Rough set based feature selection

Jensen, Richard; Shen, Qiang

Published in:
Rough Computing

DOI:
10.4018/978-1-59904-552-8.ch003

Publication date:
2007

Citation for published version (APA):
Jensen, R., & Shen, Q. (2007). Rough set based feature selection: A review. In A.-E. Hassanien, Z. Suraj, D. Slezak, & P. Lingras (Eds.), *Rough Computing: Theories, Technologies and Applications* (pp. 70-107). Information Science Reference. https://doi.org/10.4018/978-1-59904-552-8.ch003

Rough Set-based Feature Selection: A Review

Abstract: *Feature selection aims to determine a minimal feature subset from a problem domain while retaining a suitably high accuracy in representing the original features. Rough set theory (RST) has been used as such a tool with much success. RST enables the discovery of data dependencies and the reduction of the number of attributes contained in a dataset using the data alone, requiring no additional information. This chapter describes the fundamental ideas behind RST-based approaches and reviews related feature selection methods that build on these ideas. Extensions to the traditional rough set approach are discussed, including recent selection methods based on tolerance rough sets, variable precision rough sets and fuzzy-rough sets. Alternative search mechanisms are also highly important in rough set feature selection. The chapter includes the latest developments in this area, including RST strategies based on hill-climbing, genetic algorithms and ant colony optimization.*

Keywords: Feature Selection, Rough Sets, Data Preprocessing, Data Mining, Rough Set Variations, Search Algorithms.

INTRODUCTION

The main aim of feature selection (FS) is to determine a minimal feature subset from a problem domain while retaining a suitably high accuracy in representing the original features. In many real world problems FS is a must due to the abundance of noisy, irrelevant or misleading features. For instance, by removing these factors, learning from data techniques can benefit greatly. A detailed review of feature selection techniques devised for classification tasks can be found in (Dash & Liu, 1997).

The usefulness of a feature or feature subset is determined by both its *relevancy* and *redundancy*. A feature is said to be relevant if it is predictive of the decision feature(s), otherwise it is irrelevant. A feature is considered to be redundant if it is highly correlated with other features. Hence, the search for a good feature subset involves finding those features that are highly correlated with the decision feature(s), but are uncorrelated with each other.

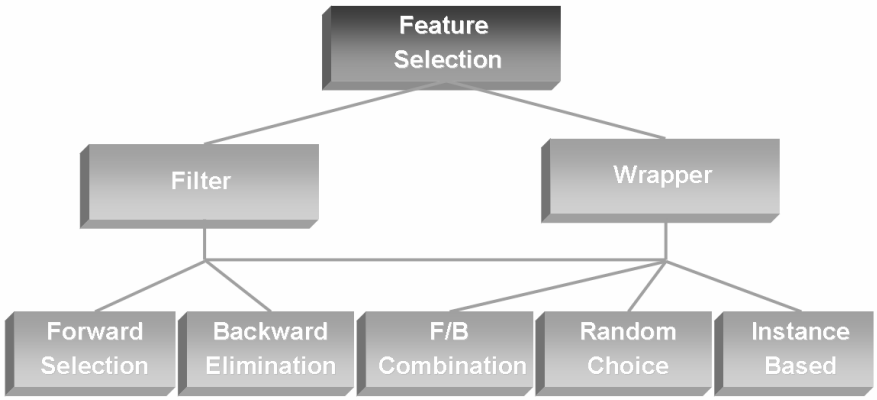

Figure 1: Aspects of feature selection

A taxonomy of feature selection approaches can be seen in Figure 1. Given a feature set size n, the task of FS can be seen as a search for an "optimal" feature subset through the competing 2^n candidate subsets. The definition of what an optimal subset

is may vary depending on the problem to be solved. Although an exhaustive method may be used for this purpose in theory, this is quite impractical for most datasets. Usually FS algorithms involve heuristic or random search strategies in an attempt to avoid this prohibitive complexity. However, the degree of optimality of the final feature subset is often reduced. The overall procedure for any feature selection method is given in Figure 2 (adapted from (Dash & Liu, 1997)).

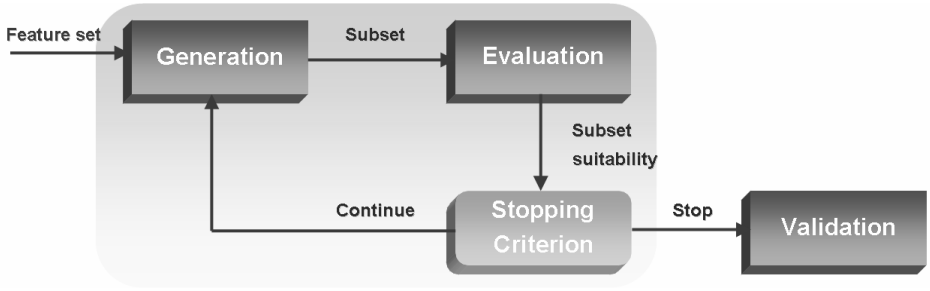

Figure 2: Feature Selection

The generation procedure implements a search method (Langley 1994; Siedlecki & Sklansky, 1988) that generates subsets of features for evaluation. It may start with no features, all features, a selected feature set or some random feature subset. Those methods that start with an initial subset usually select these features heuristically beforehand. Features are added (*forward selection*) or removed (*backward elimination*) iteratively in the first two cases (Dash & Liu, 1997). In the last case, features are either iteratively added or removed or produced randomly thereafter. An alternative selection strategy is to select instances and examine differences in their features. The evaluation function calculates the suitability of a feature subset produced by the generation procedure and compares this with the previous best candidate, replacing it if found to be better.

A stopping criterion is tested every iteration to determine whether the FS process should continue or not. For example, such a criterion may be to halt the FS process when a certain number of features have been selected if based on the generation process. A typical stopping criterion centred on the evaluation procedure is to halt the process when an optimal subset is reached. Once the stopping criterion has been satisfied, the loop terminates. For use, the resulting subset of features may be validated.

