNS) and Variable Neighborhood Decent (VND), and a new one (actually a heuristic) called Decrescent Cardinality Search (DCS) [6]. VNS and VND are well known [7]. VNS generates a random candidate solution (or reduct) with any cardinality and at every iteration explores its neighborhood searching for new solutions that are then evaluated. VND searches for solutions following a deterministic approach. It is an extension of VNS. On the other hand, DCS is a modified version of VNS that explores the search space aiming only at a lower cardinality reduct. It enforces randomly a cardinality reduction at each iteration before performing a local search. Section 4 describes VNS, VND and DCS. Local search was performed by a standard scheme or using VND itself. For the sake of simplicity, DCS is also denoted as a metaheuristic in the rest of this work.

In the scope of attribute reduction in RST applied to classification, any metaheuristic requires an evaluation function in order to rank the candidate solutions/reducts. Two different functions were evaluated in this work for the calculation of reducts in RST: the dependency degree between attributes and the relative dependency [8]. These functions, presented in Section 3.2, allows simplifying the calculation of the reducts in comparison with the standard approach based on the discernibility matrix, which presents high computational complexity, not being feasible for large and complex datasets. As mentioned above, three alternative metaheuristics are evaluated for the calculation of reducts in RST: VNS, VND and DCS, which are described in Sections 4.1, 4.2 and 4.3, respectively. In addition, two local search schemes are also evaluated. Test results for some well-known databases are shown in Section 5. Processing time is also discussed. Each test case is related to the use of a specific metaheuristic for the calculation of reducts applied to the classification of a specific database.

2. Rough Set Theory

Real world informations are often uncertain, vague or incomplete due to difficulties associated to record or report any natural phenomena or events that are under study. Some approaches are well known to tackle such issues, mainly the Fuzzy Set theory [9], the Dempster-Shafer theory [10] [11], and the possibility theory [12]. In the beginning of the eighties, another theory emerged for treating such kind of information, the Rough Set Theory-RST [1]. This theory is simple and has a good mathematical formalism. It is an extension of the set theory that deals with data uncertainty by means of an equivalence relation known as indiscernibility. Two elements of a given set are considered as indiscernible if they present the same properties, according to a defined set of features, attributes or variables. Some authors like [13] point out that the main advantage of RST is not requiring additional informations such as the probability distribution, a priori probability or pertinence degree.

In traditional set theory, crisp sets are used, *i.e.* the membership function, describing the belongingness of elements of the universe to the set, can attain one of the two values, 0 or 1. Fuzzy set is defined by the membership function which can attain values from the closed interval $[0, 1]$, allowing partial membership of the ele-

ments in the set. On the other hand, in the RST, membership is not the primary concept. Rough sets represent a different mathematical approach to vagueness and uncertainty. Definition of a set in the RST is related to our information (knowledge) and perception about elements of the universe. It is based on the concept of indiscernibility: different elements can be discernible/indiscernible based on the information about them. Indiscernibility is defined as an equivalence relation that allows partitioning the universe into regions that express the classes that compose this universe.

Although RST was proposed to deal with inconsistencies of information in the considered dataset, one of its main issues is to compact the dataset. This is achieved by means of eliminating elements with equal characteristics that are indiscernible in the sense that a unique element represents the others. A further way to compact the database is given by the reducts that eliminate redundant or meaningless attributes (superfluous attributes). A reduct is a subset of attributes that provides a classification with a minimum cardinality (minimum number of attributes) but still preserving the equivalence classes obtained with the full set of attributes. The concepts of indiscernibility and reducts help to reduce the volume of data [2]. In order to explain such basic concepts of RST in the next sections, an example dataset is shown in **Table 1**.

2.1. Indiscernibility Relations

In the RST, data are stored in a tabular format forming a pair $S = (U; A)$ called an *information system*, where U is the non-empty set of elements/objects called *universe* and A is a finite set of conditional attributes such that $a: U \rightarrow V_a$ for all $a \in A$. The set V_a contains the possible values of attribute a . The elements of A are called *conditional attributes* or simply *conditions*. In order to perform a supervised learning, the addition of a decision attribute is required. The resulting information system is $S = (U; A \cup \{d\})$, where $d \notin A$ is the *decision attribute*. For the example shown in the **Table 1**, the universe is $U = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, the set of conditional attributes is $A = \{a_1, a_2, a_3\}$ and the decision attribute is d . Possible values for the conditionals attributes values are: $V_{a_1} = \{0, 1, 2\}$ and $V_{a_2} = V_{a_3} = \{0, 1\}$. The decision attribute d allows definition of classes $X_1, \dots, X_{|V_d|}$, where $|V_d|$, denotes the cardinality of its domain. In the example dataset, $|V_d| = 2$, since $V_d = \{0, 1\}$. The corresponding classes are X_1 and X_2 , the first composed of elements with $d = 0$, and the second, of elements with $d = 1$.

RST provides the definition of an equivalence relation, which groups together elements that are indiscernible into classes, providing a partition of the universe of elements. Given an information system $S = (U; A)$, and a subset of attributes $B \subseteq A$, an equivalence relation $IND_A(B)$ is defined as follows (omitting the subscript A , it can be simplified to $IND(B)$):

$$IND(B) = \{(x, x') \in U \mid \forall a \in B, a(x) = a(x')\} \tag{1}$$

This equivalence relation is called *B-indiscernibility relation*. The equivalence classes of this relation are denoted by $[x]_B$ [2]. The partitions induced by $IND(B)$ on the universe U are represented as $U/IND(B)$ or in a short form, U/B . Consequently, an equivalence relation groups together elements that are indiscernible into classes, providing a partition of the universe of elements. In the example dataset, the partitions generated by such indiscernibility relation are:

- 1) $U/IND(a_1) = U/\{a_1\} = U/\{a_1, a_2\} = \{\{x_1, x_4, x_5\}, \{x_2, x_6\}, \{x_3\}\}$
- 2) $U/\{a_2\} = \{\{x_1, x_3, x_4, x_5\}, \{x_2, x_6\}\}$

Table 1. An example dataset with three conditional attributes and one decision attribute.

	a_1	a_2	a_3	d
x_1	2	0	0	0
x_2	0	1	0	1
x_3	1	0	1	0
x_4	2	0	0	1
x_5	2	0	1	1
x_6	0	1	1	1

- 3) $U/\{a_3\} = \{\{x_1, x_2, x_4\}, \{x_3, x_5, x_6\}\}$
- 4) $U/\{a_2, a_3\} = \{\{x_1, x_4\}, \{x_2\}, \{x_3, x_5\}, \{x_6\}\}$
- 5) $U/\{a_1, a_3\} = U/\{A\} = \{\{x_1, x_4\}, \{x_2\}, \{x_3\}, \{x_5\}, \{x_6\}\}$

In the RST, these partitions based on an indiscernibility relation are called approximations (Figure 1) and allow defining, for each class, four regions: lower approximation, upper approximations, boundary and outside regions. The lower approximation is composed of elements that certainly belong to the considered class, while the upper is composed of elements that possibly belong to it. The difference between the upper and the lower approximations is the boundary region, while the outside region is composed of the elements that surely do not belong to the considered class. In the case of an empty boundary region, the dataset is called crisp, as in the traditional data set theory. Otherwise, the dataset is said to be rough.

