Finding rough and fuzzy-rough set reducts with SAT

Jensen, Richard; Tuson, Andrew; Shen, Qiang

Published in:
Information Sciences

DOI:
10.1016/j.ins.2013.07.033

Publication date:
2014

Citation for published version (APA):
Jensen, R., Tuson, A., & Shen, Q. (2014). Finding rough and fuzzy-rough set reducts with SAT. *Information Sciences*, *255*, 100-120. https://doi.org/10.1016/j.ins.2013.07.033

Finding Rough and Fuzzy-Rough Set Reducts with SAT

Richard Jensen*,[a], Andrew Tuson[b], Qiang Shen[a]

[a] *Department of Computer Science, Aberystwyth University, Aberystwyth, Ceredigion SY23 3DB, Wales, UK.*
[b] *Department of Computing, School of Informatics, City University London, Northampton Square, London EC1V 0HB, England, UK.*

Abstract

Feature selection refers to the problem of selecting those input features that are most predictive of a given outcome; a problem encountered in many areas such as machine learning, pattern recognition and signal processing. In particular, solution to this has found successful application in tasks that involve datasets containing huge numbers of features (in the order of tens of thousands), which would otherwise be impossible to process further. Recent examples include text processing and web content classification. Rough set theory has been used as such a dataset pre-processor with much success, but current methods are inadequate at finding globally *minimal* reductions, the smallest sets of features possible. This paper proposes a technique that considers this problem from a propositional satisfiability perspective. In this framework, globally minimal subsets can be located and verified.

Key words: rough sets, fuzzy rough sets, feature selection, boolean satisfiability

*Corresponding author: rkj@aber.ac.uk

1. Introduction

Many problems in machine learning involve high dimensional descriptions of input features. It is therefore not surprising that much research has been carried out on dimensionality reduction [11]. However, existing work tends to destroy the underlying semantics of the features after reduction or require additional information about the given data set for thresholding. A technique that can reduce dimensionality using information contained within the dataset and that preserves the meaning of the features (i.e. semantics-preserving) is clearly desirable. Rough set theory (RST) can be used as such a tool to discover data dependencies and to reduce the number of attributes contained in a dataset using the data alone, requiring no additional information [31, 33].

Over the past ten years, RST has indeed become a topic of great interest to researchers and has been applied to many domains. Given a dataset with discretized attribute values, it is possible to find a subset (termed a *reduct*) of the original attributes using RST that are the most informative; all other attributes can be removed from the dataset with very little information loss. Therefore, there has been much research in the area of finding reducts, and in particular, reducts with minimal cardinality.

Heuristic methods such as [8, 16, 23, 48], although useful and relatively quick in locating reducts, are not able to guarantee such minimal reductions. This led to the application of stochastic-based approaches to this domain, such as Genetic Algorithms and extensions [44], Ant Colony Optimization [7], Particle Swarm Optimization [42], discussed further in section 2. However, there is still no guarantee of finding the smallest reducts with these methods. This motivates the work proposed in this paper. By reformulating

the rough set reduction task in a propositional satisfiability (SAT) framework [12], solution techniques from SAT may be applied that should be able to discover such subsets, guaranteeing their minimality.

Propositional satisfiability is one of the most studied NP-complete problems because of its significance in both theoretical research and practical applications. Applications of SAT include computer-aided design, model checking, planning and constraint satisfaction, and cryptography. Given a boolean formula in conjunctive normal form, the SAT problem requires an assignment of variables/features so that the formula evaluates to true, or a determination that no such assignment exists. Search algorithms based on the well-known Davis-Logemann-Loveland algorithm (DPLL) have been emerging as representatives of the most efficient methods for complete SAT solvers.

The issue of real-valued data is important and is central to real-world applications. This paper also proposes a fuzzy extension to crisp discernibility matrices that is utilized for the purpose of fuzzy-rough feature selection (FRFS) [18]. Additionally, the concepts in propositional satisfiability are fuzzified for use in a DPLL-like search to find the globally optimal subset of features.

Computational results on common machine learning benchmark problems indicate that the extended FRFS with SAT, denoted FRFS-SAT hereafter, produces no reduction in classification performance compared against the original and heuristically reduced datasets. In addition, the computational requirements are not excessive, given the ability of the algorithm to guarantee optimal data reductions.

The remainder of this paper is structured as follows. First the key concepts that underpin RST are reviewed and the minimal reduct problem

formulated in the context of current solution methods. The extension of rough set attribute reduction (RSAR) [8] with SAT, namely the RSAR-SAT algorithm, is then proposed to optimally find discrete reducts. The resulting method is further extended to the continuous case by fuzzification via formulating the concept of a fuzzy discernibility matrix. This leads onto the corresponding reduct solver FRFS-SAT. Computational results are then presented for both methods on appropriate benchmark data and conclusions drawn.

2. Rough Set Theory

Rough set theory [31] is an extension of conventional set theory that supports approximations in decision making. The rough set itself is the approximation of a vague concept (set) by a pair of precise concepts, called lower and upper approximations, which are a classification of the domain of interest into disjoint categories. The lower approximation is a description of the domain objects which are known with certainty to belong to the subset of interest, whereas the upper approximation is a description of the objects which possibly belong to the subset.

There are two main approaches to finding rough set reducts: those that consider the degree of dependency and those that are concerned with the discernibility matrix. This section describes the fundamental ideas behind both approaches. To illustrate the operation of these, an example dataset (Table 1) will be used.

2.1. Information and Decision Systems

An information system can be viewed as a table of data, consisting of objects (rows in the table) and attributes (columns). In medical datasets, for

example, patients might be represented as objects and measurements such as blood pressure, form attributes. The attribute values for a particular patient is their specific reading for that measurement. Throughout this paper, the terms attribute, feature and variable are used interchangeably.

An information system may be extended by the inclusion of decision attributes. Such a system is termed a decision system. For example, the medical information system mentioned previously could be extended to include patient classification information, such as whether a patient is ill or healthy. A more abstract example of a decision system can be found in Table 1. Here, the table consists of four conditional features (a, b, c, d), a decision feature (e) and eight objects. A decision system is *consistent* if for every set of objects whose attribute values are the same, the corresponding decision attributes are identical.

Table 1: An example dataset

$x \in \mathbb{U}$	a	b	c	d	$\Rightarrow$	e
0	S	R	T	T		R
1	R	S	S	S		T
2	T	R	R	S		S
3	S	S	R	T		T
4	S	R	T	R		S
5	T	T	R	S		S
6	T	S	S	S		T
7	R	S	S	R		S

More formally, $I = (\mathbb{U}, \mathbb{A})$ is an information system, where $\mathbb{U}$ is a non-

empty set of finite objects (the universe of discourse) and $\mathbb{A}$ is a non-empty finite set of attributes such that $a : \mathbb{U} \to V_a$ for every $a \in \mathbb{A}$. V_a is the set of values that attribute a may take. For decision systems, $\mathbb{A} = \{\mathbb{C} \cup \mathbb{D}\}$ where $\mathbb{C}$ is the set of input features and $\mathbb{D}$ is the set of class or decision indices. Here, a class index $d \in \mathbb{D}$ is itself a function $d : \mathbb{U} \to \{0, 1\}$ such that for $a \in \mathbb{U}, d(a) = 1$ if a has class d and $d(a) = 0$ otherwise.

2.2. Indiscernibility

With any $P \subseteq \mathbb{A}$ there is an associated equivalence relation $IND(P)$:

$$IND(P) = \{(x, y) \in \mathbb{U}^2 \mid \forall a \in P, a(x) = a(y)\} \tag{1}$$

Note that this corresponds to the equivalence relation for which two objects are equivalent if and only if they have the same vectors of attribute values for the attributes in P. The partition of $\mathbb{U}$, determined by *IND(P)* is denoted $\mathbb{U}/IND(P)$ or $\mathbb{U}/P$, which is simply the set of equivalence classes generated by $IND(P)$:

$$\mathbb{U}/IND(P) = \otimes\{\mathbb{U}/IND(\{a\}) \mid a \in P\}, \tag{2}$$

where

$$A \otimes B = \{X \cap Y \mid \forall X \in A, \forall Y \in B, X \cap Y \neq \emptyset\} \tag{3}$$

If $(x, y) \in IND(P)$, then x and y are indiscernible by attributes from P. The equivalence classes of the indiscernibility relation with respect to P are denoted $[x]_P$, $x \in \mathbb{U}$. For the illustrative example, if $P = \{b,c\}$, then objects 1, 6 and 7 are indiscernible; as are objects 0 and 4. *IND(P)* creates the following partition of $\mathbb{U}$:

$$
\begin{aligned}
\mathbb{U}/IND(P) &= \mathbb{U}/IND(b) \otimes \mathbb{U}/IND(c) \\
&= \{\{0,2,4\},\{1,3,6,7\},\{5\}\} \\
&\quad \otimes \{\{2,3,5\},\{1,6,7\},\{0,4\}\} \\
&= \{\{2\},\{0,4\},\{3\},\{1,6,7\},\{5\}\}
\end{aligned}
$$

2.3. Lower and Upper Approximations

Let $X \subseteq \mathbb{U}$. X can be approximated using only the information contained within P by constructing the P-*lower* and P-*upper* approximations of the classical crisp set X:

$$\underline{P}X = \{x \mid [x]_P \subseteq X\} \tag{4}$$

$$\overline{P}X = \{x \mid [x]_P \cap X \neq \emptyset\} \tag{5}$$

It is such a tuple $\langle \underline{P}X, \overline{P}X \rangle$ that is called a rough set.

2.4. Feature Dependency and Significance

Let P and Q be equivalence relations over $\mathbb{U}$, then the positive region is defined as:

$$POS_P(Q) = \bigcup_{X \in \mathbb{U}/Q} \underline{P}X \tag{6}$$

The positive region comprises all objects of $\mathbb{U}$ that can be classified to classes of $\mathbb{U}/Q$ using the information contained within attributes P. For example, let $P = \{b,c\}$ and $Q = \{e\}$, then

$$POS_P(Q) = \bigcup\{\emptyset, \{2,5\}, \{3\}\} = \{2,3,5\}$$

This means that objects 2, 3 and 5 can certainly be classified as belonging to a class in attribute e, when only considering attributes b and c. The rest of the objects cannot be classified as the information that would make them discernible is absent.