Determining subset optimality is a challenging problem. There is always a trade-off in non-exhaustive techniques between subset minimality and subset suitability - the task is to decide which of these must suffer in order to benefit the other. For some domains (particularly where it is costly or impractical to monitor many features), it is much more desirable to have a smaller, less accurate feature subset. In other areas it may be the case that the modelling accuracy (e.g. the classification rate) using the selected features must be extremely high, at the expense of a non-minimal set of features.

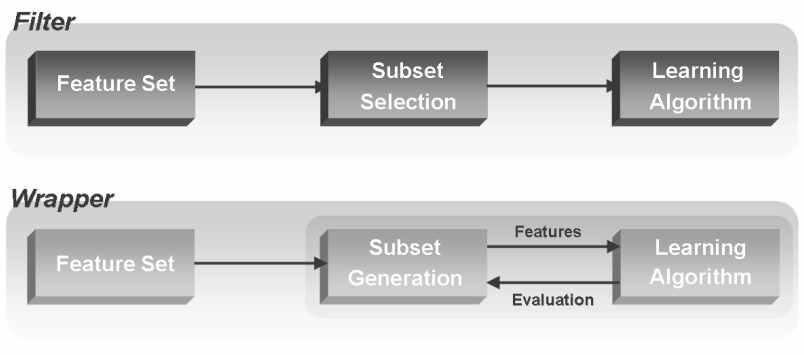

Figure 3 Filter and wrapper methods

Feature selection algorithms may be classified into two categories based on their evaluation procedure (see Figure 3). If an algorithm performs FS independently of any learning algorithm (i.e. it is a completely separate preprocessor), then it is a *filter* approach. In effect, irrelevant attributes are filtered out before induction. Filters tend to be applicable to most domains as they are not tied to any particular induction algorithm.

If the evaluation procedure is tied to the task (e.g. classification) of the learning algorithm, the FS algorithm employs the *wrapper* approach. This method searches through the feature subset space using the estimated accuracy from an induction algorithm as a measure of subset suitability. Although wrappers may produce better results, they are expensive to run and can break down with very large numbers of features. This is due to the use of learning algorithms in the evaluation of subsets, some of which can encounter problems when dealing with large datasets.

This chapter reviews generic filter-based methods to feature selection based on rough set theory. Most developments in the area concentrate on using rough sets within the evaluation function. This can be achieved by employing the rough set dependency degree (including extensions of this measure) or through the use of discernibility functions. The theoretical background and recent approaches for this are found in the next section. Following this, details are given concerning RST-based approaches to handling real-valued and noisy data. Although the techniques reviewed here are primarily for feature selection, their use is not restricted solely to this field. Alternative search mechanisms are presented in the following section that attempt to tackle the problem of locating globally optimal subsets. Finally, the chapter is concluded, including a discussion of future research issues.

ROUGH SET-BASED FEATURE SELECTION

Rough set theory (RST) can be used as 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 (Pawlak, 1991; Polkowski, 2002). Over the past ten years, RST has 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 minimal

information loss. From the dimensionality reduction perspective, informative features are those that are most predictive of the class attribute.

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.

$x \in U$	a	b	c	d	e
0	1	0	2	2	0
1	0	1	1	1	2
2	2	0	0	1	1
3	1	1	0	2	2
4	1	0	2	0	1
5	2	2	0	1	1
6	2	1	1	1	2
7	0	1	1	0	1

Table 1 An example dataset

Dependency Function-based Approaches

RSAR

Central to Rough Set Attribute Reduction (RSAR) (Chouchoulas & Shen, 2001; Jensen & Shen, 2004b) is the concept of indiscernibility. Let $I = (U, A)$ be an information system, where U is a non-empty set of finite objects (the universe) and A is a non-empty finite set of attributes such that $a: U \rightarrow V_a$ for every $a \in A$. V_a is the set of values that attribute a may take. With any $P \subseteq A$ there is an associated equivalence relation $IND(P)$:

Equation 1

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

The partition of U generated by $IND(P)$ is denoted $U/IND(P)$ (or U/P). If $(x, y) \in IND(P)$, then x and y are indiscernible by attributes from P. The equivalence classes of the P-indiscernibility relation are denoted $[x]_P$. 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 U :

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

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

Equation 2

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

Equation 3

$$\overline{P}X = \{x \mid [x]_P \cap X \neq \varnothing\}$$

Let P and Q be equivalence relations over U, then the positive region can be defined as:

Equation 4

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

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

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

Using this definition of the positive region, the rough set degree of dependency of a set of attributes Q on a set of attributes P is defined in the following way:

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

Equation 5

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

In the example, the degree of dependency of attribute $\{e\}$ from the attributes $\{b,c\}$ is:

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

The reduction of attributes is achieved by comparing equivalence relations generated by sets of attributes. Attributes are removed so that the reduced set provides the same predictive capability of the decision feature as the original. A reduct R is defined as a subset of minimal cardinality of the conditional attribute set C such that $\gamma_R(D) = \gamma_C(D)$.

QUICKREDUCT(C,D)
C, the set of all conditional features;
D, the set of decision features.

(1) $R \leftarrow \{\}$
(2) **do**
(3) $T \leftarrow R$
(4) $\forall x \in (C - R)$
(5) **if** $\gamma_{R \cup \{x\}}(D) > \gamma_T(D)$
(6) $T \leftarrow R \cup \{x\}$
(7) $R \leftarrow T$
(8) **until** $\gamma_R(D) = \gamma_C(D)$
(9) **return** R

Figure 4 The QUICKREDUCT Algorithm

The QUICKREDUCT algorithm given in Figure 4, attempts to calculate a reduct without exhaustively generating all possible subsets. 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.