The outside and boundary regions can be obtained from the lower and upper approximations *B-lower* and *B-upper* for a considered class X , denoted respectively by \underline{BX} and \overline{BX} :

$$\underline{BX} = \{x \mid [x]_B \subseteq X\} \tag{2}$$

$$\overline{BX} = \{x \mid [x]_B \cap X \neq \emptyset\} \tag{3}$$

Elements contained in \underline{BX} surely belong to class X , while the elements in \overline{BX} can only be classified as possibly belonging to class X . The set $F_B = \overline{BX} - \underline{BX}$ is the boundary region of X , and its elements cannot decisively be assigned to class X . Finally, the set $E_B = U - \overline{BX}$ is the outside region of class X , containing elements that surely do not belong to this class.

Considering $IND(\{a_1, a_2, a_3\})$ for the example dataset, with its two classes $X_1 = \{x_1, x_3\}$ and $X_2 = \{x_2, x_4, x_5, x_6\}$, the corresponding lower approximation (\underline{BX}), upper approximation (\overline{BX}), outside region (E) and boundary region (F) are:

$$\begin{aligned} \underline{BX}_1 &= \{x_3\}; & \underline{BX}_2 &= \{\{x_2\}, \{x_5\}, \{x_6\}\}; \\ \overline{BX}_1 &= \{\{x_1, x_4\}, \{x_3\}\}; & \overline{BX}_2 &= \{\{x_2\}, \{x_1, x_4\}, \{x_5\}, \{x_6\}\}; \\ F(X_1) &= \overline{BX}_1 - \underline{BX}_1 = \{x_1, x_4\}; & F(X_2) &= \overline{BX}_2 - \underline{BX}_2 = \{x_1, x_4\}; \\ F(X_1) &= U - \overline{BX}_1 = \{\{x_2\}, \{x_5\}, \{x_6\}\}. & F(X_2) &= U - \overline{BX}_2 = \{x_3\}. \end{aligned}$$

Another RST definition is the *positive region* for $IND(B)$, denoted by $POS_B(d)$, that is the union of the lower

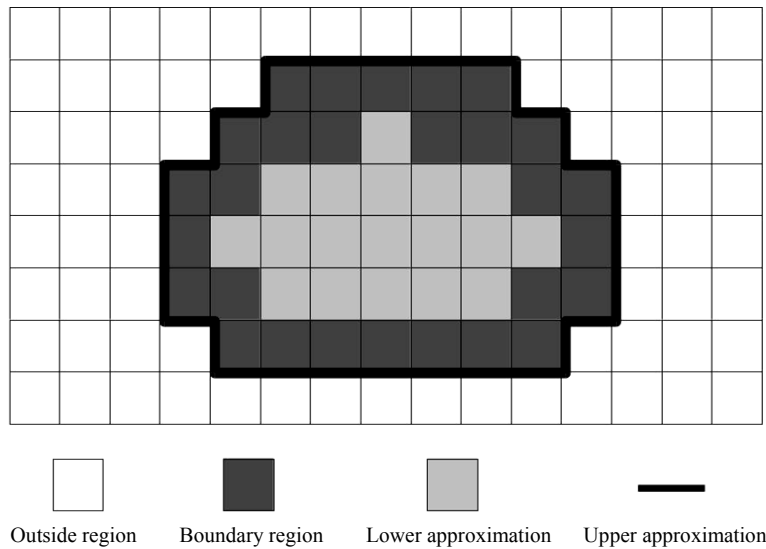


Figure 1. Approximation regions for a given class.

approximations of all classes, *i.e.* $\underline{B}X_1 \cup \dots \cup \underline{B}X_{|V_d|}$.

2.2. Reducts

In RST, the size of the dataset can be reduced either by representing a whole class given by an indiscernibility relation (by means of a single element of this class) or by eliminating superfluous attributes that do not contribute to the classification. Given the original set A , a set B is called a *reduct* when $IND_A(B) = IND_A(A)$, with $B \subseteq A$. Then, if $|B| < |A|$, where $|\cdot|$ denotes the cardinality of the set, this implies in an attribute reduction, but preserves the information contained in the data that is relevant for the classification. The total number reducts of a decision system with m attributes is given by all possible combinations of attributes, as given by [2]:

$$\binom{m}{\lfloor \frac{m}{2} \rfloor} \tag{4}$$

where the term $\lfloor m/2 \rfloor$ denotes the *floor*, the largest integer that is less or equal to $m/2$.

The calculation of reducts is a non-trivial task as reported by [2]. The simple increase of computational power does not allow the calculation of reducts for complex databases, since it constitutes a NP-hard problem for large and complex databases. This is a major performance bottleneck of the RST. **Table 2** shows the number of possible reducts for an m -attribute database according to Equation (4).

3. Attribute Reduction in Rough Set-Theory

Attribute reduction in RST implies in the calculation of the reducts. The classic approach is to obtain the discernibility matrix, to determine its correspondent discernibility function and to simplify it, in order to get the set of reducts. However, this approach is very expensive computationally, mainly for complex datasets. Another approach that circumvents this problem is to formulate the calculation of reducts as an optimization problem that is iteratively solved by some metaheuristic. It requires the use of an evaluation or objective function in order to rank the reducts to choose the best candidate solution of the current iteration. Both approaches are detailed in the next sections.

3.1. Discernibility Matrix-Based Approaches

In these approaches, the calculation of reducts requires the discernibility matrix. For given information system S with n elements has a symmetric discernibility matrix with dimension $n \times n$. The entries of this matrix are denoted by c_{ij} , for $i \neq j$. Each entry contains the subset of attributes that distinguishes element x_i from x_j , being the diagonal entries null, according the definition below. The discernibility matrix for the example dataset is appears in **Table 3**.

$$c_{ij} = \{a \in A \mid a(x_i) \neq a(x_j)\} \text{ for } i, j = 1, \dots, n \text{ and } i \neq j \tag{5}$$

The corresponding discernibility function f_A is a Boolean function of m attributes (a_1, a_2, \dots, a_m) , given by:

$$f_A(a_1^*, \dots, a_n^*) = \bigwedge \{ \bigvee c_{ij}^* \mid 1 \leq j < i \leq n, c_{ij} \neq \emptyset \} \tag{6}$$

Table 2. Number of possible reducts for a database with m attributes.

m	# Possible Reducts
10	252
20	184,756
30	155,117,520
40	137,846,528,820
50	126,410,606,437,752

Table 3. Discernibility matrix for the example dataset.

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	-					
x_2	a_1, a_2	-				
x_3	a_1, a_3	a_1, a_2, a_3	-			
x_4	\emptyset	a_1, a_2	a_1, a_3	-		
x_5	a_3	a_1, a_2, a_3	a_1	a_3	-	
x_6	a_1, a_2, a_3	a_3	a_1, a_2	a_1, a_2, a_3	a_1, a_2	-

where $c_{ij}^* = \{a^* \mid a \in c_{ij}\}$.