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

For $P,\ Q \subset \mathbb{A}$, it is said that Q depends on P in a degree k $(0 \leq k \leq 1)$, denoted $P \Rightarrow_k Q$, if

$$k = \gamma_P(Q) = \frac{|POS_P(Q)|}{|\mathbb{U}|} \tag{7}$$

where $|S|$ stands for the cardinality of set S.
If $k = 1$, Q depends totally on P, if $0 < k < 1$, Q depends partially (in a degree k) on P, and if $k = 0$ then Q does not depend on P. In the example, the degree of dependency of attribute $\{e\}$ from the attributes $\{b,c\}$ is:

$$\begin{aligned} \gamma_{\{b,c\}}(\{e\}) &= \frac{|POS_{\{b,c\}}(\{e\})|}{|\mathbb{U}|} \\ &= \frac{|\{2,3,5\}|}{|\{0,1,2,3,4,5,6,7\}|} = \frac{3}{8} \end{aligned}$$

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

$$\sigma_P(Q, x) = \gamma_P(Q) - \gamma_{P-\{x\}}(Q) \tag{8}$$

For example, if $P = \{a,b,c\}$ and $Q = e$ then

$$\begin{aligned}
\gamma_{\{a,b,c\}}(\{e\}) &= |\{2,3,5,6\}|/8 &= 4/8 \\
\gamma_{\{a,b\}}(\{e\}) &= |\{2,3,5,6\}|/8 &= 4/8 \\
\gamma_{\{b,c\}}(\{e\}) &= |\{2,3,5\}|/8 &= 3/8 \\
\gamma_{\{a,c\}}(\{e\}) &= |\{2,3,5,6\}|/8 &= 4/8
\end{aligned}$$

Also, calculating the significance of the three attributes gives:

$$\begin{aligned}
\sigma_P(Q,a) &= \gamma_{\{a,b,c\}}(\{e\}) - \gamma_{\{b,c\}}(\{e\}) &= 1/8 \\
\sigma_P(Q,b) &= \gamma_{\{a,b,c\}}(\{e\}) - \gamma_{\{a,c\}}(\{e\}) &= 0 \\
\sigma_P(Q,c) &= \gamma_{\{a,b,c\}}(\{e\}) - \gamma_{\{a,b\}}(\{e\}) &= 0
\end{aligned}$$

From this it follows that attribute a is indispensable, but attributes b and c can be dispensed with when considering the dependency between the decision attribute and the given individual conditional attributes.

2.5. Reducts

For many application problems, it is often necessary to maintain a concise form of the information system. One way to implement this is to search for a minimal representation of the original dataset. For this, the concept of a *reduct* is introduced and defined as a minimal subset R of the initial attribute set $\mathbb{C}$ such that for a given set of attributes D, $\gamma_R(D) = \gamma_{\mathbb{C}}(D)$. From the literature, R is a minimal subset if $\gamma_{R-\{a\}}(D) \neq \gamma_R(D)$ for all $a \in R$. This means that no attributes can be removed from the subset without affecting the dependency degree. Hence, a minimal subset by this definition may not be the *global* minimum (a reduct of smallest cardinality). A given dataset may have many reduct sets, and the collection of all reducts is denoted by

$$R_{all} = \{X \mid X \subseteq \mathbb{C}, \gamma_X(D) = \gamma_{\mathbb{C}}(D); \\ \gamma_{X-\{a\}}(D) \neq \gamma_X(D),\ \forall a \in X\} \quad (9)$$

The intersection of all the sets in R_{all} is called the *core*, the elements of which are those attributes that cannot be eliminated without introducing more contradictions to the representation of the dataset. For many tasks (for example, feature selection [11]), a reduct of minimal cardinality is ideally searched for. That is, an attempt is to be made to locate a single element of the reduct set $R_{min} \subseteq R_{all}$:

$$R_{min} = \{X \mid X \in R_{all},\ \forall Y \in R_{all}, |X| \leq |Y|\} \quad (10)$$

2.6. Discernibility Matrix

Many applications of rough sets make use of discernibility matrices for finding rules or reducts. A discernibility matrix [36] of a decision table $(\mathbb{U}, \mathbb{C} \cup \mathbb{D})$ is a symmetric $|\mathbb{U}| \times |\mathbb{U}|$ matrix with entries defined by:

$$c_{ij} = \{a \in \mathbb{C} | a(x_i) \neq a(x_j)\} \quad i, j = 1, ..., |\mathbb{U}| \quad (11)$$

Each c_{ij} contains those attributes that differ between objects i and j.

For finding reducts, the decision-relative discernibility matrix is of more interest. This only considers those object discernibilities that occur when the corresponding decision attributes differ. Returning to the example dataset, the decision-relative discernibility matrix is produced, as listed in Table 2. For example, it can be seen from the table that objects 0 and 1 differ in each attribute. Although some attributes in objects 1 and 3 differ, their corresponding decisions are the same so no entry appears in the decision-relative

matrix. Grouping all entries containing single attributes forms the core of the dataset (those attributes appearing in *every* reduct, which cannot be removed without introducing inconsistencies). Here, the core of the dataset is $\{d\}$.

Table 2: The decision-relative discernibility matrix

$x \in \mathbb{U}$	0	1	2	3	4	5	6	7
0								
1	a, b, c, d							
2	a, c, d	a, b, c						
3	b, c		a, b, d					
4	d	a, b, c, d		b, c, d				
5	a, b, c, d	a, b, c		a, b, d				
6	a, b, c, d		b, c		a, b, c, d	b, c		
7	a, b, c, d	d		a, c, d			a, d	

From this, the concept of discernibility functions can be introduced. This is a concise notation of how each object within the dataset may be distinguished from the others. A discernibility function f_D is a boolean function of m boolean variables $a_1^*, ..., a_m^*$ (corresponding to the membership of attributes $a_1, ..., a_m$ to a given entry of the discernibility matrix), defined as below:

$$f_D(a_1^*, ..., a_m^*) = \wedge\{\vee c_{ij}^* | 1 \leq j \leq i \leq |\mathbb{U}|, c_{ij} \neq \emptyset\} \tag{12}$$

where $c_{ij}^* = \{a^* | a \in c_{ij}\}$. By finding the set of all prime implicants of the discernibility function, all the minimal reducts of a system may be deter-

mined. From table 2, the decision-relative discernibility function is (with duplicates removed):

$$\begin{aligned} f_D(a^*, b^*, c^*, d^*) &= (a^* \vee b^* \vee c^* \vee d^*) \wedge (a^* \vee c^* \vee d^*) \\ &\wedge (b^* \vee c^*) \wedge (d^*) \wedge (a^* \vee b^* \vee c^*) \\ &\wedge (a^* \vee b^* \vee d^*) \wedge (b^* \vee c^* \vee d^*) \\ &\wedge (a^* \vee d^*) \end{aligned}$$

Further simplification can be performed by removing those clauses that are subsumed by others:

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

The reducts of the dataset may be obtained by converting the above expression from conjunctive normal form to disjunctive normal form (without negations) [32]. Hence, the minimal reducts are $\{b, d\}$ and $\{c, d\}$. Although this is guaranteed to discover all minimal subsets, it is a costly operation rendering the method impractical for even medium-sized datasets.

For most applications, a single minimal subset is required for data reduction. This has led to approaches that consider finding individual shortest prime implicants from the discernibility function. A common method is to incrementally add those attributes that occur with the highest frequency in the function, removing any clauses containing the attributes, until all clauses are eliminated [28]. However, even this does not ensure that a minimal subset is found - the search can proceed down non-minimal paths.

2.7. Techniques for Finding Reducts

In the rough set literature, a *minimal* reduct is defined as a subset of features that have the same dependency as the full set of conditional features, and also no proper subset of this exists such that the dependency remains maximal. However, this terminology is slightly misleading in that a reduct may not be minimal in the sense of having the smallest cardinality amongst all reducts. In this section, rough set-based feature selection techniques are classified into three categories: whether they find superreducts (i.e. they find subsets that have maximal dependency but do not necessarily satisfy the minimality condition), minimal reducts, or reducts of smallest cardinality. The work in this paper attempts to tackle the most challenging of these tasks, finding reducts of smallest cardinality.

2.7.1. Superreducts

The QuickReduct algorithm [8] attempts to determine reducts. It starts off with an empty set and adds in turn, one at a time, those attributes that result in the greatest increase in the rough set dependency metric, until this produces its maximum possible value for the dataset. Other such techniques may be found in [33].

In [48], a heuristic filter-based approach is presented based on rough set theory. The algorithm begins with the core of the dataset and incrementally adds attributes based on a heuristic measure. Additionally, a threshold value is required as a stopping criterion to determine when a reduct candidate is "near enough" to being a reduct. On each iteration, those objects that are consistent with the current reduct candidate are removed (an optimization that can be used with RSAR). As the process starts with the core of the dataset, this has to be calculated beforehand. Using the discernibility matrix

for this purpose can be quite impractical for datasets of large dimensionality.

The Johnson Reducer is a simple greedy heuristic algorithm that is often applied to discernibility functions to find a single reduct [29]. The algorithm begins by setting the current reduct candidate, R, to the empty set. Then, each conditional attribute appearing in the discernibility function is evaluated according to the heuristic measure. For the standard Johnson algorithm, this is typically a count of the number of appearances an attribute makes within the logical contexts termed clauses; attributes that appear more frequently are considered to be more significant. The attribute with the highest heuristic value is added to the reduct candidate and all clauses in the discernibility function containing this attribute are removed. As soon as all clauses have been removed, the algorithm terminates and returns the reduct R. R is assured to be a (super)reduct as all clauses contained within the discernibility function have been addressed. Variations of the algorithm involve alternative heuristic functions in an attempt to guide search down better paths [28, 41]. However, no perfect heuristic exists, and hence there is still no guarantee of subset optimality.

Recent approaches to the problem include the work conducted in [20], where a technique is detailed that breaks down large information systems into a master table and several more manageable sub-tables (an approach not dissimilar to that of finding dynamic reducts [2]). Reducts are then calculated for the smaller tables and used to find reducts for the master table in a more efficient manner. Additionally, there have been attempts to accelerate the search for reducts within existing heuristic methods, e.g. [34] where a framework is proposed based on positive approximations for this purpose, and [45] where a condensing tree structure is used for optimization.

Also worth mentioning are the approaches reported in [3, 44] which use

genetic algorithms to discover optimal or close-to-optimal reducts. Reduct candidates are encoded as bit strings, with the value in position i set if the ith attribute is present. The fitness function depends on two parameters. The first is the number of bits set. The function penalises those strings which have larger numbers of bits set, driving the process to find smaller reducts. The second is the number of classifiable objects given this candidate. The reduct should discern between as many objects as possible (ideally all of them).

Although this approach is not guaranteed to find minimal subsets, it may find many subsets for any given dataset. It is also useful for situations where new objects are added to or old objects are removed from a dataset - the reducts generated previously can be used as the initial population for the new reduct-determining process. The main drawback is the time taken to compute the fitness of each bit string, which is $O(a.o^2)$, where a is the number of attributes and o the number of objects in the dataset. The extent to which this hampers performance depends on several factors, including the population size.

Other applications of evolutionary algorithms to reduct finding include Ant Colony Optimization [7, 17], Particle Swarm Optimization [42] and Estimation of Distribution Algorithms [5]. Although all these methods have been demonstrated to be successful in locating small subsets, the algorithms cannot guarantee that the results found are minimal reducts.

2.7.2. Minimal reducts

A method based on QuickReduct, called ReverseReduct [17], has been proposed where the strategy is the backward elimination of attributes as opposed to the forward selection process used by QuickReduct. Ini-

tially, all attributes appear in the reduct candidate; the least informative ones are incrementally removed until no further attribute can be eliminated without introducing inconsistencies. This method results in minimal reducts, as it will terminate when the removal of any feature results in a decrease in dependency degree. Obviously, this method may be restrictive if the number of the initial attributes is large.

In [41] an algorithm is proposed to find minimal reducts with user preferences via the discernibility matrix. The matrix itself is transformed into a set of subsets of attributes corresponding to the matrix elements. A reduct is constructed through repeated application of absorption and grouping operations. The complexity of this method is $O(c^2)$ where the discernibility matrix has c clauses.