According to the algorithm, the dependency of each attribute is calculated, and the best candidate chosen. Attribute d generates the highest dependency degree, so that attribute is chosen and the sets $\{a,d\}$, $\{b,d\}$ and $\{c,d\}$ are evaluated. This process continues until the dependency of the reduct equals the consistency of the dataset (1 if the dataset is consistent). In the example, the algorithm terminates after evaluating the subset $\{b,d\}$. The generated reduct shows the way of reducing the dimensionality of the original dataset by eliminating those conditional attributes that do not appear in the set.

This, however, is not guaranteed to find a *minimal* subset. Using the dependency function to discriminate between candidates may lead the search down a non-minimal path. It is impossible to predict which combinations of attributes will lead to an optimal reduct based on changes in dependency with the addition or deletion of single attributes. It does result in a close-to-minimal subset, though, which is still useful in greatly reducing dataset dimensionality.

VPRS

Variable precision rough sets (VPRS) (Ziarko, 1993) extends rough set theory by the relaxation of the subset operator. It was proposed to analyse and identify data patterns which represent statistical trends rather than functional. The main idea of VPRS is to allow objects to be classified with an error smaller than a certain predefined level. This introduced threshold relaxes the rough set notion of requiring no information outside the dataset itself. Let $X, Y \subseteq U$, the relative classification error is defined by:

Equation 6

$$c(X,Y) = 1 - \frac{|X \bigcap Y|}{|X|}$$

Observe that $c(X,Y) = 0$ if and only if $X \subseteq Y$. A degree of inclusion can be achieved by allowing a certain level of error, β, in classification:

$$X \subseteq_\beta Y \text{ iff } c(X,Y) \leq \beta, \quad 0 \leq \beta < 0.5$$

Using $\subseteq_\beta$ instead of $\subseteq$, the β-upper and β-lower approximations of a set X can be defined as:

Equation 7

$$\underline{P}_\beta X = \cup\{[x]_P \in U/P \mid [x]_P \subseteq_\beta X\}$$

Equation 8

$$\overline{P}_\beta X = \cup\{[x]_P \in U/P \mid c([x]_P, X) < 1 - \beta\}$$

Note that $\underline{P}_\beta X = \underline{P}X$ for $\beta = 0$. The positive, negative and boundary regions in the original rough set theory can now be extended to:

$$POS_{P,\beta}(Q) = \bigcup_{X \in U/Q} \underline{P}_\beta X$$

$$NEG_{P,\beta}(Q) = U - \bigcup_{X \in U/Q} \overline{P}_\beta X$$

$$BND_{P,\beta}(Q) = \bigcup_{X \in U/Q} \overline{P}_\beta X - \bigcup_{X \in U/Q} \underline{P}_\beta X$$

where P is also an equivalence relation on U. This can then be used to calculate dependencies and thus determine β-reducts. The dependency function becomes:

Equation 9

$$\gamma_{P,\beta}(Q) = \frac{|POS_{P,\beta}(Q)|}{|U|}$$

Returning to the example dataset in Table 1, the β-positive region can be calculated for $P = \{b,c\}$, $Q = \{e\}$ and $\beta = 0.4$. Setting β to this value means that a set is considered to be a subset of another if at least 60% of their elements are shared. The partitions of the universe of objects for P and Q are:

$U/P = \{\{2\},\{0,4\},\{3\},\{1,6,7\},\{5\}\}$
$U/Q = \{\{0\},\{1,3,6\},\{2,4,5,7\}\}$

For each set $A \in U/P$ and $B \in U/Q$ the value of $c(A,B)$ must be less than β if the equivalence class A is to be included in the β-positive region. Considering $A = \{2\}$ gives

$c(\{2\},\{0\}) = 1 > \beta$
$c(\{2\},\{1,3,6\}) = 1 > \beta$
$c(\{2\},\{2,4,5,7\}) = 0 < \beta$

So object 2 is added to the β-positive region as it is a β-subset of {2,4,5,7} (and is in fact a traditional subset of the equivalence class). Taking A = {1,6,7}, a more interesting case is encountered:

$c(\{1,6,7\},\{0\}) = 1 > \beta$
$c(\{1,6,7\},\{1,3,6\}) = 0.3333 < \beta$
$c(\{1,6,7\},\{2,4,5,7\}) = 0.6667 > \beta$

Here the objects 1, 6 and 7 are included in the β-positive region as the set {1,6,7} is a β-subset of {1,3,6}. Calculating the subsets in this way leads to the following β-positive region:

$POS_{P,\beta}(X) = \{1,2,3,5,6,7\}$

Compare this with the positive region generated previously: {2,3,5}. Objects 1, 6 and 7 are now included due to the relaxation of the subset operator. If the original dataset contained noise, it could have been the case that these objects did indeed belong to the positive region. Using traditional rough set theory, this would not have been possible due to the inflexibility of the subset operator.

It can be seen that the QUICKREDUCT algorithm outlined previously can be adapted to incorporate the reduction method built upon the VPRS theory. By supplying a suitable β value to the algorithm, the β-lower approximation, β-positive region, and β-dependency can replace the traditional calculations. This will result in a more approximate final reduct, which may be a better generalization when encountering unseen data. Additionally, setting β to 0 forces such a method to behave exactly like RSAR.

Extended classification of reducts in the VPRS approach may be found in (Beynon, 2000; Beynon, 2001; Kryszkiewicz, 1994). As yet, there have been no comparative experimental studies between rough set methods and the VPRS method. However, the variable precision approach requires the additional parameter β which has to be specified from the start. By repeated experimentation, this parameter can be suitably approximated. However, problems arise when searching for true reducts as VPRS incorporates an element of inaccuracy in determining the number of classifiable objects.

Dynamic Reducts

Reducts generated from an information system are sensitive to changes in the system. This can be seen by removing a randomly chosen set of objects from the original object set. Those reducts frequently occurring in random subtables can be considered to be stable; it is these reducts that are encompassed by *dynamic reducts* (Bazan et al, 1994).