The Boolean simplification of f_A yields the set of reducts of A , as follows. For the example dataset, the application of Equation (6) gives its discernibility function:

$$f(x) = [(a_1 \vee a_2) \wedge (a_1 \vee a_3) \wedge (a_3) \wedge (a_1 \vee a_2 \vee a_3)] \wedge [(a_1 \vee a_2 \vee a_3) \wedge (a_1 \vee a_2) \wedge (a_1 \vee a_2 \vee a_3) \wedge (a_3)] \\ \wedge [(a_1 \vee a_3) \wedge (a_1) \wedge (a_1 \vee a_2)] \wedge [(a_3) \wedge (a_1 \vee a_2 \vee a_3)] \wedge [(a_1 \vee a_2)]$$

Note that this logical function is expressed by the conjunction of many terms, with each one corresponding to a column of the related discernibility matrix. The Boolean simplification of this function yields then a single reduct:

$$f(x) = a_1 \wedge a_3$$

Therefore, this reduct is composed of attributes a_1 and a_3 , and the original dataset shown in **Table 1** can be represented in a shorter form without loss of information by a new table (**Table 4**).

3.2. Dependency Function-Based Approaches

In these approaches, a metaheuristic is employed to iteratively refine candidate solutions (reducts) that are ranked by an evaluation or objective function. A common choice for this function is the dependency function, employed in the Rough Set Attribute Reduction method (RSAR), exposed in Section 3.2.1. It measures the redundant information or dependency between a subset of conditional attributes and the decision attribute. If such a subset has a dependency degree equal or higher than the one of the complete set of attributes, then it is a possible solution, *i.e.* it constitutes a reduct. However, it still presents high computational complexity, since it demands the calculation of the discernibility function and of the positive region for the considered solution. Alternatively, the concept of relative dependency, presented in Section 3.2.2, provides an evaluation function with lower computational cost than the dependency degree. This work employs both evaluation functions for any of the proposed metaheuristics that were tested (VNS, VND and DCS).

3.2.1. Rough Sets Attribute Reduction (RSAR)

A most common approach for the calculation of reducts in RST is the Rough Set Attribute Reduction method (RSAR), that employs as evaluation function the dependency degree of rough sets [14] [15]. The dependency degree, defined in Equation (7), evaluates if a subset of attributes has redundant information in respect to another subset of attributes. Considering two subsets of attributes, a subset B to be evaluated and a subset given by the decision attribute itself ($\{d\}$), their corresponding indiscernibility relations in U are $IND(B)$ and $IND(\{d\})$, respectively.

The dependency degree between B and $\{d\}$ is then defined as:

$$\gamma_B(\{d\}) = \frac{|POS_B(\{d\})|}{|U|} \tag{7}$$

where $POS_B(\{d\}) = \cup_{\Theta} B\Theta$, with $\Theta = U/IND(\{d\})$, is the *positive region*, $|POS_B(\{d\})|$ and $|U|$, denote, respectively, the cardinality of this region and of the universe. If $\gamma_B(\{d\}) = 1$ then $\{d\}$ depends totally on B . If

Table 4. A reduced dataset considering a reduct with attributes a_1 and a_3 .

	a_1	a_3	d
x_1	2	0	0
x_2	0	0	1
x_3	1	1	0
x_4	2	0	1
x_5	2	1	1
x_6	0	1	1

$\gamma_B(\{d\}) < 1$ then $\{d\}$ depends partially on B . Finally, if $\gamma_B(\{d\}) = 0$ then B does not depend on $\{d\}$.

The dependence degree between subsets of attributes minimizes inconsistencies in the information system, since it ranks a subset according to the redundancy of its attributes. The dependency degree between a subset and the decision attribute, shown above, allows calculating reducts. Considering the complete set of attributes A , any subset B such that $\gamma_B(\{d\}) \geq \gamma_A(\{d\})$ is a reduct. This can be proved as follows.

If B is a subset of the complete set of attributes A , then $B \subseteq A$, and $\{d\}$ the set containing the decision attribute, then the set R of all possible reductions is given by:

$$R = \{B : B \subseteq A \mid \gamma_B(\{d\}) \geq \gamma_A(\{d\})\} \quad (8)$$

In the example dataset, $d = 0$ determines class $X_1 = \{x_1, x_4\}$, and $d = 1$, class $X_2 = \{x_2, x_3, x_5, x_6\}$. Equation (7) yields the dependency degree of the full set of attributes $A = \{a_1, a_2, a_3\}$:

$$\gamma_A(\{d\}) = \frac{|\{x_2, x_3, x_5, x_6\}|}{|U|} = \frac{4}{6}$$

Analogously, the dependency degrees of the subsets of A are calculated:

$$\begin{aligned} \gamma_{\{a_1\}}(\{d\}) &= \frac{3}{6}, \quad \gamma_{\{a_2\}}(\{d\}) = \frac{2}{6}, \quad \gamma_{\{a_3\}}(\{d\}) = 0, \\ \gamma_{\{a_1, a_2\}}(\{d\}) &= \frac{3}{6}, \quad \gamma_{\{a_1, a_3\}}(\{d\}) = \frac{4}{6}, \quad \gamma_{\{a_2, a_3\}}(\{d\}) = \frac{2}{6} \end{aligned}$$

Finally, subset $\{a_1, a_3\}$ can be considered reduct, since it has dependency degree equal to A .

3.2.2. Attribute Reduction Based on Relative Attribute Dependency

A less computationally expensive way to evaluate a candidate solution (*i.e.* a reduct) is by means of an alternative metric called *relative dependency*, proposed by [8] in the scope of the RST. The relative attribute dependency degree can be calculated by the ratio of the number of partitions of the universe for a considered subset of conditional attributes and the number of partitions for the union of such subset with the decision attribute. Given A , the set of all conditional attributes, $B \subseteq A$ and $\{d\}$ the decision attribute, the relative dependency is given by:

$$\kappa_B(\{d\}) = \frac{|U/IND(B)|}{|U/IND(B \cup \{d\})|} \quad (9)$$

Employing the relative dependency κ_B instead of the dependency degree γ_B in Equation (8) also yields the set of reducts:

$$R = \{B : B \subseteq A \mid \kappa_B(\{d\}) \geq \kappa_A(\{d\})\} \quad (10)$$

For the example dataset shown in **Table 1**, according to the decision attribute d , the universe is partitioned into subsets $X_1 = \{x_1, x_4\}$ and $X_2 = \{x_2, x_3, x_5, x_6\}$. Considering the full set of attributes $A = \{a_1, a_2, a_3\}$ it is possible to calculate its $\kappa_A(\{d\})$ by Equation (10):

$$\kappa_A(\{d\}) = \frac{|\{x_1, x_4, \{x_2\}, \{x_3\}, \{x_5\}, \{x_6\}\}|}{|\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}, \{x_6\}|} = \frac{5}{6}$$

Repeating the calculation of κ_B for every subset of A results in:

$$\begin{aligned} \kappa_{\{a_1\}}(\{d\}) &= \frac{3}{4}, \quad \kappa_{\{a_2\}}(\{d\}) = \frac{2}{3}, \quad \kappa_{\{a_3\}}(\{d\}) = \frac{2}{4}, \\ \kappa_{\{a_1, a_2\}}(\{d\}) &= \frac{3}{4}, \quad \kappa_{\{a_1, a_3\}}(\{d\}) = \frac{5}{6}, \quad \kappa_{\{a_2, a_3\}}(\{d\}) = \frac{4}{6} \end{aligned}$$

In the same way as in the RSAR approach, only the subset $\{a_1, a_3\}$ that have a relative dependency greater or equal as $\kappa_A(\{d\})$ are considered as a reduct. Therefore, the only reduct is $R = \{a_1, a_3\}$.