Another approach that determines minimal reducts by simplification of the discernibility matrix is presented in [46]. A number of elementary matrix operations are introduced that, by applying them a finite number of times, result in a minimum discernibility matrix. The union of all elements in the minimum matrix produces a minimal reduct.

A method for determining globally optimal reducts by first finding a sub-optimal reduct for a decision system, and then examining all possible sub-systems is proposed in [23]. However, its complexity is rather high, and in fact, may not necessarily find the true globally optimal reduct, as the reduct discovered by the heuristic process needs to be a superset of the globally optimal reduct, which may not be the case.

2.7.3. Globally optimal reducts

As mentioned previously, for small datasets, reducts of smallest cardinality may be found by calculating the prime implicants of the discernibility

function. This is typically achieved by converting the function to disjunctive normal form - a costly operation, impractical for most datasets.

In [37, 38], a method for the generation of all reducts in an information system by manipulating the clauses in discernibility functions is reported. In addition to the standard simplification laws, the concept of strong compressibility is introduced and applied in conjunction with an expansion algorithm. Although the method can output a globally optimal reduct (as all reducts are generated), it still exhibits a prohibitively high computational complexity.

3. RSAR-SAT

The Propositional Satisfiability (SAT) problem [12] is one of the most studied NP-complete problems because of its significance in both theoretical research and practical applications. Given a boolean formula (typically in conjunctive normal form (CNF)), the SAT problem requires an assignment of variables/features so that the formula evaluates to true, or a determination that no such assignment exists. In recent years, search algorithms based on the well-known Davis-Logemann-Loveland algorithm (DPLL) [12] have been emerging as representatives of the most efficient methods for complete SAT solvers. Such solvers can either find a solution or prove that no solution exists.

Stochastic techniques have also been developed in order to reach a solution quickly. These pick random locations in the space of possible assignments and perform limited local searches from them. However, as these techniques do not examine the entire search space, they are unable to prove unsatisfiability.

A CNF formula on n binary variables $x_1, ..., x_n$ is the conjunction of m clauses $C_1, ..., C_m$ each of which is the disjunction of one or more literals. A literal is the occurrence of a variable or its negation. A formula denotes a unique n-variable boolean function $f(x_1, ..., x_n)$. Clearly, such a function can be represented by many equivalent CNF formulas. The satisfiability problem is concerned with finding an assignment to the arguments of $f(x_1, ..., x_n)$ that makes the function equal to 1, signalling that it is satisfiable, or proving that the function is equal to 0 and hence unsatisfiable [47]. By viewing the selection problem as a variant of SAT, with a bound on true assignments, techniques from this field can be applied to reduct search.

3.1. Finding Rough Set Reducts

The problem of finding the smallest feature subsets using rough set theory can be formulated as a SAT problem. This theory allows the generation from datasets of clauses of features in conjunctive normal form. If after assigning truth values to all features appearing in the clauses the formula is satisfied, then those features set to true constitute a valid subset for the data. The task is to find the smallest number of such features so that the CNF formula is satisfied. In other words, the problem here concerns finding a minimal assignment amongst the arguments of $f(x_1, ..., x_n)$ that makes the function equal to 1. There will be at least one solution to the problem (i.e. all x_i, $i = 1, 2, ..., n$, set to 1) for consistent datasets. Preliminary work has been carried out in this area [1], though this does not adopt a DPLL-style approach to finding solutions.

The DPLL algorithm for finding minimal subsets can be found in Figure 1, where a search is conducted in a depth-first manner. The key operation in this procedure is the unit propagation step, unitPropagate(F), in lines

(6) and (7). Clauses in the formula that contain a single literal will only be satisfied if that literal is assigned the value 1 (for positive literals). These are called unit clauses. Unit propagation examines the current formula for unit clauses and automatically assigns the appropriate value to the literal they contain. The elimination of a literal can create new unit clauses, and thus unit propagation eliminates variables by repeated passes until there is no unit clause in the formula. The order of the unit clauses within the formula makes no difference to the results or the efficiency of the process.

Branching occurs at lines (9) to (12) via the function selectLiteral(F). Here, the next literal is chosen heuristically from the current formula, assigned the value 1, and the search continues. If this branch eventually results in unsatisfiability, the procedure will assign the value 0 to this literal instead and continue the search. The importance of choosing good branching literals is well known - different branching heuristics may produce drastically different sized search trees for the same basic algorithm, thus significantly affecting the efficiency of the solver. The heuristic currently used within RSAR-SAT is to select the variable that appears in the most clauses in the current set of clauses. Many other heuristics exist which may be used as alternatives for this purpose [47], but these are not addressed here.

A degree of pruning can take place in the search by remembering the size of the currently considered subset and the smallest optimal subset encountered so far. If the number of literals currently assigned the value 1 equals the number of those in the presently optimal subset, and the satisfiability of F is still not known, then any further search down this branch will not result in a smaller optimal subset.

Although stochastic methods have been applied to SAT problems [15], these are not applicable here as they provide no guarantee of solution min-

DPLL(F).

F, the formula containing the current set of clauses.

```
(1)  if (F contains an empty clause)
(2)      return unsatisfiable
(3)  if (F is empty)
(4)      output current assignment
(5)      return satisfiable
(6)  if (F contains a unit clause {l})
(7)      F' ← unitPropagate(F)
(8)      return DPLL(F')
(9)  x ← selectLiteral(F)
(10) if ( DPLL(F ∪ {x}) is satisfiable)
(11)     return satisfiable
(12) else return DPLL(F ∪ {−x})
```

Figure 1: Definition of the DPLL algorithm

imality. The DPLL-based algorithm will always find the minimal optimal subset. However, this will come at the expense of time taken to find it.

3.2. Example

The dataset given in table 1 is used here to illustrate the operation of RSAR-SAT. The initial step is to generate the list of clauses, which is equivalent to generating the decision-relative discernibility function:

$$
\begin{aligned}
f_D(a^*, b^*, c^*, d^*) \quad = \quad & (a^* \vee b^* \vee c^* \vee d^*) \wedge (a^* \vee c^* \vee d^*) \\
& \wedge (b^* \vee c^*) \wedge (d^*) \wedge (a^* \vee b^* \vee c^*) \\
& \wedge (a^* \vee b^* \vee d^*) \wedge (b^* \vee c^* \vee d^*) \\
& \wedge (a^* \vee d^*)
\end{aligned}
$$

As was discussed earlier, this can be simplified further. For the purposes of this example, the simplification phase is skipped in order to better illustrate the steps involved.

Clause (d) is a unit clause, and therefore must be assigned the value 1 (otherwise the set of clauses cannot be satisfied). This assignment is then propagated through the other clauses, such that all clauses containing the literal d are removed. The remaining clauses are:

$$(b^* \vee c^*) \wedge (a^* \vee b^* \vee c^*)$$

The algorithm recurses using this new set of clauses. The best literal is then selected as there are no unit clauses; the frequency of occurrence is used here as the heuristic for selection. As can be seen, there are two literals, b and c, that appear in both clauses, whilst a only appears in one clause. The algorithm then makes the arbitrary choice between b and c, and all clauses have been satisfied. The final reduct is $\{b, d\}$ or $\{c, d\}$, and these are the reducts of minimal cardinality for this dataset.

4. Fuzzy Discernibility Matrices

The RSAR process above can only operate effectively with datasets containing discrete values. There is also no way of handling noisy data. As

most datasets contain real-valued attributes, it is necessary to perform a discretization step beforehand. This is typically implemented by standard fuzzification techniques, enabling linguistic labels to be associated with attribute values. However, membership degrees of attribute values to fuzzy sets are not exploited in the process of dimensionality reduction. By using *fuzzy-rough* sets [13], it is possible to use this information to better guide feature selection; this already has been shown to be a highly useful technique in reducing data dimensionality [17]. The technique outlined here was proposed in [18], which also contains a walkthrough of the fuzzy-rough feature selection process.

4.1. Fuzzy-Rough Approximations

Definitions for the fuzzy lower and upper approximations can be found in [9, 35], where a T-transitive fuzzy similarity relation is used to approximate a fuzzy concept X:

$$\mu_{\underline{R_P}X}(x) = \inf_{y \in \mathbb{U}} I(\mu_{R_P}(x,y), \mu_X(y)) \tag{13}$$

$$\mu_{\overline{R_P}X}(x) = \sup_{y \in \mathbb{U}} T(\mu_{R_P}(x,y), \mu_X(y)) \tag{14}$$

Here, I is a fuzzy implicator, T is a t-norm and $x \in \mathbb{U}$. R_P is the fuzzy similarity relation induced by the subset of features P:

$$\mu_{R_P}(x,y) = \mathcal{T}_{a \in P}\{\mu_{R_a}(x,y)\} \tag{15}$$

$\mu_{R_a}(x,y)$ is the degree to which objects $x \in \mathbb{U}$ and $y \in \mathbb{U}$ are similar for feature a. Many fuzzy similarity relations can be constructed for this purpose, for example:

$$\mu_{R_a}(x,y) = \exp(-\frac{(a(x) - a(y))^2}{2{\sigma_a}^2}) \tag{16}$$

$$\mu_{R_a}(x,y) = \max(\min(\frac{(a(y)-(a(x)-\sigma_a))}{(\sigma_a)}, \frac{((a(x)+\sigma_a)-a(y))}{(\sigma_a)}), 0) \tag{17}$$

where ${\sigma_a}^2$ is the variance of feature a. As these relations do not necessarily display T-transitivity, the fuzzy transitive closure must be computed for each attribute. The combination of feature relations in equation (15) has been shown to preserve T-transitivity [40].

In a similar way to the original FRFS approach [17], the fuzzy positive region can be defined as:

$$\mu_{POS_{R_P}(Q)}(x) = \sup_{X \in \mathbb{U}/Q} \mu_{\underline{R_P}X}(x) \tag{18}$$

The resulting degree of dependency is:

$$\gamma'_P(Q) = \frac{\sum_{x \in \mathbb{U}} \mu_{POS_{R_P}(Q)}(x)}{|\mathbb{U}|} \tag{19}$$

A fuzzy-rough reduct R can be defined as a (locally minimal) subset of features that preserves the dependency degree of the entire dataset, i.e. $\gamma'_R(\mathbb{D}) = \gamma'_{\mathbb{C}}(\mathbb{D})$. Core features may be determined by considering the change in dependency of the full set of conditional features when individual attributes are removed:

$$Core(\mathbb{C}) = \{a \in \mathbb{C} | \gamma'_{\mathbb{C}-\{a\}}(Q) < \gamma'_{\mathbb{C}}(Q)\} \tag{20}$$

4.2. Fuzzy Discernibility Matrix-based Feature Selection

As indicated previously, there are two main branches of research in crisp rough set-based feature selection: those based on the dependency degree and those based on discernibility matrices. The developments above are solely concerned with the extension of the dependency degree to the fuzzy-rough

case. Hence, methods constructed based on the crisp dependency degree can be employed for fuzzy-rough FS. By extending the discernibility matrix to the fuzzy case, it is possible to employ approaches similar to those in crisp rough set FS to determine fuzzy-rough reducts. A first step toward this is presented in [6, 39] where a crisp discernibility matrix is constructed for fuzzy-rough selection. Thresholding is used, breaking the rough set ideology (which ensures that no information other than the dataset itself is needed for reduct search), which determines which features are to appear in the matrix entries. However, as membership degrees are not considered, search based on the crisp discernibility may result in reducts that are not true fuzzy-rough reducts.