Let $A = (U, C \cup d)$ be a decision table, then any system $B = (U', C \cup d)$ $(U' \subseteq U)$ is called a subtable of A. If F is a family of subtables of A, then

Equation 10

$$DR(A,F) = \text{Red}(A,d) \cap \{\bigcap_{B \in F} \text{Red}(B,d)\}$$

defines the set of F-dynamic reducts of A. From this definition, it follows that a relative reduct of A is dynamic if it is also a reduct of all subtables in F. In most cases this is too restrictive, so a more general notion of dynamic reducts is required.

By introducing a threshold, $0 \le \varepsilon \le 1$, the concept of (F, ε)-dynamic reducts can be defined:

Equation 11

$$DR_\varepsilon(A,F) = \{C \in \text{Red}(A,d) : s_F(C) \ge \varepsilon\}$$

where

Equation 12

$$s_F(C) = \frac{|\{B \in F : C \in \text{Red}(B,d)\}|}{|F|}$$

is the F-stability coefficient of C. This lessens the previous restriction that a dynamic reduct must appear in *every* generated subtable. Now, a reduct is considered to be dynamic if it appears in a certain proportion of subtables, determined by the value ε. For example, by setting ε to 0.5 a reduct is considered to be dynamic if it appears in at least half of the subtables. Note that if $F = \{A\}$ then $DR(A,F) = \text{Red}(A,d)$. Dynamic reducts may then be calculated according to the algorithm given in Figure 5.

DynamicRed(*A, ε, its*)
A, the original decision table;
ε, the dynamic reduct threshold;
its, the number of iterations.

(1) $R \leftarrow \{\}$
(2) $T \leftarrow$ *calculateAllReducts*(*A*)
(3) **for** $j = 1 \ldots its$
(4) $A_j \leftarrow$ *deleteRandomRows(A)*
(5) $R \leftarrow R \cup$ *calculateAllReducts*(A_j)
(6) $\forall C \in T$
(7) **if** $s_F(C,R) \ge \varepsilon$
(8) **output** C

Figure 5 Dynamic reduct algorithm

Firstly, all reducts are calculated for the given information system, A. Then, the new subsystems A_i are generated by randomly deleting one or more rows from A. All reducts are found for each subsystem, and the dynamic reducts are computed using $s_F(C,R)$ which denotes the significance factor of reduct C within all reducts found, R.

Returning to the example decision table (call this A), the first step is to calculate all its reducts. This produces the set of all reducts $A = \{\{b,d\},\{c,d\},\{a,b,d\},\{a,c,d\},\{b,c,d\}\}$. The reduct $\{a,b,c,d\}$ is not included as this will always be a reduct of any generated subtable (it is the full set of conditional attributes). The next step randomly deletes a number of rows from the original table A. From this, all reducts are again calculated; for one subtable this might be $R = \{\{b,d\},\{b,c,d\},\{a,b,d\}\}$ In this case, the subset $\{c,d\}$ is not a reduct (though it was for the original dataset). If the number of iterations is set to just one, and if ε is set to a value less than 0.5 (implying that a reduct should appear in half of the total number of discovered reducts), then the reduct $\{c,d\}$ is deemed not to be a dynamic reduct.

Intuitively, this is based on the hope that by finding stable reducts they will be more representative of the real world, i.e. it is more likely that they will be reducts for unseen data. A comparison of dynamic and non-dynamic approaches can be found in (Bazan, 1998), where various methods were tested on extracting laws from decision tables. In the experiments, the dynamic method and the conventional RS method both performed well. In fact, it appears that the RS method has on average a lower error rate of classification than the dynamic RS method.

A disadvantage of this dynamic approach is that several subjective choices have to be made before the dynamic reducts can be found (for instance the choice of the value of ε; these values are not contained in the data. Also, the huge complexity of finding all reducts within subtables forces the use of heuristic techniques such as genetic algorithms to perform the search. For large datasets, this step may well be too costly.

Han et al.

In (Han et al., 2004), a feature selection method based on an alternative dependency measure is presented. The technique was originally proposed to avoid the calculation of discernibility functions or positive regions, which can be computationally expensive without optimizations

The authors replace the traditional rough set degree of dependency with an alternative measure, the relative dependency, defined as follows for an attribute subset R:

Equation 13

$$\kappa_R(D) = \frac{|U / IND(R)|}{|U / IND(R \cup D)|}$$

The authors then show that R is a reduct if and only if $\kappa_R(D) = \kappa_C(D)$ and $\forall X \subset R,\ \kappa_X(D) \neq \kappa_C(D)$.

Two algorithms are constructed for feature selection based on this measure. The first (Figure 6) performs backward elimination of features where attributes are removed from the set of considered attributes if the relative dependency equals 1 upon their removal. Attributes are considered one at a time, starting with the first, evaluating their relative dependency. The second algorithm initially ranks the individual

attributes beforehand using an entropy measure before the backward elimination is performed.

RelativeReduct(C,D)
C, the conditional attributes;
D, the decision attributes;

(1) $R \leftarrow C$
(2) $\forall a \in C$
(3) **if** ($\kappa_{R-\{a\}}(D) == 1$)
(4) $R \leftarrow R - \{a\}$
(6) **return** R

Figure 6 Backward elimination based on relative dependency

Returning to the example dataset, the backward elimination algorithm initializes R to the set of conditional attributes, $\{a,b,c,d\}$. Next, the elimination of attribute a is considered:

$$\kappa_{\{b,c,d\}}(D) = \frac{|U/IND(\{b,c,d\})|}{|U/IND(\{b,c,d,e\})|} = \frac{|\{\{0\},\{1,6\},\{2\},\{3\},\{4\},\{5\},\{7\}\}|}{|\{\{0\},\{1,6\},\{2\},\{3\},\{4\},\{5\},\{7\}\}|} = 1$$