4. Metaheuristics Applied to the Calculation of Reducts in RST

Metaheuristics are computational methods employed when a given problem is formulated as an optimization problem to be solved iteratively. Many of them are inspired in some natural behavior. At each iteration, a candidate solution is generated and evaluated by an objective function. It is expected to improve the quality of the solutions along the iterations. Typically, metaheuristics are stochastic, being applied to NP-hard problems or problems without a known solution due to their ability to explore huge search spaces. Another issue is their ability to escape from local minima, in comparison to deterministic methods. However, since there is no guarantee of reaching an optimal solution, it is expected to reach a suboptimal solution, considered as “good” according to its evaluation. Early stochastic approaches were proposed in the fifties, and some cornerstones were given by Genetic Algorithms [7] [16] [17], Simulated Annealing [18], Tabu Search [19], or Ant Colony [20].

This work is about attribute selection in RST, as exposed in Section 3.2, employing a given metaheuristic with a suitable evaluation function. The innovative use of three metaheuristics (VNS, VND and DCS) is proposed for the calculation of reducts in RST. These metaheuristics are described in Sections 4.1, 4.2 and 4.3, and were tested for some well-known datasets as shown in Section 5. The two evaluation functions to rank the candidate solutions (reducts) were already presented in Section 3 (dependency degree or relative dependency).

However, any metaheuristic demands a representation for these solutions. Since each reduct is composed of a subset of attributes, it was chosen to represent a solution as a string of zeros and ones, where a zero represents the absence of a given attribute, and a one, its presence in the subset. The elements of the string may be called bits, while the size of the string is the cardinality of the solution. For instance, if the complete set of conditional attributes is given by $A = \{a_1, a_2, a_3\}$, then the string $A^* = \{0, 1, 1\}$ represents a solution that contains only the attributes a_2 and a_3 .

Since all of the proposed metaheuristics perform explorations of the search space around a given candidate solution, it is necessary to define its corresponding neighborhood. In this work, a neighbor solution is obtained by switching any bit of the string (from zero to one or vice-versa). For instance, assuming that is only allowed to switch the value of a single bit, (anyone of the string) in our example string, the choice of the first bit of the string results in the neighbor solution $A^* = \{1, 1, 1\}$ (attributes a_1, a_2 and a_3 are considered).

Considering the operation of bit switching in a string of size $|A|$, a *neighborhood structure* N_k is defined by the set of solutions generated from a particular solution s by randomly switching $k \leq |A|$ bits of the string. Therefore, the size of such neighborhood is given by $C_k^{|A|}$. For example, assuming the switching of a single bit, corresponding to a neighborhood structure of order 1, the solution $s = \{0, 1, 1\}$ with $k = 1$ and $|A| = |s| = 3$, will present neighbors $(N_1(s)) \{1, 1, 1\}, \{0, 0, 1\}$ and $\{0, 1, 0\}$, matching $C_1^3 = 3$.

Such neighborhood structures define not only the neighbor solutions, but also the Hamming distance from the original solution s , that is the number of positions in the string where a bit differs between s and its neighbors. Obviously, if $k = 1$ the Hamming distance is 1 and so on. In the remaining of this text, the cardinality of the neighborhood structure of a given solution, that is equal to the maximum Hamming distance between this solution and a neighbor solution, is denoted by L .

4.1. Local Search Schemes

Two of the proposed metaheuristics, VNS, and DCS demand a local search scheme, while VND can be considered itself a local search scheme, employed in this work with Hamming distance of neighborhood structure re-

stricted to 2. VNS and DCS were implemented using a standard local search algorithm or using VND as an alternative local search scheme [7]. These two local search schemes (standard local and VND) correspond to hill climbing algorithms that perform a local ascent search. VNS and DCS variants are denoted in this work as VNS-s and DCS-s when employing the standard local search or as VNS-v and DCS-v when employing VND as local search scheme.

The standard local search algorithm explores the neighborhood of the current candidate solution s , generating many solutions s' that are evaluated and compared to the current solution. This algorithm, returns a solution s_{local} , that corresponds to a neighbor solution that is better than the current solution s or, if it is not, returns s itself. The pseudocode of such local search is shown below for the adopted case, $k = 1$, that allows the switching of one bit at a time (one-bit local search), considering an evaluation function $f(\cdot)$:

Algorithm 1. Local Search pseudocode with $k=1$.

Considering a given candidate solution s :

```

begin local search
   $s_{local} = s$ 
  for  $i = 1$  to  $I \leq |A|$  do
     $s' \leftarrow$  switch the  $i$ -th bit of string  $s$ 
    if  $f(s') > f(s)$  then
       $s_{local} \leftarrow s'$ 
    end if
  end for
returns  $s_{local}$ 
end local search

```

4.2. Variable Neighborhood Search (VNS)

The Variable Neighborhood Search metaheuristic is an iterative procedure that systematically explores neighborhoods using as increasing structure cardinality by randomly generating new neighbors (s') from the initial/current candidate solution (s). A new neighbor s' is generated by switching k bits randomly (initially, $k = 1$), and then a standard local search is performed in its neighborhood structure trying to find a better solution s'' . If it succeeds, then a new neighbor s'' of s' is generated followed by a new standard local search. If it does not succeed, k is increased ($k = k + 1$) in order to generate a new neighbor in a neighborhood with increased structure cardinality. The best solution found is assumed as the current solution for the next iteration, and the procedure stops when a limit iteration number is reached [7] [21]. Some key assumptions of this approach are: 1) a local optimum for a given neighborhood is not necessarily a local optimum for another neighborhood; 2) a global optimum is also a local optimum for all neighborhoods; and 3) local optima for the same neighborhood are very near, in most problems. Here, the standard local search is the one-bit search described above, k is limited by the upper bound $L \leq |A|$, and the VNS pseudocode follows, considering an evaluation function $f(\cdot)$:

Algorithm 2. VNS pseudocode.