4.2.1. Fuzzy Discernibility

The crisp discernibility matrix is herein extended by employing fuzzy clauses. Entries in the fuzzy discernibility matrix is a fuzzy set, to which every feature belongs to a certain degree. The extent to which a feature a belongs to the fuzzy clause C_{ij} is determined by the fuzzy discernibility measure:

$$\mu_{C_{ij}}(a) = N(\mu_{R_a}(i,j)) \tag{21}$$

where N denotes fuzzy negation and $\mu_{R_a}(i,j)$ is the fuzzy similarity of objects i and j, and hence $\mu_{C_{ij}}(a)$ is a measure of the fuzzy discernibility. For the crisp case, if $\mu_{C_{ij}}(a) = 1$ then the two objects are distinct for this feature; if $\mu_{C_{ij}}(a) = 0$, the two objects are identical. For fuzzy cases where $\mu_{C_{ij}}(a) \in (0,1)$, the objects are partly discernible. Note that the choice of fuzzy similarity relation must be identical to that of the fuzzy-rough dependency degree approach to find corresponding reducts. Each entry in the

fuzzy indiscernibility matrix is a set of attributes and their memberships:

$$C_{ij} = \{a_x | a \in \mathbb{C}, x = N(\mu_{R_a}(i,j))\}\ i,j = 1,...,|\mathbb{U}| \tag{22}$$

For example, an entry C_{ij} in the fuzzy discernibility matrix might be: $\{a_{0.4}, b_{0.8}, c_{0.2}, d_{0.0}\}$. This denotes that $\mu_{C_{ij}}(a) = 0.4$, $\mu_{C_{ij}}(b) = 0.8$, etc. In crisp discernibility matrices, these values are either 0 or 1 as the underlying relation is an equivalence relation. The example clause can be viewed as indicating the significance value of each feature - the extent to which the feature discriminates between the two objects i and j. The core of the dataset is defined as:

$$\begin{aligned} Core(\mathbb{C}) = \{a \in \mathbb{C} | \exists C_{ij}, \mu_{C_{ij}}(a) > 0, \\ \forall f \in \{\mathbb{C} - a\}\ \mu_{C_{ij}}(f) = 0\} \end{aligned} \tag{23}$$

For clauses generated via the decision-relative discernibility matrix (see later), this is equivalent to equation (20).

4.2.2. Fuzzy Discernibility Function

As with the crisp approach, the entries in the matrix can be used to construct the fuzzy discernibility function:

$$f_D(a_1^*, ..., a_m^*) = \wedge\{\vee C_{ij}^* | 1 \leq j < i \leq |\mathbb{U}|\} \tag{24}$$

where $C_{ij}^* = \{a_x^* | a_x \in C_{ij}\}$. The function returns values in $[0,1]$, which can be seen to be a measure of the extent to which the function is satisfied for a given assignment of truth values to variables. To discover globally minimal reducts from the fuzzy discernibility function, the task is to find the minimal assignment of the value 1 to the variables such that the formula is maximally satisfied. By setting all variables to 1, the maximal value for the function can be obtained as this provides the most discernibility between objects.

4.2.3. Decision-relative Fuzzy Discernibility Matrix

As with the crisp discernibility matrix, for a decision system the decision feature must be taken into account for achieving reductions; only those clauses with different decision values are included in the crisp discernibility matrix. For the fuzzy version, this is encoded as:

$$f_D(a_1^*, ..., a_m^*) = \{\wedge\{\{\vee C_{ij}^*\} \leftarrow q_{N(\mu_{R_q}(i,j))}\} | \\ 1 \leq j < i \leq |\mathbb{U}|\} \tag{25}$$

for decision feature q, where $\leftarrow$ denotes fuzzy implication. This allows the extent to which decision values differ to affect the overall satisfiability of the clause. If $\mu_{C_{ij}}(q) = 1$ then this clause provides maximum discernibility (i.e., the two objects are maximally different according to the fuzzy similarity measure). When the decision is crisp and crisp equivalence is used, $\mu_{C_{ij}}(q)$ becomes 0 or 1.

5. FRFS-SAT

Reducts are calculated via the fuzzy clauses from the construction of the fuzzy discernibility function above. Crisp discernibility matrices can be adapted with suitable extensions. The aim here is to determine those reducts that are minimal in the global sense (i.e., of smallest cardinality). Thus, heuristic techniques are not applicable as the resulting reducts may not satisfy this property, and there is no computationally efficient way of determining this for a particular reduct. This section proposes a fuzzy extension to propositional satisfiability for the purpose of determining globally minimal reducts.

5.1. Formulation

The degree of satisfaction of a clause C_{ij} for a subset of features P is defined as:

$$SAT_P(C_{ij}) = \mathcal{S}_{a \in P}\{\mu_{C_{ij}}(a)\} \tag{26}$$

for a t-conorm $\mathcal{S}$. Returning to the example clause $\{a_{0.4}, b_{0.8}, c_{0.2}, d_{0.0}\}$, if the subset $P = \{a, c\}$ is chosen, the resulting degree of satisfaction of the clause is

$$SAT_P(C_{ij}) = \mathcal{S}\{0.4, 0.2\} = 0.6$$

using the Łukasiewicz t-conorm, $\min(1, x + y)$.

In traditional (crisp) propositional satisfiability, a clause is fully satisfied if at least one variable in the clause has been set to `true`. For the fuzzy case, clauses may be satisfied to a certain degree depending on which variables have been assigned the value `true`. By setting $P = \mathbb{C}$, the maximum satisfiability degree of a clause can be obtained:

$$maxSAT_{ij} = SAT_{\mathbb{C}}(C_{ij}) = \mathcal{S}_{a \in \mathbb{C}}\{\mu_{C_{ij}}(a)\} \tag{27}$$

This is the maximal amount that clause C_{ij} may be satisfied. The maximum satisfiability degree of the example clause is $\mathcal{S}(0.4, 0.8, 0.2, 0.0)$ which evaluates to 1 if the Łukasiewicz t-conorm is used. Here it can be seen that, depending on the t-conorm used, clauses may in fact be maximally satisfied by the selection of several sub-maximal features. Using the max t-conorm, the maximum satisfiability degree is 0.8, obtained only by the inclusion of feature b in P.

In this setting, a fuzzy-rough reduct corresponds to a (minimal) truth assignment to variables such that each clause has been satisfied to its maximum extent. See the appendix for a proof that fuzzy-rough reducts maximally satisfy the set of clauses for a given dataset.

5.2. Algorithm

The DPLL-based algorithm for finding minimal subsets is given in Figure 2, where search is conducted in a depth-first manner. The key operation in this procedure is the unit propagation step, unitPropagate(CL), in lines (6) and (7). Clauses in the formula that contain a single literal will only be satisfied if that literal is assigned the value `true` (unit clauses). Unit propagation examines the current formula for unit clauses and assigns the appropriate value to the literal they contain. The elimination of a literal can create new unit clauses, and thus unit propagation eliminates variables by repeated passes until there is no unit clause in the formula. The order of the unit clauses within the formula makes no difference to the results or the efficiency of the process.

Branching occurs at lines (10) to (14) via the function selectLiteral(CL). Here, the next literal is chosen heuristically from the current formula, assigned the value `true`, and the search continues. If this branch eventually results in unsatisfiability, the procedure assigns the value `false` to this literal instead and continues the search. Choosing good branching literals is important - different branching heuristics may produce drastically different sized search trees for the same basic algorithm, affecting the efficiency of the solver.

One heuristic is to select the variable whose fuzzy discernibility is non-zero in the most clauses of the current set of clauses. Alternatively, the sum of the fuzzy discernibilities for a particular attribute across all clauses gives a good indication of attribute importance. This is the heuristic adopted in this work.

Pruning can be carried out in the search by remembering the size of the currently considered subset d and the smallest optimal subset encountered

so far D. If the number of variables currently assigned the value `true` equals the number of those in the presently optimal subset then any further search down this branch will not result in a smaller optimal subset. Also, if an empty clause is generated during UPDATE-FALSE, the algorithm stops the search down this branch.

Line (3) is reached when all clauses have been maximally satisfied (namely, a fuzzy-rough reduct has been reached) and the corresponding variable assignment is returned as the output. This finally returned variable assignment is the globally minimal reduct.

Figure 3 shows the update of the current clause list if the variable x is set to `true`. The updated clause list is stored in CL' and returned upon completion. Line (4) determines if the clause C will be maximally satisfied if variable x is set to true. If not, the fuzzy clause is retained and added to the updated clause list. Once a clause is maximally satisfied, it is not considered further down this branch in the search.

When the chosen literal is assigned the value `false` (i.e., it does not appear in subsets beyond this branching point), the fuzzy clauses are updated according to Figure 4. Each clause C in the current set of clauses is examined. In line (4), $|C|$ denotes the number of literals in the clause that can be set to `true`; if this is zero, then this clause cannot be satisfied. Line (4) also checks to see if the clause is satisfiable, i.e. whether it could potentially reach the maximum satisfiability degree if further literals are chosen. If not, the current variable assignment cannot lead to a fuzzy-rough reduct, and so search down this branch need not be considered any further.

5.2.1. Example

Table 3 illustrates the operation of FRFS-SAT, using an example dataset. The fuzzy connectives used are the Łukasiewicz t-norm ($\max(x + y - 1, 0)$)

Table 3: Example dataset

Object	a	b	c	q
1	-0.4	-0.3	-0.5	no
2	-0.4	0.2	-0.1	yes
3	-0.3	-0.4	-0.3	no
4	0.3	-0.3	0	yes
5	0.2	-0.3	0	yes
6	0.2	0	0	no

and the Łukasiewicz fuzzy implicator ($\min(1 - x + y, 1)$). The use of this implicator is recommended as it is both a residual and S-implicator.

Using the fuzzy similarity measure in (17), the resulting relations are as follows for each feature in the dataset:

$$R_a(x, y) = \begin{pmatrix} 1.0 & 1.0 & 0.699 & 0.0 & 0.0 & 0.0 \\ 1.0 & 1.0 & 0.699 & 0.0 & 0.0 & 0.0 \\ 0.699 & 0.699 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.699 & 0.699 \\ 0.0 & 0.0 & 0.0 & 0.699 & 1.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 0.699 & 1.0 & 1.0 \end{pmatrix}$$

$R_b(x, y) =$

$$\begin{pmatrix} 1.0 & 0.0 & 0.568 & 1.0 & 1.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.137 \\ 0.568 & 0.0 & 1.0 & 0.568 & 0.568 & 0.0 \\ 1.0 & 0.0 & 0.568 & 1.0 & 1.0 & 0.0 \\ 1.0 & 0.0 & 0.568 & 1.0 & 1.0 & 0.0 \\ 0.0 & 0.137 & 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$$

$R_c(x, y) =$

$$\begin{pmatrix} 1.0 & 0.0 & 0.036 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.036 & 0.518 & 0.518 & 0.518 \\ 0.036 & 0.036 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.518 & 0.0 & 1.0 & 1.0 & 1.0 \\ 0.0 & 0.518 & 0.0 & 1.0 & 1.0 & 1.0 \\ 0.0 & 0.518 & 0.0 & 1.0 & 1.0 & 1.0 \end{pmatrix}$$

Next, the fuzzy discernibility matrix needs to be constructed on the basis of the fuzzy discernibility given in equation (21). For objects 2 and 3, the resulting fuzzy clause is $\{a_{0.301} \vee b_{1.0} \vee c_{0.964}\} \leftarrow q_{1.0}$.