As the relative dependency is equal to 1, attribute a can be removed from the current reduct candidate $R \leftarrow \{b,c,d\}$. The algorithm then considers the elimination of attribute b from R:

$$\kappa_{\{c,d\}}(D) = \frac{|U/IND(\{c,d\})|}{|U/IND(\{c,d,e\})|} = \frac{|\{\{0\},\{1,6\},\{2,5\},\{3\},\{4\},\{7\}\}|}{|\{\{0\},\{1,6\},\{2,5\},\{3\},\{4\},\{7\}\}|} = 1$$

Again, the relative dependency of $\{c,d\}$ evaluates to 1, so attribute b is removed from R, ($R = \{c,d\}$). The next step evaluates the removal of c from the reduct candidate:

$$\kappa_{\{d\}}(D) = \frac{|U/IND(\{d\})|}{|U/IND(\{d,e\})|} = \frac{|\{\{0,3\},\{1,2,5,6\},\{4,7\}\}|}{|\{\{0\},\{3\},\{1,6\},\{2,5\},\{4,7\}\}|} = \frac{3}{5}$$

As this does not equal 1, attribute d is not removed from R. The algorithm then evaluates the elimination of attribute d from R ($R = \{c,d\}$):

$$\kappa_{\{c\}}(D) = \frac{|U/IND(\{c\})|}{|U/IND(\{c,e\})|} = \frac{|\{\{0,4\},\{1,6,7\},\{2,3,5\}\}|}{|\{\{0\},\{4\},\{1,6\},\{7\},\{2,5\},\{3\}\}|} = \frac{3}{6}$$

Again, the relative dependency does not evaluate to 1, hence attribute d is retained in the reduct candidate. As there are no further attributes to consider, the algorithm terminates and outputs the reduct $\{c,d\}$.

select(C,D,O,ε)
C, the set of all conditional features;
D, the set of decision features.
O, the set of objects (instances)
ε, the reduct threshold

(1) $R \leftarrow$ calculateCore()
(2) **while** $\gamma_R(D) < \varepsilon$
(3) $O \leftarrow O - POS_R(D)$ *//optimization*
(4) $\forall x \in (C - R)$
(5) $v_a = | POS_{R \cup \{a\}}(D) |$
(6) $m_a \leftarrow | largestEquivClass(POS_{R \cup \{a\}}(D)) |$
(7) *Choose a with largest* $v_a \times m_a$
(8) $R \leftarrow R \cup \{a\}$
(9) **return** R

Figure 7 Heuristic filter-based algorithm

In (Zhong et al, 2001), a heuristic filter-based approach is presented based on rough set theory. The algorithm proposed, as reformalised in Figure 7, begins with the core of the dataset (those attributes that cannot be removed without introducing inconsistencies) 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. However, there are other methods that can calculate the core in an efficient manner (Pawlak 1991). For example, this can be done by calculating the degree of dependency of the full feature set and the corresponding dependencies of the feature set minus each attribute. Those features that result in a dependency decrease are core attributes. There are also alternative methods available that allow the calculation of necessary information about the discernibility matrix without the need to perform operations directly on it (Nguyen & Nguyen, 1996).

EBR

A further technique for rough set feature selection is entropy-based reduction (EBR), developed from work carried out in (Jensen & Shen, 2004a). This approach is based on the entropy heuristic employed by machine learning techniques such as C4.5 (Quinlan, 1993). A similar approach has been adopted in (Dash & Liu, 1997) where an entropy measure is used for ranking features. EBR is concerned with examining a dataset and determining those attributes that provide the most gain in information. The entropy of attribute A (which can take values $a_1,...,a_m$) with respect to the conclusion C (of possible values $c_1,...,c_n$) is defined as:

Equation 14

$$H(C \mid A) = -\sum_{j=1}^{m} p(a_j) \sum_{i=1}^{n} p(c_i \mid a_j) \log_2 p(c_i \mid a_j)$$

EBR(*C*,*D*)
C, the set of all conditional features;
D, the set of decision features.

(1) $R \leftarrow \{\}$
(2) **do**
(3) $T \leftarrow R$
(4) $\forall x \in (C - R)$
(5) **if** $H(R \cup \{x\}) < H(T)$
(6) $T \leftarrow R \cup \{x\}$
(7) $R \leftarrow T$
(8) **until** $H(D \mid R) = H(D \mid C)$
(9) **return** R

Figure 8 Entropy-based reduction

This can be extended to dealing with *subsets* of attributes instead of individual attributes only. Using this entropy measure, the algorithm used in rough set-based attribute reduction (Chouchoulas & Shen, 2001) can be modified to that shown in Figure 8. This algorithm requires no thresholds in order to function - the search for the best feature subset is stopped when the resulting subset entropy is equal to that of the entire feature set. For consistent data, the final entropy of the subset will be zero. It is interesting to note that any subset with an entropy of 0 will also have a corresponding rough set dependency of 1. Hence, this technique can be used for finding rough set reducts if the data is consistent.

Returning to the example dataset, EBR first evaluates the entropy of each individual attribute:

Subset	Entropy
$\{a\}$	1.1887219
$\{b\}$	0.75
$\{c\}$	0.9387219
$\{d\}$	0.75

The subsets with lowest entropy here are $\{b\}$ and $\{d\}$. The algorithm selects attribute *b* due to it being evaluated first and adds it to the current feature subset. The next step is to calculate the entropy of all subsets containing *b* and one other attribute:

Subset	Entropy
$\{a,b\}$	0.5
$\{b,c\}$	0.59436095
$\{b,d\}$	0.0

Here, the subset $\{b,d\}$ is chosen as this results in the lowest entropy. Additionally, the stopping criterion has been met as this value equals the entropy for the entire feature set ($H(D|\{b,d\}) = 0 = H(D|C)$). The algorithm terminates and returns this feature subset – the dataset can now be reduced to these features only. As the resulting entropy is zero, the returned subset is a rough set reduct.