Considering a randomly generated initial candidate solution s :

```

begin VNS
while (limit number of iterations not reached) do
   $k \leftarrow 1$ ;
  while ( $k \leq L$ ) do
     $s' \leftarrow$  randomly generate a neighbor  $s' \in N_k(s)$ 
     $s'' \leftarrow$  perform the one-bit local search for  $s'$ ;
    if  $f(s'') > f(s)$  then
       $s \leftarrow s''$ 
       $k \leftarrow 1$ 
    else
       $k \leftarrow k + 1$ 
    end if
  end while
end while
returns  $s$ 
end VNS

```

4.3. Variable Neighborhood Descent (VND)

The Variable Neighborhood Descent metaheuristic is a deterministic variant of VNS, being commonly employed as a local search method. VND returns the best neighbor s' of the initial candidate solution (s), instead of generating a random neighbor [21]. It starts exploring the neighborhood $N_k(s)$, for $k = 1$, of the initial solution and then neighborhoods with increasing structure cardinality. Therefore, it guarantees to find the best neighbor of the initial solution if $L \leq |A|$. Its pseudocode follows, considering an evaluation function $f(\cdot)$:

Algorithm 3. VND pseudocode.

Considering a randomly generated initial candidate solution s :

```

begin VND
   $k \leftarrow 1$ ;
  while ( $k \leq L$ ) do
     $s' \leftarrow$  apply  $k$ -bit local search from  $s \in N_k(s)$ 
    if  $f(s') > f(s)$  then
       $s = s'$ 
       $k = 1$ 
    else
       $k = k + 1$ 
    end if
  end while
  return  $s$ 
end VND

```

4.4. Decrescent Cardinality Search (DCS)

The Decrescent Cardinality Search is a new heuristic proposed in this work, derived of VNS. DCS is an iterative procedure that randomly generates a neighbor solution (s') with lower cardinality (in this case one less) in the neighborhood $N_{|A|}(s)$ of the initial/current candidate solution (s) and performs the one-bit local search in s' trying to find a better solution s'' . The best solution found is assumed as the current solution for the next iteration. Iterations proceed until a stopping criterion is met or a limit number of iterations is reached. Its pseudocode follows, considering an evaluation function $f(\cdot)$:

Algorithm 4. DCS pseudocode.

Considering a randomly generated initial candidate solution s :

```

begin DCS
  while (limit number of iterations not reached) do
     $s' \leftarrow$  randomly generate a neighbor  $s' \in N_{|A|}(s)$  and  $|s'| = |s| - 1$ 
     $s'' \leftarrow$  perform a one-bit local search for  $s'$ ;
    if  $f(s'') > f(s)$  then
       $s \leftarrow s''$ 
    end if
  end while
  return  $s$ 
end DCS

```

5. Results

This work proposes the use of the three metaheuristics (VNS, VND and DCS) for the calculation of reducts in RST using 13 datasets. These datasets were employed in [5] and [15] to test four other metaheuristics: Ant Colony Optimization, Simulated Annealing, Genetic Algorithm, and Tabu Search, denoted here by ACO, SA, GA and TS, respectively.

Each test case is related to the use of: 1) a specific metaheuristic (VNS, VND or DCS) for the calculation of reducts; with 2) a given local search scheme (if applicable); 3) a given evaluation function (κ or γ); 4) a Hamming distance (if applicable); and 5) the choice of a specific database.

The proposed metaheuristics were executed 20 times for each of the 13 datasets, for each evaluation function (degree of dependency γ or relative dependency κ), and for each scheme of local search (standard or VND local

search, represented by s and v , respectively), yielding a total of 5720 tests. Considering the choice of evaluation function and local search scheme results, there are 10 possible variants of the proposed metaheuristics or heuristic: $VNS(\gamma)$ - s , $VNS(\kappa)$ - s , $VNS(\gamma)$ - v , $VNS(\kappa)$ - v , $DCS(\gamma)$ - s , $DCS(\kappa)$ - s , $DCS(\gamma)$ - v , $DCS(\kappa)$ - v , $VND(\gamma)$, and $VND(\kappa)$. Any of these variants start from a randomly generated solution. VNS and VND variants were executed using maximum Hamming distance values $L = 4, 8$ and 16 , denoted by L_4, L_8 and L_{16} , respectively (not applicable for DCS variants).

Execution of VNS variants was limited to 10 iterations, and executions of DCS , to 30 iterations (not applicable to VND). These limits were chosen empirically based on the average number of iterations that algorithms VNS and DCS spend to achieve convergence. Such number is typically less than 10 for VNS , and less than 30 for DCS . Each iteration of VNS is slower, since it performs a local search in the neighborhood of the current solution, while iterations of DCS are faster, since it uses a more aggressive search scheme, looking only for lower cardinality solutions.

Table 5 denotes by $|A|$ the cardinality of the conditional attribute set, by $|U|$ the number of elements for each dataset and by $|V_d|$ the number of classes of the dataset.

All these datasets were downloaded from the UCI Machine Learning Repository [22]. The 5720 tests were executed in a PC with a quadcore 3.4 GHz processor and 16 GB of main memory. All metaheuristics were coded in Perl (*Practical Extraction and Report Language*) using object-oriented programming. The datasets were stored in the Database Management System (DBMS). Perl is suitable for the proposed application, since it is 1) an interpreted language, 2) it is portable, 3) deals easily with strings, and 4) provides text manipulation and pattern matching. DBMS makes easy the storing and retrieval of data and provides evaluation functions in the SQL (Structured Query language).

The benchmarks results, taken as a reference, are those found in [5] and [15]. These results refer to the use of the metaheuristics ACO, SA, GA and TS to perform the calculation of reducts in RST aiming at classification for the set of databases described in **Table 5**. These results were obtained employing only the degree of dependency (γ) as evaluation function.

The new results obtained by the application of the proposed metaheuristics to these databases are shown in the following tables, except for VND , shown in a table that summarizes the results, for convenience. VND seems more suitable to be used as a local search scheme, as discussed in [21]. However, VND is also employed as a local search scheme instead of the standard local search scheme. It is worth to note that both VNS and DCS were evaluated using as local search scheme either standard one or VND itself. In addition, VNS and DCS were

Table 5. Datasets employed in the tests.

Datasets	$ A $	$ U $	$ V_d $
M-of-N	13	1000	2
Exactly	13	1000	2
Exactly2	13	1000	2
Heart	13	294	2
Vote	16	300	2
Credit	20	1000	2
Mushroom	22	8124	2
LED	24	2000	10
Letters	25	26	26
Derm	34	366	6
Derm2	34	358	6
WQ	38	521	13
Lung	56	32	3

also evaluated using different evaluation functions: RSAR (γ function) or Relative Dependency (κ function).

Table 6 show results for VNS, and **Table 7**, for DCS. Each entry of the table denotes the reducts as $Q^{(n)}$, where Q denotes the cardinality and the superscript n denotes the number of times a reduct with such cardinality was obtained, out of the 20 executions of the given metaheuristics. This superscript was omitted when $n = 20$, *i.e.*

Table 6. Reducts obtained by the VNS metaheuristics using either the standard or the VND local search schemes for the 13 considered datasets.