The fuzzy discernibility of objects 2 and 3 for attribute a is 0.301, indicating that the objects are partly discernible for this feature. The objects are fully discernible with respect to the decision feature, indicated by $q_{1.0}$. The set of clauses is:

$$
\begin{aligned}
C_{12}: &\quad \{a_{0.0} \vee b_{1.0} \vee c_{1.0}\} &\leftarrow\quad & q_{1.0}\\
C_{13}: &\quad \{a_{0.301} \vee b_{0.432} \vee c_{0.964}\} &\leftarrow\quad & q_{0.0}\\
C_{14}: &\quad \{a_{1.0} \vee b_{0.0} \vee c_{1.0}\} &\leftarrow\quad & q_{1.0}\\
C_{15}: &\quad \{a_{1.0} \vee b_{0.0} \vee c_{1.0}\} &\leftarrow\quad & q_{1.0}\\
C_{16}: &\quad \{a_{1.0} \vee b_{1.0} \vee c_{1.0}\} &\leftarrow\quad & q_{0.0}\\
C_{23}: &\quad \{a_{0.301} \vee b_{1.0} \vee c_{0.964}\} &\leftarrow\quad & q_{1.0}\\
C_{24}: &\quad \{a_{1.0} \vee b_{1.0} \vee c_{0.482}\} &\leftarrow\quad & q_{0.0}\\
C_{25}: &\quad \{a_{1.0} \vee b_{1.0} \vee c_{0.482}\} &\leftarrow\quad & q_{0.0}\\
C_{26}: &\quad \{a_{1.0} \vee b_{0.863} \vee c_{0.482}\} &\leftarrow\quad & q_{1.0}\\
C_{34}: &\quad \{a_{1.0} \vee b_{0.431} \vee c_{1.0}\} &\leftarrow\quad & q_{1.0}\\
C_{35}: &\quad \{a_{1.0} \vee b_{0.431} \vee c_{1.0}\} &\leftarrow\quad & q_{1.0}\\
C_{36}: &\quad \{a_{1.0} \vee b_{1.0} \vee c_{1.0}\} &\leftarrow\quad & q_{0.0}\\
C_{45}: &\quad \{a_{0.301} \vee b_{0.0} \vee c_{0.0}\} &\leftarrow\quad & q_{0.0}\\
C_{46}: &\quad \{a_{0.301} \vee b_{1.0} \vee c_{0.0}\} &\leftarrow\quad & q_{1.0}\\
C_{56}: &\quad \{a_{0.0} \vee b_{1.0} \vee c_{0.0}\} &\leftarrow\quad & q_{1.0}
\end{aligned}
$$

Due to the properties of implicators, all clauses with $q_{0.0}$ may be removed without influencing the final outputted reduct, hence the clause list can be reduced to (with duplicates removed):

$$
\begin{aligned}
C_{12}: &\quad \{a_{0.0} \vee b_{1.0} \vee c_{1.0}\} &\leftarrow\quad & q_{1.0}\\
C_{14}: &\quad \{a_{1.0} \vee b_{0.0} \vee c_{1.0}\} &\leftarrow\quad & q_{1.0}\\
C_{23}: &\quad \{a_{0.301} \vee b_{1.0} \vee c_{0.964}\} &\leftarrow\quad & q_{1.0}\\
C_{26}: &\quad \{a_{1.0} \vee b_{0.863} \vee c_{0.482}\} &\leftarrow\quad & q_{1.0}\\
C_{34}: &\quad \{a_{1.0} \vee b_{0.431} \vee c_{1.0}\} &\leftarrow\quad & q_{1.0}\\
C_{46}: &\quad \{a_{0.301} \vee b_{1.0} \vee c_{0.0}\} &\leftarrow\quad & q_{1.0}\\
C_{56}: &\quad \{a_{0.0} \vee b_{1.0} \vee c_{0.0}\} &\leftarrow\quad & q_{1.0}
\end{aligned}
$$

The DPLL-SOLVE algorithm is then used to determine the minimal reduct. Clause C_{56} is a unit clause (here feature b is a core attribute), so variable b is set to `true`. The UPDATE-TRUE procedure is then executed, removing all clauses that are now maximally satisfied as a result of this assignment:

$$\begin{aligned} C_{14}: &\quad \{a_{1.0} \vee 0.0 \vee c_{1.0}\} &\leftarrow\ & q_{1.0} \\ C_{26}: &\quad \{a_{1.0} \vee 0.863 \vee c_{0.482}\} &\leftarrow\ & q_{1.0} \\ C_{34}: &\quad \{a_{1.0} \vee 0.431 \vee c_{1.0}\} &\leftarrow\ & q_{1.0} \end{aligned}$$

Next, line (12) of the algorithm is executed. There are no unit clauses, so line (10) is reached and the variable a is chosen as the sum of its fuzzy discernibilities is greater than that of c. With a set to `true`, all clauses have been maximally satisfied and $\{a, b\}$ is returned. The algorithm terminates at this point, as the choice of setting b to `false` is unavailable because b was chosen via a unit clause (and hence must be set to `true`).

5.3. Simplification

Crisp discernibility matrices are simplified by removing duplicate entries and clauses that are supersets of others. This can be achieved for fuzzy discernibility matrices: duplicate clauses can be removed as a subset that satisfies one clause to a certain degree will always satisfy the other to the same degree. Also, clauses whose decision component is zero can also be removed due to the inherent properties of fuzzy implication.

A further degree of simplification is obtained by an extension of the crisp approach where clauses that are supersets of others are removed (termed absorption), for the fuzzy case:

$$S(C_{ij}, C_{kl}) = \frac{\sum_{a \in \mathbb{C}} \mathcal{T}(\mu_{C_{ij}}(a), \mu_{C_{kl}}(a))}{\sum_{a \in \mathbb{C}} \mu_{C_{ij}}(a)} \tag{28}$$

If $S(C_{ij}, C_{kl}) = 1$ then clause C_{kl} is subsumed by clause C_{ij} and can be removed. Of course, further simplification techniques from the literature [37] on crisp discernibility matrices and functions could be extended and applied, but only fuzzy absorption is considered here.

Returning to the example, the original set of clauses used as input to DPLL-SOLVE are:

$$\begin{array}{lrcl}
C_{12}: & \{a_{0.0} \vee b_{1.0} \vee c_{1.0}\} & \leftarrow & q_{1.0} \\
C_{14}: & \{a_{1.0} \vee b_{0.0} \vee c_{1.0}\} & \leftarrow & q_{1.0} \\
C_{23}: & \{a_{0.301} \vee b_{1.0} \vee c_{0.964}\} & \leftarrow & q_{1.0} \\
C_{26}: & \{a_{1.0} \vee b_{0.863} \vee c_{0.482}\} & \leftarrow & q_{1.0} \\
C_{34}: & \{a_{1.0} \vee b_{0.431} \vee c_{1.0}\} & \leftarrow & q_{1.0} \\
C_{46}: & \{a_{0.301} \vee b_{1.0} \vee c_{0.0}\} & \leftarrow & q_{1.0} \\
C_{56}: & \{a_{0.0} \vee b_{1.0} \vee c_{0.0}\} & \leftarrow & q_{1.0}
\end{array}$$

The fuzzy absorption simplification process compares each pair of clauses and removes those that are subsumed. For example, clauses C_{46} and C_{23}:

$$\begin{aligned}
S(C_{46}, C_{23}) &= \frac{\sum_{a \in \mathbb{C}} \mathcal{T}(\mu_{C_{46}}(a), \mu_{C_{23}}(a))}{\sum_{a \in \mathbb{C}} \mu_{C_{46}}(a)} \\
&= \frac{\mathcal{T}(0.301, 0.301) + \mathcal{T}(1, 1) + \mathcal{T}(0, 0.964)}{1.301}
\end{aligned}$$

In this case, $S(C_{46}, C_{23}) = 1$ so clause C_{23} can be removed. Any assignment of truth values to variables such that C_{46} is maximally satisfied also implies that C_{23} is maximally satisfied. The reverse is not true, so C_{23} provides no further information than that already possessed by C_{46}. Applying this

process to all clauses results in:

$$
\begin{aligned}
C_{14}: &\quad \{a_{1.0} \vee b_{0.0} \vee c_{1.0}\} &\leftarrow\ q_{1.0} \\
C_{26}: &\quad \{a_{1.0} \vee b_{0.863} \vee c_{0.482}\} &\leftarrow\ q_{1.0} \\
C_{56}: &\quad \{a_{0.0} \vee b_{1.0} \vee c_{0.0}\} &\leftarrow\ q_{1.0}
\end{aligned}
$$

The number of clauses has been reduced to 3 from the original 7, and the DPLL search from this point is straightforward, resulting in the reduct $\{a, b\}$. The subset $\{b, c\}$ is also a reduct, as discovered by the original FRFS algorithm [18]. Again, use of the Łukasiewicz t-conorm can lead to a clause being maximally satisfied with the choice of several sub-maximal features. In this case, $\mathcal{S}(0.863, 0.482) = 1$, so $\{b, c\}$ is a valid fuzzy-rough reduct.

This simplification process is effective, but computationally expensive: the process must compare each clause with every other clause in the clause list. For the worst case, $c = (n^2 - n)/2$ clauses are generated initially, so $(c^2 - c)/2$ clause comparisons are to be made. This can be reduced by integrating the simplification into the discernibility matrix construction process; as clauses are generated, they are checked for fuzzy absorption against existing clauses and vice versa.

Another simplification method for crisp discernibility matrices is to exploit local strong compressibility [38]. If a subset of attributes is simultaneously present or absent in the set of clauses, then they can be replaced by a single representative attribute (since all attributes in this class possess exactly the same information, then with one of the attributes selected, the rest are redundant). Figure 5 shows the extension of this concept to the fuzzy case, where attribute a_1 is tested to see if it is redundant in the presence of attribute a_2.

5.4. Unsupervised selection

The use of rough and fuzzy-rough sets for unsupervised feature selection has also been investigated [30]. This is achieved in this framework by setting all decision components to 1, specifying that all pairs of objects must be distinguishable.

6. Evaluation

This section presents the initial experimental evaluation of the proposed method on nominal and real-valued benchmark datasets from [4] and [17]. For the statistical comparisons, paired t-testing was used (with significance level 0.05) and the test base is the first algorithm in each table.

JRip [10] was employed for the purpose of evaluating the resulting subsets. JRip learns propositional rules by repeatedly growing rules and pruning them. During the growth phase, features are added greedily until a termination condition is satisfied. Features are then pruned in the next phase subject to a given pruning metric. Once the ruleset is generated, a further optimization is performed where classification rules are evaluated and deleted based on their performance on randomized data. For the experiments themselves, 10×10-fold cross validation was performed, where each feature selection algorithm is applied to the training folds and then the resulting subsets used to reduce the test fold each time.

6.1. Crisp Datasets

The considered feature selection methods are: the proposed method (RSAR-SAT), standard rough set attribute reduction (RSAR), a boundary region-based method that employs hillclimbing (BR), a discernibility function-based method (Disc), RSAR employing a genetic algorithm for

search (GA), RSAR employing Particle Swarm Optimization for search (PSO), and finally the Johnson Reducer using the clauses generated from the calculation of the discernibility matrix (JR)[1]. For the GA, the parameters empirically selected for use were: 50 generations, population size of 100 individuals, probability of mutation 0.033, probability of crossover 0.6. For the PSO search method, the parameters used were: 50 generations, 100 particles.