Other Algorithms

Among the first rough set-based approaches is the *Preset* algorithm (Modrzejewski, 1993) which is another feature selector that uses rough set theory to rank heuristically the features, assuming a noise free binary domain. Since Preset does not try to explore all combinations of the features, it is certain that it will fail on problems whose attributes are highly correlated. There have also been investigations into the use of different reduct quality measures (see (Polkowski et al, 2000) for details).

In (Zhang &Yao, 2004), a new rough set based feature selection heuristic, Parameterized Average Support Heuristic (PASH), is proposed. Unlike the existing methods, PASH is based on a special parameterized lower approximation which is defined to include all predictive instances. Predictive instances are instances that may produce predictive rules which hold true with a high probability but are not necessarily always true. The traditional model could exclude predictive instances that may produce such rules. The main advantage of PASH is that it considers the overall quality of the potential rules, thus producing a set of rules with balanced support distribution over all decision classes. However, it requires a parameter to be defined by the user that adjusts the level of approximation. One of the main benefits of rough set theory is that it does not require such additional information, and hence eliminates the need for user interaction or repeated experimentation.

DISCERNIBILITY MATRIX-BASED APPROACHES

Many applications of rough sets to feature selection make use of discernibility matrices for finding reducts. A discernibility matrix (Skowron & Rauszer, 1992) of a decision table $D = (U, C \cup d)$ is a symmetric $|U| \times |U|$ matrix with entries defined:

Equation 15

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

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 found in Table 2 is produced. 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). Here, the core of the dataset is $\{d\}$.

$x \in U$	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*	

Table 2 The decision-relative discernibility matrix

From this, the discernibility function can be defined. 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 attributes $a_1,..., a_m$) defined as below:

Equation 16

$$f_D(a_1^*,...,a_m^*) = \wedge\{\vee c_{ij}^* \mid 1 \le j \le i \le |U|, c_{ij} \neq \varnothing\}$$

where $c_{ij}^* = \{a^* \mid a \in c_{ij}\}$. By finding the set of all prime implicants of the discernibility function, all the minimal reducts of a system may be determined. From Table 2, the decision-relative discernibility function is (with duplicates removed):

$$f_D(a,b,c,d) = \{a \vee b \vee c \vee d\} \wedge \{a \vee c \vee d\} \wedge \{b \vee c\} \wedge \{d\}$$
$$\wedge \{a \vee b \vee c\} \wedge \{a \vee b \vee d\} \wedge \{b \vee c \vee d\} \wedge \{a \vee d\}$$

Further simplification can be performed by removing those sets (clauses) that are supersets of 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). 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.

Johnson Reducer

This is a simple greedy heuristic algorithm that is often applied to discernibility functions to find a single reduct (Øhrn, 1999). Reducts found by this process have no guarantee of minimality, but are generally of a size close to the minimal.

Johnson(C,f_D)
C, the set of conditional attributes
f_D, the discernibility function.

(1) $R \leftarrow \varnothing$; bestc=0;
(2) **while** (f_D not empty)
(3) **for each** $a \in C$ that appears in f_D
(4) c = heuristic(a)
(5) **if** (c > bestc)
(6) bestc=c; bestAttr $\leftarrow a$
(7) $R \leftarrow R \cup a$
(8) $f_D \leftarrow$ removeClauses(f_D, a)
(9) **return** R

Figure 9 Johnson algorithm

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 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 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 (Nguyen & Skowron, 1997b; Wang & Wang, 2001). However, no perfect heuristic exists, and hence there is still no guarantee of subset optimality.

Compressibility algorithm

In (Starzyk et al, 2000), the authors present a method for the generation of all reducts in an information system by manipulating the clauses in discernibility functions. In addition to the standard simplification laws (such as the removal of supersets), the concept of strong compressibility is introduced and applied in conjunction with an expansion algorithm.

The strong compressibility simplification applies where clause attributes are either simultaneously present or absent in all clauses. In this situation, the attributes may be replaced by a single representative attribute. As an example, consider the formula:

$$f_D = \{a \vee b \vee c \vee f\} \wedge \{b \vee d\} \wedge \{a \vee d \vee e \vee f\} \wedge \{d \vee c\}$$

The attributes *a* and *f* can be replaced by a single attribute as they are both present in the first and third clauses, and absent in the second and fourth. Replacing $\{a \vee f\}$ with *g* results in:

$$f_D = \{g \vee b \vee c\} \wedge \{b \vee d\} \wedge \{g \vee d \vee e\} \wedge \{d \vee c\}$$

If a reduct resulting from this discernibility function contains the new attribute *g*, then this attribute may be replaced by either *a* or *f*. Here, {*g,d*}is a reduct and so {*a,d*}and {*f,d*}are reducts of the original set of clauses. Hence, fewer attributes are considered in the reduct-determining process with no loss of information. The complexity of this step is $O(a*c + a^2)$, where *a* is the number of attributes and *c* is the number of clauses. The overall algorithm for determining reducts can be found in Figure 10. It uses concepts from Boolean algebra (such as the absorption and expansion laws) with strong compressibility for simplifying the discernibility function.

Compressibility(f_D)
f_D, the discernibility function.