Datasets	$ A $	VNS(γ)-s	VNS(κ)-s	VNS(γ)-v	VNS(κ)-v
M-of-N	13	6	6	6	6
Exactly	13	6	6	6	6
Exactly2	13	10	10	10	10
Heart	13	6	6	6	6
Vote	16	8	8	8	8
Credit	20	8 L_8	8	8	8
Mushroom	22	3	3 L_8	3	3
LED	24	5	5	5	5
Letters	25	8	8	8	8
Derm	34	6	6	6	6
Derm2	34	8 ⁽⁹⁾ 9 ⁽¹¹⁾ L_{16}	8 ⁽¹⁸⁾ 9 ⁽⁸⁾ L_{16}	8 ⁽¹⁷⁾ 9 ⁽³⁾ L_{16}	8 ⁽¹²⁾ 9 ⁽⁸⁾ L_{16}
WQ	38	12 ⁽¹⁵⁾ 13 ⁽⁵⁾ L_{16}	12 ⁽¹⁶⁾ 13 ⁽⁴⁾ L_8	12 ⁽¹⁹⁾ 13 ⁽¹⁾ L_{16}	12 ⁽¹⁹⁾ 13 ⁽¹⁾ L_{16}
Lung	56	3 ⁽¹⁹⁾ 4 ⁽¹⁾ L_{16}	3 L_{16}	3 L_{16}	3 L_8

Table 7. Reducts obtained by the DCS metaheuristics using either the standard or the VND local search schemes for the 13 considered datasets.

Datasets	$ A $	DCS(γ)-s	DCS(κ)-s	DCS(γ)-v	DCS(κ)-v
M-of-N	13	6	6	6	6
Exactly	13	6	6	6	6
Exactly2	13	10	10	10	10
Heart	13	6	6	6	6
Vote	16	8	8	8	8
Credit	20	8	8 ⁽¹⁹⁾ 9 ⁽¹⁾	8	8
Mushroom	22	3	3	3	3
LED	24	5	5	5	5
Letters	25	8 ⁽¹⁸⁾ 9 ⁽²⁾	8	8 ⁽¹⁹⁾ 9 ⁽¹⁾	8
Derm	34	6	6	6	6
Derm2	34	8 ⁽³⁾ 9 ⁽¹⁶⁾	8 ⁽³⁾ 9 ⁽¹⁷⁾	8 ⁽²⁾ 9 ⁽¹⁷⁾	8 ⁽³⁾ 9 ⁽¹⁷⁾
WQ	38	12 ⁽¹⁹⁾ 13 ⁽¹⁾	12 ⁽¹⁹⁾ 13 ⁽¹⁾	12 ⁽¹⁹⁾ 13 ⁽¹⁾	12 ⁽¹⁵⁾ 13 ⁽⁵⁾
Lung	56	3	3 ⁽¹⁹⁾ 4 ⁽¹⁾	3	3

the same cardinality Q was obtained for all executions. In the case of VNS, most of the results were obtained with the lower Hamming distance $L = 4$. However, for test cases where higher values of L yielded better results, the value of L (8 or 16) appeared below the resulting cardinalities. As mentioned in Section 4.1, the default value for standard local search is $L = 1$, and for the VND local search scheme is $L = 2$.

It can be observed that VNS yielded better reducts using VND than the standard local search scheme. VNS results are in general slightly better than those of DCS. A comprehensive comparison is given by Table 8 that depicts an average ranking of the results for all databases using a skill score described below. The same table also ranks average processing times for all databases, showing that DCS is in general faster. These processing times are presented in Table 9 for all the variants of the metaheuristics.

In face of the multitude of reducts obtained by the different metaheuristic variants for the 13 datasets, it is difficult to compare the performance of the different metaheuristics. Therefore, a new metric was introduced,

Table 8. Ranking of the average skill score (R_s), ranking of the average processing time (R_t), VNS hamming distance (L), average skill score (S_{avg}) and standard deviation (σ) of the skill score for each proposed metaheuristic variant considering 20 executions for the 13 datasets.

Metaheuristic variant	R_s	R_t	L	S_{avg}	σ
VNS(γ)-v	1	22	16	0.9985	0.0055
VNS(κ)-v	2	13	16	0.9969	0.0111
VNS(κ)-s	3	11	16	0.9954	0.0120
VNS(γ)-s	4	21	16	0.9931	0.0170
VNS(κ)-v	5	9	8	0.9923	0.0196
VNS(γ)-v	6	20	8	0.9915	0.0203
DCS(κ)-v	7	7	-	0.9915	0.0251
VNS(κ)-s	8	8	8	0.9892	0.0229
DCS(κ)-s	9	5	-	0.9885	0.0300
DCS(γ)-s	10	16	-	0.9877	0.0298
VNS(γ)-v	11	17	4	0.9869	0.0284
DCS(γ)-v	12	18	-	0.9869	0.0269
VNS(γ)-s	13	19	8	0.9862	0.0299
VNS(κ)-v	14	6	4	0.9838	0.0355
VND(γ)	15	15	16	0.9808	0.0323
VND(κ)	16	4	16	0.9785	0.0378
VNS(γ)-s	17	14	4	0.9769	0.0413
VNS(κ)-s	18	3	4	0.9746	0.0475
VND(γ)	19	12	8	0.9638	0.0536
VND(κ)	20	2	8	0.9562	0.0558
ACO	21	-	-	0.9331	0.0901
VND(γ)	22	10	4	0.9323	0.0793
VND(κ)	23	1	4	0.9308	0.0764
SA	24	-	-	0.9269	0.1089
TS	25	-	-	0.9038	0.1364
GA	26	-	-	0.7662	0.1880

Table 9. Average execution times (in seconds) for 20 executions for each one of the 13 datasets.

Algorithm	Hamming distance			
	L_4	L_8	L_{16}	N.A.
VNS(γ)-s	1135	3178	13163	-
VNS(κ)-s	52	145	467	-
VNS(γ)-v	2117	4884	16813	-
VNS(κ)-v	107	246	684	-
VND(γ)	343	559	1673	-
VND(κ)	14	26	64	-
DCS(γ)-s	-	-	-	1682
DCS(γ)-v	-	-	-	2165
DCS(κ)-s	-	-	-	83
DCS(κ)-v	-	-	-	114

called skill score called skill score (S), as a means to evaluate the quality of the reducts. Considering a given dataset, the skill score S for N executions of a metaheuristic is defined as follows taking into account the obtained reducts with M different cardinalities:

$$S = \frac{C_{\min}}{N} \sum_{i=1}^M \frac{Q_i}{C_i} \quad (11)$$

where, C_{\min} is the cardinality of the best reduct found by any execution of any metaheuristic (assumed as being the minimum cardinality), C_i denotes the i -th cardinality, while Q_i denotes the number of reducts obtained with such cardinality out of the N executions. Therefore, if all executions yield reducts with the minimum cardinality, the skill score will be equal to one. The summation imposes a penalty for obtaining bad reducts many times. As an example, the 20 executions of VNS(γ)-s for the Derm2 dataset in **Table 6**, yielded $8^{(9)}9^{(11)}$. Since 8 is the best cardinality found, the resulting the skill score is $(8/20) \times [(9/8) + (11/9)]$ or 0.9389.