The nominal-valued datasets used in the experimentation for this section are given in Table 4.

Table 4: Number of features possessed by each dataset

Dataset	Features	Objects
M-of-n	14	1000
Exactly	14	1000
Exactly2	14	1000
Heart	14	294
Vote	17	300
Credit	21	1000
LED	25	2000
Derm	35	366
Derm2	35	358
WQ	39	521
Lung	57	32

The average subset sizes found over all folds in the cross validation procedure can be seen in Table 5; the average times of the execution of the algorithms are presented in Table 6. Those methods employing a greedy

[1]All evaluation measures described in this paper have been implemented in Weka [43]. The program can be downloaded from http://users.aber.ac.uk/rkj/

hill-climbing strategy (i.e., RSAR, BR, Disc and JR) often fail to find the optimal reducts. For RSAR, there is only one dataset for which it always finds the optimal subset, demonstrating the inadequacy of this method in returning the smallest subsets. Of the hill-climbing methods, Disc and JR appear to utilise more informed heuristics.

Perhaps surprising is the relatively poor performance of the GA-based method. This may be due to the complexity of the problem of trying to find optimal subsets. A larger number of generations and population size would help, however this would greatly affect the time taken (which is already significantly greater than that for RSAR-SAT overall). PSO performs much better for this task.

Interestingly, RSAR-SAT is often faster than the other algorithms compared here. This is both surprising and pleasing in that the algorithm not only has to locate the optimum but then, unlike competitor methods, completes the search to prove optimality. Note that the time taken for the method includes the construction of the discernibility matrix and simplification of the resulting clauses using the techniques described in this paper, as well as the search itself.

Table 7 shows the resulting average classification accuracies across all folds. As can be seen from this table, the reduction methods perform comparably. This shows that, in general, the smallest reducts possess similar amounts of information as larger ones. This is an interesting finding in further support of the argument presented in [19].

6.2. Comparison with Leading Feature Selectors (I)

In order to further assess the utility of this work, RSAR-SAT was compared against three leading feature selectors: a correlation-based filter method

Table 5: Average number of features selected

Dataset	RSAR-SAT	RSAR	BR	Disc	GA	PSO	JR
M-of-n	6.00	7.00 ∘	7.00 ∘	6.00	7.16 ∘	6.00	6.00
Exactly	6.00	8.08 ∘	8.08 ∘	7.02 ∘	8.34 ∘	6.00	6.00
Exactly2	10.00	10.00	10.00	10.44 ∘	10.68 ∘	10.00	10.00
Heart	5.78	6.72 ∘	6.72 ∘	5.80	6.66 ∘	6.00	6.24 ∘
Vote	7.84	9.12 ∘	9.12 ∘	8.46 ∘	8.16	7.90	7.84
Credit	8.00	8.54 ∘	8.54 ∘	8.08	10.20 ∘	9.40 ∘	9.70 ∘
LED	5.00	6.00 ∘	5.00	17.68 ∘	8.44 ∘	5.10	8.02 ∘
Derm	5.32	6.86 ∘	5.64	5.66	9.62 ∘	8.42 ∘	5.80 ∘
Derm2	7.86	9.96 ∘	9.78 ∘	8.22	12.56 ∘	9.92 ∘	8.38 ∘
WQ	11.04	14.10 ∘	12.84 ∘	12.42 ∘	15.54 ∘	13.56 ∘	12.84 ∘
Lung	3.22	3.72 ∘	3.72 ∘	3.38	13.00 ∘	4.72 ∘	3.38
Average	6.91	8.19	7.86	8.47	10.03	7.91	7.65

•, ∘ statistically significant improvement or degradation

(CFS) [14], a consistency-based method (Cons) [24], and Relief-F [21]. The results can be seen in Table 8 and Table 9. RSAR-SAT often finds the smallest subsets for the datasets whilst retaining classification performance when JRip is applied. The approach performs particularly well for the Exactly dataset, improving the classification accuracy from 69.26% to 99.40%. For the datasets Derm and Derm2 the performance is quite poor, however the method selects fewest features for these datasets.

6.3. Real-valued Datasets

The algorithms were applied to the datasets given in Table 10. The number of conditional features ranges from 10 to 39 over the datasets. The methods used in the comparison were the fuzzy dependency (FRFS), fuzzy boundary region (FBR) and fuzzy discernibility (FD) [18] measures, all using

Table 6: Selection time (ms)

Dataset	RSAR-SAT	RSAR	BR	Disc	GA	PSO	JR
M-of-n	2881.96	7897.68 ○	9646.90 ○	3972.98 ○	25926.44 ○	75330.68 ○	2914.36
Exactly	2857.82	8423.10 ○	10373.86 ○	4399.76 ○	25433.90 ○	68913.04 ○	2890.56
Exactly2	2845.74	9969.76 ○	14951.06 ○	4995.08 ○	24076.52 ○	74388.24 ○	2870.34
Heart	266.10	655.46 ○	1319.96 ○	280.72	2281.16 ○	5701.30 ○	262.54
Vote	333.08	1383.76 ○	1576.32 ○	506.68 ○	3370.54 ○	10297.40 ○	340.88
Credit	4781.34	17215.16 ○	59061.88 ○	6823.10 ○	38544.66 ○	127809.42 ○	4756.32
LED	32921.90	66590.60 ○	57996.62 ○	115483.32 ○	218065.26 ○	689636.42 ○	32078.72
Derm	1542.76	3616.42 ○	13042.28 ○	1849.54 ○	6077.22 ○	21198.08 ○	1170.86 ●
Derm2	2170.14	5145.22 ○	11643.22 ○	2409.72	5419.88 ○	19217.94 ○	1158.00 ●
WQ	54131.96	16157.84 ●	92534.58 ○	7795.88 ●	14440.94 ●	51416.74	3032.88 ●
Lung	169.54	29.16 ●	30.26 ●	9.08 ●	9.72 ●	204.48	24.98 ●
Average	9536.58	12462.20	24743.36	13502.35	33058.75	104010.34	4681.86

●, ○ statistically significant improvement or degradation

a greedy hill-climbing search process. Again, two alternative search methods were used with the fuzzy dependency measure, genetic algorithms (GA) and particle swarm optimization (PSO), in order to search for the smallest subsets. Additionally the Johnson Reducer (JR) was applied, using the fuzzy clauses generated by the construction of the fuzzy discernibility matrix.

The three measures that employ a hill-climbing search strategy all locate reducts of a small size, though not necessarily globally optimal. The boundary region measure and discernibility measure appear to be more informed heuristics. The difficulty of finding globally minimal reducts can be seen in the results for the more advanced search strategies (GA and PSO). Neither method consistently finds such reducts: PSO always finds the global minimum for two datasets (Australian and Glass), the GA approach only finds

Table 7: Classification accuracy

Dataset	Unreduced	RSAR-SAT	RSAR	BR	Disc	GA	PSO	JR
M-of-n	97.88	99.12	98.88	98.88	99.12	99.04	99.12	99.12
Exactly	69.26	99.40 ○	92.18 ○	92.18 ○	95.46 ○	90.48 ○	99.40 ○	99.40 ○
Exactly2	73.52	73.66	73.66	73.66	73.50	73.82	73.66	73.66
Heart	80.64	76.44	80.14	80.14	76.80	77.83	77.75	76.18
Vote	94.67	94.40	94.40	94.40	94.40	94.33	94.33	94.40
Credit	70.78	70.32	70.52	70.52	70.64	69.96	71.40	70.80
LED	100.00	100.00	100.00	100.00	96.99	100.00	100.00	100.00
Derm	90.28	64.97 •	80.60 •	59.41 •	59.96 •	82.27 •	68.83 •	64.15 •
Derm2	88.78	68.12 •	90.23	89.43	69.89 •	84.91	72.96 •	65.03 •
WQ	68.75	65.52	66.64	66.79	66.07	66.76	66.52	65.65
Lung	76.83	79.33	73.67	73.67	84.17	82.17	83.00	84.17
Average	82.85	81.03	83.72	81.73	80.64	83.78	82.45	81.14

○, • statistically significant improvement or degradation

the minimum for the Glass dataset. Overall the PSO method outperforms the GA approach. However, the reducts found by these methods are not guaranteed to be minimal.

The average time taken by the algorithms when performing selection can be found in Table 12. The timings for FRFS-SAT include the time taken to calculate the fuzzy discernibility matrix as well as the search itself. It can be seen that, in general, FRFS-SAT can find globally optimal reducts in a similar amount of time to the other methods. However, as the dimensionality increases an increasing amount of time is spent verifying that the discovered reduct is indeed globally optimal, which is the case for the Water datasets. Though in principle methods that guarantee optimality would not scale as well as methods that do not, it does not appear to be a problem for

Table 8: Average number of features selected

Dataset	RSAR-SAT	CFS	Cons	Relief-F
M-of-n	6.00	7.42 ○	4.96	6.86 ○
Exactly	6.00	5.66	13.00 ○	7.92 ○
Exactly2	10.00	4.64 ●	13.00 ○	12.98 ○
Heart	5.78	5.70	7.56 ○	12.42 ○
Vote	7.84	2.54 ●	7.72	15.94 ○
Credit	8.00	3.30 ●	9.60 ○	19.98 ○
LED	5.00	12.94 ○	5.00	16.38 ○
Derm	5.32	19.20 ○	5.84 ○	33.94 ○
Derm2	7.86	18.64 ○	9.46 ○	34.00 ○
WQ	11.04	21.04 ○	13.70 ○	38.00 ○
Lung	3.22	7.28 ○	3.60 ○	44.96 ○
Average	6.91	9.85	6.13	22.13

●, ○ statistically significant improvement or degradation

benchmark datasets that are representative of practical problems.

The resulting classification accuracies for JRip can be found in Table 13. From this, it can be seen that FRFS-SAT finds the globally optimal reduct for each dataset without a statistically significant loss in classification accuracy.

6.4. Comparison with Leading Feature Selectors (II)

Here, the fuzzy approach is compared with the leading feature selectors whose details can be found in Section 6.2. The results can be seen in Table 14 and Table 15. Again, the method often produces the smallest subsets across the datasets with no statistically significant loss of classification accuracy.

Table 9: Classification accuracy

Dataset	Unreduced	RSAR-SAT	CFS	Cons	Relief-F
M-of-n	97.88	99.12	96.02	93.40	98.78
Exactly	69.26	99.40 ∘	67.14	68.80	93.08 ∘
Exactly2	73.52	73.66	75.60 ∘	75.80 ∘	73.80
Heart	80.64	76.44	81.51	80.64	80.15
Vote	94.67	94.40	94.27	94.40	94.53
Credit	70.78	70.32	72.90	71.66	70.96
LED	100.00	100.00	100.00	100.00	100.00
Derm	90.28	64.97 •	89.78	72.89 •	89.89
Derm2	88.78	68.12 •	90.39	85.41	89.16
WQ	68.75	65.52	67.72	67.22	68.06
Lung	76.83	79.33	81.50	79.83	79.33
Average	82.85	81.03	83.35	80.91	85.25

∘, • statistically significant improvement or degradation

7. Conclusion and Discussion

This paper has presented a new DPLL-based technique for locating and verifying minimal subsets in the rough set, and fuzzy rough contexts. The experimentation presented here has shown that the approach performs well in comparison to a number of existing methods, which often fail to find the smallest subsets. Additional investigations to be carried out include evaluating the proposed work against further well established heuristic-based approaches to reduct finding; typical methods can be found in [16, 22, 28, 37].