(1) **while** (f_D not in simple form)
(2) applyAbsorptionLaws(f_D) //remove supersets
(3) replaceStronglyCompressibleAttributes(f_D)
(4) $a \leftarrow$ mostFrequentAttribute(f_D)
(5) applyExpansionLaw(a,f_D)
(6) substituteStronglyCompressibleClasses(f_D)
(7) Reds $\leftarrow$ calculateReducts(f_D)
(8) **return** minimalElements(Reds)

Figure 10 Compressibility algorithm

Returning to the example above and following the algorithm from step 4, the most commonly occurring attribute can be seen to be *d*. Hence, the expansion law is applied with respect to this attribute to obtain:

$$\begin{aligned} f_D &= f_1 \vee f_2 \\ &= (\{d\} \wedge \{g \vee b \vee c\}) \vee (\{g \vee b \vee c\} \wedge \{b\} \wedge \{g \vee e\} \wedge \{c\}) \\ &= (\{d\} \wedge \{g \vee b \vee c\}) \vee (\{b\} \wedge \{g \vee e\} \wedge \{c\}) \end{aligned}$$

As all components are in simple form, step 6 is carried out, where all strongly equivalent classes are replaced by their equivalent attributes:

$$\begin{aligned} f_D &= f_1 \vee f_2 \\ &= (\{d\} \wedge \{a \vee f \vee b \vee c\}) \vee (\{b\} \wedge \{a \vee f \vee e\} \wedge \{c\}) \end{aligned}$$

The corresponding reducts for the function components are as follows (Step 7):

$$\text{Red}(f_1) = (\{a,d\},\{d,f\},\{b,d\},\{c,d\})$$
$$\text{Red}(f_2) = (\{b,a,c\},\{b,f,c\},\{b,e,c\})$$

The reducts for the system are generated by taking the union of the components and determining the minimal elements:

$$\text{Red}(f_D) = (\{a,d\},\{d,f\},\{b,d\},\{c,d\},\{b,a,c\},\{b,f,c\},\{b,e,c\})$$

RSAR-SAT

The problem of finding the smallest feature subsets using rough set theory can be formulated as a SAT problem. Rough sets 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 to 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_is set to 1) for consistent datasets. Preliminary work has been carried out in this area (Bakar et al, 2002), though this does not adopt a DPLL-style approach to finding solutions.

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) $x \leftarrow$ selectLiteral(*F*)
(10) **if** (DPLL($F \cup \{x\}$) is satisfiable)
(11) **return** satisfiable
(12) **else return** DPLL($F \cup \{-x\}$)

Figure 11 The definition of the DPLL algorithm

The DPLL algorithm for finding minimal subsets can be found in Figure 11, 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 for this purpose (Zhang & Malik, 2002), but are not considered 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 variables currently assigned 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 (Hoos & Stützle, 1999) these are not applicable here as they provide no guarantee of solution minimality. The DPLL-based algorithm will always find the minimal optimal subset. However, this will come at the expense of time taken to find it. The initial experimentation (Jensen et al, 2005) has shown that the method performs well in comparison to RSAR, which often fails to find the smallest subsets.

HANDLING CONTINUOUS VALUES

The reliance on discrete data for the successful operation of RST can be seen as a significant drawback of the approach. Indeed, this requirement of RST implies an objectivity in the data that is simply not present (Koczkodaj et al., 1998). For example, in a medical dataset, values such as *Yes* or *No* cannot be considered objective for a *Headache* attribute as it may not be straightforward to decide whether a person has a headache or not to a high degree of accuracy. Again, consider an attribute *Blood Pressure*. In the real world, this is a real-valued measurement but for the purposes of RST must be discretised into a small set of labels such as *Normal*, *High*, etc. Subjective judgments are required for establishing boundaries for objective measurements.

In the rough set literature, there are two main ways of handling real-valued attributes – through fuzzy-rough sets and tolerance rough sets. Both approaches replace the traditional equivalence classes of crisp rough set theory with alternatives that are better suited to dealing with this type of data. In the fuzzy-rough case, fuzzy equivalence classes are employed within a fuzzy extension of rough set theory, resulting in a hybrid approach. In the tolerance case, indiscernibility relations are replaced with similarity relations that permit a limited degree of variability in attribute

values. Approximations are constructed based on these tolerance classes in a manner similar to that of traditional rough set theory.

$x \in U$	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

Table 3 Example dataset: crisp decisions

To illustrate the operation of the techniques involved, an example dataset is given in Table 3. The table contains three real-valued conditional attributes and a crisp-valued decision attribute.

Fuzzy Rough Sets

A way of handling this problem is through the use of *fuzzy-rough* sets. Subjective judgments are not entirely removed as fuzzy set membership functions still need to be defined. However, the method offers a high degree of flexibility when dealing with real-valued data, enabling the vagueness and imprecision present to be modelled effectively.

Fuzzy Equivalence Classes

In the same way that crisp equivalence classes are central to rough sets, *fuzzy* equivalence classes are central to the fuzzy-rough set approach (Dubois & Prade, 1992; Thiele, 1998; Yao, 1998). For typical applications, this means that the decision values and the conditional values may all be fuzzy. The concept of crisp equivalence classes can be extended by the inclusion of a fuzzy similarity relation *S* on the universe, which determines the extent to which two elements are similar in *S*. For example, if $\mu_S(x, y) = 0.9$, then objects *x* and *y* are considered to be quite similar. The usual properties of reflexivity ($\mu_S(x, x) = 1$), symmetry ($\mu_S(x, y) = \mu_S(y, x)$) and transitivity ($\mu_S(x, z) \geq \mu_S(x, y) \wedge \mu_S(y, z)$) hold.

Using the fuzzy similarity relation, the fuzzy equivalence class $[x]_S$ for objects close to *x* can be defined:

Equation 17

$$\mu_{[x]_S}(y) = \mu_S(x, y)$$

The following axioms should hold for a fuzzy equivalence class *F* (Höhle, 1988):

$\exists\, \mu_F(x) = 1$ (μ_F is normalised)

$\mu_F(x) \wedge \mu_S(x, y) \leq \mu_F(y)$

$\mu_F(x) \wedge \mu_F(y) \leq \mu_S(x, y)$

The first axiom corresponds to the requirement that an equivalence class is non-empty. The second axiom states that elements in *y*'s neighbourhood are in the equivalence class of *y*. The final axiom states that any two elements in *F* are related via *S*. Obviously, this definition degenerates to the normal definition of equivalence classes when *S* is non-fuzzy.