Table 8 presents the average skill score (and its mean deviation) of each metaheuristic variant for a particular choice of parameters (local search scheme, evaluation function and L). Considering a particular metaheuristic variant, it is executed 20 times for each of the 13 datasets, yielding the corresponding 13 partial averages. The average skill score is then given by the average of these partial averages. A compilation of such results separately for each dataset would be too extensive. Each choice of metaheuristic variant parameters is then ranked in the table according its skill score. It is worth to note that all the proposed metaheuristics variants attained better skill scores, in average, than the ones proposed in [5]. Observing the results for each dataset, the proposed metaheuristic variants always found reducts with minimum cardinality. For a given dataset, such minimum is not necessarily the best minimum, but the better minimum found in the tests (*i.e.* not an optimal minimum, but more likely a suboptimal minimum). However, some few sets of executions of ACO or TS may have yielded better reducts for some datasets than some proposed metaheuristic variants. For instance, for the dataset Letters, ACO obtained 20 times reducts with cardinality 8 (not shown here). The variants of the proposed metaheuristic VNS, VND and DCS achieved the same result, except for VND (γ) with $L = 8$ and DCS (κ)-s that obtained $8^{(19)}9^{(1)}$ and for DCS (γ)-s that obtained $8^{(18)}9^{(2)}$, but these can also be considered good solutions.

The choice of the evaluation function, RSAR (γ) or Relative Dependency (κ), does not seem to make difference in the cardinality of the reducts for the proposed metaheuristic variant. However, as explained ahead, the use of Relative Dependency implies in a much lower processing time. On the other hand, for VNS and VND variants, the use of higher values of L tends only to improve slightly the quality of the reducts for some databases. Typically $L = 4$ ensures good reducts, while $L = 8$ or $L = 16$ may allow better reducts for some few datasets.

The discussion about the results would be incomplete if processing times were not mentioned. Typically,

there is a trade-off between the quality of the results and processing time. Unfortunately, [5] or [15] did not mention execution times for ACO, SA, GA or TS, concerning the calculation of reducts for cited datasets. However, some conclusions can be drawn for the proposed metaheuristic variants based on their average total processing times, as shown in **Table 9**. Each entry corresponds to the average execution time for a particular variant, *i.e.* the average of the 20 executions for each one of the 13 datasets. In the case of VNS and VND, average times appear for the different Hamming distances.

A first point about computational performance is that the proposed metaheuristic and heuristic variants executes much faster (in general) using the Relative Dependency as evaluation function. On the other hand, the results obtained for ACO, SA, GA and TS were obtained using only the RSAR evaluation function. A second point is the choice of local search scheme in VNS and DCS variants. The choice of VND as local search scheme demands a slightly higher processing time, as would be expected, since the standard local search is simpler. A third point is the cardinality of the neighborhood structure (L). The choice of $L = 4$ seems suitable, since $L = 8$ or $L = 16$ roughly doubles or triplicates processing times, respectively. This issue does not apply for DCS variants.

The following discussion assumes the use of Relative Dependency evaluation function. In general, VNS variants obtained the best reducts, but using more process time. The average skill scores of VNS variants with $L = 16$ are the top ranked in **Table 8**, but it can be considered acceptable choices the use of VNS with $L = 8$ (up to 30 seconds using standard local search or up to 50 seconds using VND local search) or even VNS with $L = 4$ (up to 10 seconds using standard local search and up to 20 seconds using VND local search). The variants of VND are much faster due to its simplicity, demanding less than one second with $L = 4$ or $L = 8$ for most databases, but with lower ranked skill scores. Finally, DCS attained better skill scores than VND variants (except for the Mushroom dataset), but with processing times similar to VNS variants with $L = 8$. DCS variants demand a higher limit number of iterations to converge (30 iterations) than VNS variants, which require 10 iterations.

In particular, DCS variants obtained the best reducts only for the Lung dataset (cardinality 3), in comparison to most of the results of VNS and VND variants, that obtained reducts with cardinality 3 mixed with reducts with higher cardinality (4, 5 or 6). It can be noted that the best cardinality obtained by GA, ACO, SA or TS for the Lung dataset was 4. However, the computational performance of DCS variants was better than VNS or VND ones, since, in terms of skill score, the best ranked was DCS (κ)-v (7th with $s = 0.9915$) that had an overall average execution time ranking of also 7th (corresponding to 114 seconds), while VNS(κ)-v with $L = 8$ (5th with $s = 0.9923$) has a corresponding average time ranked at 9th (corresponding to 246 seconds).

In general, VNS variants always converge to an optimal or suboptimal reduct. The top six skill scores in **Table 8** were obtained by VNS variants using either of the local search schemes or either of the evaluation functions. Variants that employ the Relative Dependency evaluation function (κ) are much faster.

6. Final Remarks

This work presented the innovative application of three metaheuristics for attribute reduction based on the Rough Set theory (RST). Subsets of attributes/features called reducts are obtained for the given dataset and allow performing classification using the same dataset with reduced dimensionality. Normally, classification and computational performance are improved. Large and complex datasets require that the reduct calculation in RST be formulated as an optimization problem, and therefore solved by some metaheuristic. A standard metaheuristic for this purpose is GA, but others were proposed in the late years, like ACO, SA, or TS. Some commonly employed datasets were adopted to test these metaheuristics.

Variants of the three proposed metaheuristics, VNS, VND and DCS, were extensively tested for attribute reduction in RST with a set of well known datasets. In particular, DCS is a new metaheuristic developed for this work. The proposed metaheuristic variants obtained reducts with equal or better (*i.e.* lower) cardinality than GA, ACO, SA or TS, except for VND. VNS and DCS can probably be employed in a competitive manner for reduct calculation in RST at a (probably) lower computational cost, in part due to the choice of a less costly evaluation function. VND is a deterministic algorithm, while VNS and DCS have some stochasticity, but are inspired in deterministic methods. This would explain the fast convergence they present: the limit number of iterations was 10 for all variants of VNS, and 30 for the DCS ones.

In addition, the proposed metaheuristics do not require settings other than the limit number of iterations and the limit cardinality of neighborhood structure, which can be easily chosen. The other metaheuristics considered as references, ACO, SA, GA and TS, demand a careful choice of many parameters that are specific each algo-

rithm and that, sometimes, can be optimized only for a particular dataset, with loss of generality and at cost of many executions.

It would be difficult to choose the best metaheuristic variant between VNS and DCS. Results show that VNS variants have more robustness, *i.e.* always find good reducts, but at the cost of higher processing time. In general, due to its deterministic nature, VND variants demanded far less processing time, but generated reducts slightly poorer than VNS variants. It is worthy to note that VND variants did not seem prone to get trapped in local minima of the search space. The better choice may be guided by the dimensionality of the dataset: considering the same number of elements, a dataset with low dimensionality would be tackled with better accuracy by VNS, while another with high dimensionality, by DCS, since VNS would require too much processing time.