This paper has also presented an extension of the discernibility matrix to the fuzzy case, allowing features to belong to entries to a certain degree. Based on this, the propositional satisfiability problem has been extended to allow SAT-style search of the resulting fuzzy clauses. From these, the

Table 10: Number of features possessed by each dataset

Dataset	Features	Objects
Australian	15	690
Cleveland	14	297
Glass	14	214
Heart	10	270
Ionosphere	35	230
Olitos	26	120
Water 2	39	390
Water 3	39	390
Wine	14	178

globally minimal reduct for a dataset can be calculated. This generalisation extends the applicability of the underlying approach from discrete datasets to those with continuous variables.

DPLL resorts to chronological backtracking if the current assignment of variables results in the unsatisfiability of F. Much research has been carried out in developing solution techniques for SAT that draws on related work in solvers for constraint satisfaction problems (CSPs) [26, 27]. Indeed the SAT problem can be translated to a CSP by retaining the set of boolean variables and their $\{0, 1\}$ domains, and to translate the clauses into constraints. Each clause becomes a constraint over the variables in the constraint. Unit propagation can be seen to be a form of forward checking.

In CSPs, more intelligent ways of backtracking have been proposed such as backjumping, conflict-directed backjumping and dynamic backtracking [25]. Many aspects of these have been adapted to the SAT problem solvers. In these solvers, whenever a conflict (dead-end) is reached, a new clause is recorded to prevent the occurrence of the same conflict again during the

Table 11: Average number of features selected

Dataset	FRFS-SAT	FRFS	FBR	FD	GA	PSO	JR
Australian	12.66	12.90	12.90	12.84	12.74	12.66	12.94
Cleveland	7.56	7.64	7.68	7.64	8.10 ∘	7.84	7.78
Glass	8.36	9.00 ∘	8.36	8.36	8.36	8.36	8.36
Heart	7.00	7.06	7.06	7.10	7.50 ∘	7.04	7.40 ∘
Ionosphere	6.00	7.04 ∘	7.04 ∘	7.08 ∘	9.52 ∘	7.26 ∘	7.66 ∘
Olitos	4.98	5.00	5.00	5.00	6.04 ∘	5.08	5.14
Water 2	5.82	6.00	6.00	5.98	7.00 ∘	6.72 ∘	6.30 ∘
Water 3	5.88	6.00	6.00	5.98	7.42 ∘	6.98 ∘	6.08
Wine	4.48	5.00 ∘	4.84 ∘	4.86 ∘	5.06 ∘	4.96 ∘	4.76
Average	6.97	7.29	7.21	7.20	7.97	7.43	7.38

•, ∘ statistically significant improvement or degradation

subsequent search. Non-chronological backtracking backs up the search tree to one of the identified causes of failure, skipping over irrelevant variable assignments.

With the addition of intelligent backtracking, it is expected that the method will be able to deal with datasets containing larger numbers of features. As seen in the results presented, the bottleneck in the process is the verification stage - the time taken to confirm that the subset is indeed minimal. This requires an exhaustive search of all subtrees containing fewer variables than the current best solution. Much of this search could be avoided through the use of more intelligent backtracking. Assuming the scale of improvements seen in the SAT solver literature, e.g. [47], this would result in a selection method that can cope with many thousands of features, whilst guaranteeing resultant subset minimality - something that is particularly sought after in feature selection and would scale beyond current heuristic

Table 12: Selection time (ms)

Dataset	FRFS-SAT	FRFS	FBR	FD	GA	PSO	JR
Australian	4228.10	8043.88 ○	21868.92 ○	3588.94 •	15009.24 ○	40615.10 ○	4169.14
Cleveland	731.88	1042.94 ○	3709.02 ○	596.90 •	2838.26 ○	6949.82 ○	681.94 •
Glass	459.50	370.54 •	1450.74 ○	230.12 •	777.18 ○	2554.00 ○	445.74
Heart	614.46	822.62 ○	1758.84 ○	388.80 •	2362.04 ○	5709.64 ○	528.30 •
Ionosphere	19200.42	2127.30 •	3944.62 •	913.96 •	2329.36 •	11055.88 •	1430.36 •
Olitos	2299.84	281.28 •	867.04 •	160.74 •	136.22 •	1813.42 •	343.18 •
Water 2	87519.50	5592.50 •	12539.02 •	1804.68 •	1080.50 •	23721.64 •	5395.90 •
Water 3	107178.90	5590.70 •	15276.54 •	2049.82 •	2808.04 •	23233.98 •	5410.94 •
Wine	649.58	302.06 •	741.34 ○	153.48 •	903.24 ○	2166.72 ○	402.90 •
Average	24764.69	2685.98	6906.23	1098.60	3138.23	13091.13	2089.82

•, ○ statistically significant improvement or degradation

methods.

Active research regarding the fuzzified method includes further experimental investigation, including analysis of the impact of the choice of relations and connectives. Additionally, the development of fuzzy discernibility matrices here allows the extension of many existing crisp techniques for the purposes of finding fuzzy-rough reducts. In particular, other SAT solution techniques may be applied that should be able to discover such subsets, guaranteeing their global minimality. The performance may also be improved through simplifying the fuzzy discernibility function further. This could be achieved by considering the properties of the fuzzy connectives and removing clauses that are redundant in the presence of others.

A. FRFS Reducts are Fuzzy-Rough Reducts

Theorem 1. *FRFS-SAT reducts are fuzzy-rough reducts. Suppose that* $P \subseteq$

Table 13: Classification accuracy

Dataset	Unreduced	FRFS-SAT	FRFS	FBR	FD	GA	PSO	JR
Australian	85.16	84.93	85.16	85.16	85.28	84.93	84.09	85.22
Cleveland	54.03	54.21	54.62	54.68	54.75	54.75	53.48	54.01
Glass	67.17	65.25	67.17	65.25	65.25	65.25	65.25	65.25
Heart	72.96	72.22	74.15	74.15	74.15	74.74	72.07	73.78
Ionosphere	87.57	85.48	87.74	87.74	85.22	83.04	87.48	86.09
Olitos	68.50	61.00	62.83	63.67	62.50	60.83	69.50	60.83
Water 2	82.15	82.00	83.28	83.28	82.00	83.18	79.08	82.00
Water 3	82.72	81.18	81.23	81.23	81.95	81.44	74.82 •	81.44
Wine	93.54	90.24	91.46	92.56	90.42	90.33	90.08	90.37
Average	77.09	75.17	76.41	76.41	75.72	75.39	75.09	75.44

○, • statistically significant improvement or degradation

$\mathbb{C}$, *a is an arbitrary conditional feature that belongs to the dataset and q is the decision feature. If P maximally satisfies the fuzzy discernibility function then P is a fuzzy-rough reduct.*

Proof. The fuzzy positive region for a subset P is

$$\mu_{POS_{R_P}(Q)}(x) = \sup_{X \in \mathbb{U}/Q} \inf_{y \in \mathbb{U}} \{\mu_{R_P}(x, y) \to \mu_X(y)\}$$

The dependency function is maximized when each x belongs maximally to the fuzzy positive region. Hence,

$$\inf_{x \in \mathbb{U}} \sup_{X \in \mathbb{U}/Q} \inf_{y \in \mathbb{U}} \{\mu_{R_P}(x, y) \to \mu_X(y)\}$$

is maximized only when P is a fuzzy-rough reduct. This can be rewritten as the following:

$$\inf_{x,y \in \mathbb{U}} \{\mu_{R_P}(x, y) \to \mu_{R_q}(x, y)\}$$

Table 14: Average number of features selected

Dataset	FRFS-SAT	CFS	Cons	Relief-F
Australian	12.66	6.46 •	12.72	13.12 ◦
Cleveland	7.56	7.86	10.06 ◦	13.00 ◦
Glass	8.36	6.30 •	6.78 •	9.00 ◦
Heart	7.00	7.70	10.26 ◦	12.98 ◦
Ionosphere	6.00	11.16 ◦	7.64 ◦	33.00 ◦
Olitos	4.98	15.02 ◦	10.00 ◦	25.00 ◦
Water 2	5.82	9.06 ◦	12.38 ◦	37.10 ◦
Water 3	5.88	10.80 ◦	10.54 ◦	38.00 ◦
Wine	4.48	10.84 ◦	4.48	13.00 ◦
Average	6.97	9.47	9.43	21.58

•, ◦ statistically significant improvement or degradation

when using a fuzzy similarity relation in the place of crisp decision concepts, as $\mu_{[x]_R} = \mu_R(x, y)$ [13]. Each $\mu_{R_P}(x, y)$ is constructed from the t-norm of its constituent relations:

$$\inf_{x,y \in \mathbb{U}} \{T_{a \in P}(\mu_{R_a}(x, y)) \rightarrow \mu_{R_q}(x, y)\}$$

This may be reformulated as

$$\inf_{x,y \in \mathbb{U}} \{S_{a \in P}(\mu_{R_a}(x, y) \rightarrow \mu_{R_q}(x, y))\} \tag{29}$$

Considering the fuzzy discernibility matrix approach, the fuzzy discernibility function is maximally satisfied when

$$\{\wedge\{\{\vee C^*_{xy}\} \leftarrow q_{N(\mu_{R_q}(x,y))}\} | 1 \leq y < x \leq |\mathbb{U}|\}$$

is maximized. This can be rewritten as:

$$T_{x,y \in \mathbb{U}}(S_{a \in P}(N(\mu_{R_a}(x, y))) \leftarrow N(\mu_{R_q}(x, y)))$$

Table 15: Classification accuracy

Dataset	Unreduced	FRFS-SAT	CFS	Cons	Relief-F
Australian	85.16	84.93	86.87	84.70	84.70
Cleveland	54.03	54.21	54.09	55.23	54.08
Glass	67.17	65.25	65.24	67.02	65.67
Heart	72.96	72.22	73.33	74.67	75.11
Ionosphere	87.57	85.48	88.17	86.26	87.22
Olitos	68.50	61.00	69.00	67.83	67.67
Water 2	82.15	82.00	83.44	83.38	83.28
Water 3	82.72	81.18	83.69	83.33	82.56
Wine	93.54	90.24	92.34	90.80	92.09
Average	77.09	75.17	77.35	77.03	76.93

$\circ$, $\bullet$ statistically significant improvement or degradation

because each clause C_{xy} is generated by considering the fuzzy similarity of values of each pair of objects x, y. Through the properties of the fuzzy connectives, this may be rewritten as:

$$T_{x,y\in\mathbb{U}}(S_{a\in P}(\mu_{R_a}(x,y) \rightarrow \mu_{R_q}(x,y))) \tag{30}$$

When this is maximized, (29) is maximized and so the subset P must be a fuzzy-rough reduct.

□

References

[1] A.A. Bakar, M.N. Sulaiman, M. Othman, M.H. Selamat. IP algorithms in compact rough classification modeling, Intelligent Data Analysis, vol. 5, no. 5, pp. 419–429. 2001.

[2] J. Bazan, A. Skowron, P. Synak. Dynamic reducts as a tool for extracting laws from decision tables. In Z. W. Ras, M. Zemankova (Eds.), Proceedings of the Eighth Symposium on Methodologies for Intelligent Systems, Lecture Notes in Artificial Intelligence 869, Springer-Verlag, pp. 346–355, 1994.

[3] A.T. Bjorvand and J. Komorowski. Practical applications of genetic algorithms for efficient reduct computation, In Proceedings of the 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics, vol. 4, pp. 601–606, 1997.

[4] C.L. Blake and C.J. Merz. UCI Repository of machine learning databases. Irvine, University of California, 1998. `http://www.ics.uci.edu/~mlearn/.`

[5] Y. Caballero, R. Bello, D. Alvarez, and M.M. Garca. Two new feature selection algorithms with Rough Sets Theory, IFIP AI 2006, pp. 209–216, 2006.

[6] D. Chen, L. Zhang, S. Zhao, Q. Hu, and P. Zhu. A Novel Algorithm of Finding Reducts with Fuzzy Rough Set, IEEE Transactions on Fuzzy Systems, vol. 20, no. 2, pp. 385-389, 2012.

[7] Y. Chen, D. Miao and R. Wang. A rough set approach to feature selection based on ant colony optimization, Pattern Recognition Letters, vol. 31, no. 3, pp.226-233, 2010.

[8] A. Chouchoulas and Q. Shen. Rough set-aided keyword reduction for text categorisation, Applied Artificial Intelligence, vol. 15, no. 9, pp. 843–873, 2001.

[9] M. De Cock, C. Cornelis, and E.E. Kerre. Fuzzy Rough Sets: The Forgotten Step, IEEE Transactions on Fuzzy Systems, vol. 15, no. 1, pp. 121–130, 2007.

[10] W.W. Cohen. Fast effective rule induction, In Proceedings of the 12th International Conference on Machine Learning, pp. 115–123, 1995.

[11] M. Dash and H. Liu. Feature Selection for Classification. Intelligent Data Analysis, vol. 1, no. 3, pp. 131–156, 1997.

[12] M. Davis, G. Logemann and D. Loveland. A machine program for theorem proving, Communications of the ACM, vol. 5, pp. 394–397, 1962.

[13] D. Dubois and H. Prade. Putting Rough Sets and Fuzzy Sets Together, Intelligent Decision Support, pp. 203–232, 1992.

[14] M.A. Hall, L.A. Smith. Practical feature subset selection for machine learning, Australian Computer Science Conference, pp. 181–191, 1998.

[15] H.H. Hoos and T. Stützle. Towards a characterisation of the behaviour of stochastic local search algorithms for SAT, Artificial Intelligence, vol. 112, pp. 213–232, 1999.

[16] R. Jensen and Q. Shen. Semantics-preserving dimensionality reduction: rough and fuzzy-rough based approaches, IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 12, pp. 1457–1471, 2004.

[17] R. Jensen and Q. Shen. Computational Intelligence and Feature Selection: Rough and Fuzzy Approaches, Wiley-IEEE Press, 2008.

[18] R. Jensen and Q. Shen. New approaches to fuzzy-rough feature selec-

tion, IEEE Transactions on Fuzzy Systems, vol. 17, no. 4, pp. 824–838, 2009.

[19] R. Jensen and Q. Shen. Are more features better? A response to attributes reduction using fuzzy rough sets, IEEE Transactions on Fuzzy Systems, vol. 17, no. 6, pp. 1456–1458, 2009.

[20] N. Jiao, D. Miao and J. Zhou. Two novel feature selection methods based on decomposition and composition. Expert Systems with Applications: An International Journal, vol. 37, no. 12, pp. 7419–7426, 2010.

[21] I. Kononenko, E. Simec and M. Robnik-Sikonja. Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF, Applied Intelligence, vol. 7, pp. 39-55, 1997.

[22] M. Kryszkiewicz. Comparative Study of Alternative Types of Knowledge Reduction in Inconsistent Systems, International Journal of Intelligent Systems, vol. 16, no. 1, pp. 105–120, 2001.

[23] T.Y. Lin, P. Yin. Heuristically Fast Finding of the Shortest Reducts, Rough Sets and Current Trends in Computing, Lecture Notes in Computer Science, vol. 3066/2004, pp. 465–470, 2004.

[24] H. Liu and R. Setiono. A probabilistic approach to feature selection - A filter solution, In: 13th International Conference on Machine Learning (ICML'96), pp. 319–327, 1996.

[25] I. Miguel and Q. Shen. Dynamic Flexible Constraint Satisfaction, Applied Intelligence, 13(3), pp. 231–245, 2000.

[26] I. Miguel and Q. Shen. Solution Techniques for Constraint Satisfaction

Problems: Foundations, Artificial Intelligence Review, 15(4), pp. 243–267, 2001.

[27] I. Miguel and Q. Shen. Fuzzy rrDFCSP and Planning, Artificial Intelligence, 148(1-2), pp. 11–52, 2003.

[28] H.S. Nguyen and A. Skowron. Boolean Reasoning for Feature Extraction Problems, In Proceedings of the 10th International Symposium on Methodologies for Intelligent Systems, pp. 117–126, 1997.

[29] A. Øhrn. Discernibility and Rough Sets in Medicine: Tools and Applications, Department of Computer and Information Science. Trondheim, Norway, Norwegian University of Science and Technology: 239, 1999.

[30] N. Mac Parthaláin and R. Jensen. Unsupervised Fuzzy-Rough Set-based Dimensionality Reduction, submitted to Information Sciences.

[31] Z. Pawlak. Rough Sets: Theoretical Aspects of Reasoning About Data, Kluwer Academic Publishing, Dordrecht. 1991.

[32] Z. Pawlak and A. Skowron. Rough sets and Boolean reasoning, Information Sciences, vol. 177, no. 1, pp. 41–73, 2007.

[33] L. Polkowski. Rough Sets: Mathematical Foundations, Advances in Soft Computing, Physica Verlag, Heidelberg, Germany, 2002.

[34] Y. Qian, J. Liang, W. Pedrycz, C. Dang. Positive approximation: An accelerator for attribute reduction in rough set theory, Artificial Intelligence, vol. 174, pp. 597–618, 2010.

[35] A.M. Radzikowska and E.E. Kerre. A comparative study of fuzzy rough sets, Fuzzy Sets and Systems, vol. 126, no. 2, pp. 137–155, 2002.

[36] A. Skowron and C. Rauszer. The discernibility matrices and functions in Information Systems, In: R. Slowinski (Ed.), Intelligent Decision Support, Kluwer Academic Publishers, Dordrecht, pp. 331–362. 1992.

[37] J.A. Starzyk, D.E. Nelson, and K. Sturtz. Reduct Generation in Information Systems, Bulletin of the International Rough Set Society, vol. 3, no. 1–2, pp. 19–22. 1999.

[38] J.A. Starzyk, D.E. Nelson, and K. Sturtz. A Mathematical Foundation for Improved Reduct Generation in Information Systems, Knowledge and Information Systems, vol. 2, no. 2, pp. 131–146, 2000.

[39] G.C.Y. Tsang, D. Chen, E.C.C. Tsang, J.W.T. Lee, and D.S. Yeung. On attributes reduction with fuzzy rough sets, IEEE International Conference on Systems, Man and Cybernetics, vol. 3, pp. 2775–2780, 2005.

[40] M. Wallace, Y. Avrithis and S. Kollias. Computationally efficient sup-t transitive closure for sparse fuzzy binary relations, Fuzzy Sets and Systems, vol. 157, no. 3, pp. 341–372, 2006.

[41] J. Wang, J. Wang. Reduction algorithms based on discernibility matrix: the ordered attributes method, Journal of Computer Science and Technology, vol. 16, pp. 489-504, 2001.

[42] X. Wang, J. Yang, X. Teng, W. Xia, and R. Jensen. Feature Selection based on Rough Sets and Particle Swarm Optimization, Pattern Recognition Letters, vol. 28, no. 4, pp. 459-471, 2007.

[43] I. H. Witten and E. Frank, Data Mining: Practical machine learning tools with Java implementations, Morgan Kaufmann Publishers, San Francisco, 2000.

[44] J. Wróblewski. Finding minimal reducts using genetic algorithms, In Proceedings of the 2nd Annual Joint Conference on Information Sciences, pp. 186–189, 1995.

[45] M. Yang and P. Yang. A novel condensing tree structure for rough set feature selection, Neurocomputing, vol. 71, nos. 4-6, pp. 1092–1100, 2008.

[46] Y. Yao and Y. Zhao. Discernibility matrix simplification for constructing attribute reducts, Information Sciences, vol. 179, no. 7, pp. 867–882, 2009.

[47] L. Zhang and S. Malik. The Quest for Efficient Boolean Satisfiability Solvers, Proceedings of the 18th International Conference on Automated Deduction, pp. 295–313, 2002.

[48] N. Zhong, J. Dong and S. Ohsuga. Using Rough Sets with Heuristics for Feature Selection, Journal of Intelligent Information Systems, vol. 16, no. 3, pp. 199–214, 2001.

DPLL-SOLVE(d, CL, D).

d, the current depth of search;

CL, the current list of clauses;

D, the depth of the best reduct found so far (initially $|\mathbb{C}|$).

(1) **if** ($d \geq D$) **or** ($CL ==$ null)

(2) // Further search down this branch is unnecessary

(3) **else if** (CL.size() == 0) **and** ($d < D$)

(4) $D \leftarrow d$

(5) **output** current assignment

(6) **else if** (CL contains a unit clause $\{l\}$)

(7) $CL' \leftarrow$ unitPropagate(CL)

(8) DPLL-SOLVE($d+1$,CL',D)

(9) **else**

(10) $x \leftarrow$ selectLiteral(CL)

(11) $CL' \leftarrow$ UPDATE-TRUE(CL, x)

(12) DPLL-SOLVE($d+1$,CL',D)

(13) $CL' \leftarrow$ UPDATE-FALSE(CL, x)

(14) DPLL-SOLVE(d,CL',D)

Figure 2: The DPLL-SOLVE algorithm

UPDATE-TRUE(CL, x).

CL, the current clause list;

x, the variable to be set to `true`.

```
(1)  CL' ← ∅
(2)  foreach C ∈ CL
(3)      if (!isSatisfied(C))
(4)          CL' ← CL' ∪ C
(5)  return CL'
```

Figure 3: The UPDATE-TRUE algorithm

UPDATE-FALSE(CL, x).

CL, the current clause list;

x, the variable to be set to `false`.

```
(1)  CL' ← ∅
(2)  foreach C ∈ CL
(3)      if (|C|==0) or (!isSatisfiable(C))
(4)          return null //Further search is pointless
(5)      else CL' ← CL' ∪ C
(6)  return CL'
```

Figure 4: The UPDATE-FALSE algorithm

FUZZY-COMPRESSIBILITY(CL, a_1, a_2).

CL, the current clause list;

a_1,a_2, conditional attributes.

```
(1)  foreach C ∈ CL
(2)      if (S(μ_C(a_1), μ_C(a_2)) > μ_C(a_2))
(3)          return false
(4)  return true
```

Figure 5: The FUZZY-COMPRESSIBILITY algorithm