Finally, the new metaheuristic proposed in this work, DCS is a VNS variation that adopts a more aggressive search scheme, only accepting neighbor solutions with lower cardinality. DCS variants were competitive compared to VNS and VND variants, and for some datasets, obtained the best reducts (*i.e.* reducts with lower cardinality). Processing times of DCS variants are higher than those of VND variants, but lower than processing times of VNS variants with medium/high cardinality of neighborhood structure. Therefore, DCS variants seem to provide a good trade-off between the quality of the results and processing time.

The alternative metaheuristics proposed in this work applied for reducts calculation in RST performed better than formerly employed metaheuristics for the considered datasets, achieving the 20 top positions in the ranking of average skill scores. The three alternative metaheuristics (VNS, VND and DCS) do not require the tuning of specific parameters having thus more robustness than these previous approaches. In particular, DCS provides competitive results with lower processing time.

In recent years, a myriad of new optimization algorithms have been proposed for many different areas. A further work would exploit such new algorithms in the scope of attribute reduction in RST in order to tackle databases with even higher dimensionality than those presented here.

Acknowledgements

Authors Alex Sandro Aguiar Pessoa and Stephan Stephany are thankful to the Brazilian Council for Scientific and Technological Development (CNPq) for grants 140161/2010-4 and 305639/2012-9, respectively.

References

- [1] Pawlak, Z. (1982) Rough Sets. *International Journal of Computing and Information Sciences*, **11**, 341-356. <http://www.springerlink.com/index/r5556398717921x5.pdf>
- [2] Komorowski, J., Pawlak, Z., Polkowski, L. and Skowron, A. (1999) Rough Sets: A Tutorial. Warsaw. <http://secs.ceas.uc.edu/~mazlack/dbm.w2011/Komorowski.RoughSets.tutor.pdf>
- [3] Ohrn, A. (1999) Discernibility and Rough Sets in Medicine: Tools and Applications. Norwegian University of Science and Technology, Trondheim.
- [4] Jensen, R. and Shen, Q. (2003) Finding Rough Set Reducts with Ant Colony Optimization. *Proceedings of the 2003 UK Workshop on Computational Intelligence*, 15-22. <http://users.aber.ac.uk/rkj/pubs/papers/antRoughSets.pdf>
- [5] Hedar, A.-R., Wang, J. and Fukushima, M. (2008) Tabu Search for Attribute Reduction in Rough Set Theory. *Soft Computing*, **12**, 909-918. <http://www.springerlink.com/index/c4181mh5123816u3.pdf>
- [6] Pessoa, A.S.A., Stephany, S. (2012) Feature Selection with New Metaheuristics in the Rough Sets Theory. *XXXIV National Congress of Computational and Applied Mathematics*, **1**, 1-7.
- [7] Hansen, P. and Mladenovic, N. (2003) A Tutorial on Variable Neighborhood Search. Montreal. <http://yalma.fime.uanl.mx/~roger/work/teaching/mecbs5122/5-VNS/VNS-tutorial-G-2003-46.pdf>
- [8] Han, J., Sanchez, R. and Hu, X. (2005) Feature Selection Based on Relative Attribute Dependency: An Experimental Study. *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, **3641**, 214-223. <http://www.springerlink.com/index/743agvmf3etlaq0p.pdf> http://dx.doi.org/10.1007/11548669_23
- [9] Zadeh, L.A. (1965) Fuzzy Sets. *Information and Control*, **8**, 338-353. [http://dx.doi.org/10.1016/S0019-9958\(65\)90241-X](http://dx.doi.org/10.1016/S0019-9958(65)90241-X)
- [10] Shafer, G. (1976) A Mathematical Theory of Evidence. Princeton University Press, Princeton.
- [11] Dempster, A.P. (1967) Upper and Lower Probabilities Induced by a Multivalued Mapping. *The Annals of Mathematical Statistics*, **38**, 325-339. <http://dx.doi.org/10.1214/aoms/1177698950>
- [12] Zadeh, L.A. (1999) Fuzzy Sets as a Basis for a Theory of Possibility. *Fuzzy Sets and Systems*, **100**, 9-34.

- [http://dx.doi.org/10.1016/S0165-0114\(99\)80004-9](http://dx.doi.org/10.1016/S0165-0114(99)80004-9)
- [13] Nicoletti, M. do C. and Uchôa, J.Q. (1997) Rough Sets under the Perspective of Pertinence Function. *3rd Brazilian Symposium on Intelligent Automation*, Vitoria, Brazil, 307-313, in Portuguese.
- [14] Chouchoulas, A. and Shen, Q. (2001) Rough Set-Aided Keyword Reduction for Text Categorization. *Applied Artificial Intelligence*, **15**, 843-873. <http://www.tandfonline.com/doi/abs/10.1080/088395101753210773>
- [15] Jensen, R. and Shen, Q. (2005) Semantics-Preserving Dimensionality Reduction: Rough and Fuzzy-Rough-Based Approaches. *IEEE Transactions on Knowledge and Data Engineering*, **16**, 1457-1471. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1350758
- [16] Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor. <http://www.citeulike.org/group/664/article/400721>
- [17] Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Boston. <http://www.mendeley.com/research/genetic-algorithms-in-search-optimization-and-machine-learning/>
- [18] Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983) Optimization by Simulated Annealing. *Science*, **220**, 671-680. <http://www.sciencemag.org/content/220/4598/671.short> <http://dx.doi.org/10.1126/science.220.4598.671>
- [19] Glover, F. and McMillan, C. (1986) The General Employee Scheduling Problem. An Integration of MS and AI. *Computers Operations Research*, **13**, 563-573. [http://dx.doi.org/10.1016/0305-0548\(86\)90050-X](http://dx.doi.org/10.1016/0305-0548(86)90050-X)
- [20] Colomi, A., Dorigo, M. and Maniezzo, V. (1991) Distributed Optimization by Ant Colonies. *European Conference on Artificial Life*, Paris, France 134-142.
- [21] Hansen, P. and Mladenovic, N. (1997) Variable Neighborhood Search. *Computers & Operations Research*, **24**, 1097-1100. <http://www.sciencedirect.com/science/article/pii/S0305054897000312> [http://dx.doi.org/10.1016/S0305-0548\(97\)00031-2](http://dx.doi.org/10.1016/S0305-0548(97)00031-2)
- [22] Blake, C.L. and Merz, C.J. (1998) *UCI Repository of Machine Learning Databases*. University of California, Oakland. <http://www.ics.uci.edu/~mllearn>

Scientific Research Publishing (SCIRP) is one of the largest Open Access journal publishers. It is currently publishing more than 200 open access, online, peer-reviewed journals covering a wide range of academic disciplines. SCIRP serves the worldwide academic communities and contributes to the progress and application of science with its publication.

Other selected journals from SCIRP are listed as below. Submit your manuscript to us via either submit@scirp.org or [Online Submission Portal](#).