The family of normal fuzzy sets produced by a fuzzy partitioning of the universe of discourse can play the role of fuzzy equivalence classes (Dubois & Prade, 1992). Consider the crisp partitioning of a universe of discourse, *U*, by the attributes in *Q*: $U/Q = \{\{1,3,6\},\{2,4,5\}\}$. This contains two equivalence classes ($\{1,3,6\}$ and $\{2,4,5\}$) that can be thought of as degenerated fuzzy sets, with those elements belonging to the class possessing a membership of one, zero otherwise. For the first class, for instance, the objects 2, 4 and 5 have a membership of zero. Extending this to the case of fuzzy equivalence classes is straightforward: objects can be allowed to assume membership values, with respect to any given class, in the interval [0,1]. *U/Q* is not restricted to crisp partitions only; fuzzy partitions are equally acceptable.

Fuzzy-Rough Feature Selection

Fuzzy-Rough Feature Selection (FRFS) (Jensen & Shen, 2004a; Jensen & Shen, 2004b; Shen & Jensen, 2004) provides a means by which discrete or real-valued noisy data (or a mixture of both) can be effectively reduced without the need for user-supplied information. Additionally, this technique can be applied to data with continuous or nominal decision attributes, and as such can be applied to regression as well as classification datasets. The only additional information required is in the form of fuzzy partitions for each feature which can be automatically derived from the data.

From the literature, the fuzzy *P*-lower and *P*-upper approximations are defined as (Dubois & Prade, 1992):

Equation 18

$$\mu_{\underline{P}X}(F_i) = \inf_x \max\{1 - \mu_{F_i}(x), \mu_X(x)\} \quad \forall i$$

Equation 19

$$\mu_{\overline{P}X}(F_i) = \sup_x \min\{\mu_{F_i}(x), \mu_X(x)\} \quad \forall i$$

where F_i denotes a fuzzy equivalence class belonging to *U/P*. Note that although the universe of discourse in feature selection is finite, this is not the case in general, hence the use of *sup* and *inf*. These definitions diverge a little from the crisp upper and lower approximations, as the memberships of individual objects to the approximations are not explicitly available. As a result of this, the fuzzy lower and upper approximations are herein redefined as:

Equation 20

$$\mu_{\underline{P}X}(x) = \sup_{F \in U/P} \min(\mu_F(x), \inf_{y \in U} \max\{1 - \mu_F(y), \mu_X(y)\})$$

Equation 21

$$\mu_{\underline{P}X}(x) = \sup_{F \in U/P} \min(\mu_F(x), \sup_{y \in U} \min\{\mu_F(y), \mu_X(y)\})$$

In implementation, not all $y \in U$ need to be considered - only those where $\mu_F(y)$ is non-zero, i.e. where object y is a fuzzy member of (fuzzy) equivalence class F. The tuple $< \underline{P}X, \overline{P}X >$ is called a *fuzzy-rough* set.

The crisp positive region in traditional rough set theory is defined as the union of the lower approximations. By the extension principle (Zadeh, 1975), the membership of an object $x \in U$, belonging to the fuzzy positive region can be defined by

Equation 22

$$\mu_{POS_P(Q)}(x) = \sup_{X \in U/Q} \mu_{\underline{P}X}(x)$$

Object x will not belong to the positive region only if the equivalence class it belongs to is not a constituent of the positive region. This is equivalent to the crisp version where objects belong to the positive region only if their underlying equivalence class does so. Similarly, the negative and boundary regions can be defined. For this particular feature selection method, the upper approximation is not used, though this may be useful for other methods.

Using the definition of the fuzzy positive region, the new dependency function can be defined as follows:

Equation 23

$$\gamma'_P(Q) = \frac{|\mu_{POS_P(Q)}(x)|}{|U|} = \frac{\sum_{x \in U} \mu_{POS_P(Q)}(x)}{|U|}$$

As with crisp rough sets, the dependency of Q on P is the proportion of objects that are discernible out of the entire dataset. In the present approach, this corresponds to determining the fuzzy cardinality of $\mu_{POS_P(Q)}(x)$ divided by the total number of objects in the universe. The definition of dependency degree covers the crisp case as its specific instance.

If the fuzzy-rough reduction process is to be useful, it must be able to deal with multiple features, finding the dependency between various subsets of the original feature set. For example, it may be necessary to be able to determine the degree of dependency of the decision feature(s) with respect to $P = \{a,b\}$. In the crisp case, *U/P* contains sets of objects grouped together that are indiscernible according to both features *a* and *b*. In the fuzzy case, objects may belong to many equivalence classes, so the cartesian product of *U/IND*($\{a\}$) and *U/IND*($\{b\}$) must be considered in determining *U/P*.

Each set in *U/P* denotes an equivalence class. For example, if $P = \{a,b\}$, *U/IND*($\{a\}$) $= \{N_a, Z_a\}$ and *U/IND*($\{b\}$) $= \{N_b, Z_b\}$, then

$$U / P = \{N_a \cap N_b, N_a \cap Z_b, Z_a \cap N_b, Z_a \cap Z_b\}$$

The extent to which an object belongs to such an equivalence class is therefore calculated by using the conjunction of constituent fuzzy equivalence classes, say F_i, i=1,2,...,n:

Equation 24

$$\mu_{F_1 \cap F_2 \cap .. \cap F_n}(x) = \min(\mu_{F_1}(x), \mu_{F_2}(x), ..., \mu_{F_n}(x))$$

Fuzzy-Rough QUICKREDUCT

A problem may arise when this approach is compared to the crisp approach. In conventional RSAR, a reduct is defined as a subset R of the features which have the same information content as the full feature set A. In terms of the dependency function this means that the values $\gamma_R(Q)$ and $\gamma_A(Q)$ are identical and equal to 1 if the dataset is consistent. However, in the fuzzy-rough approach this is not necessarily the case as the uncertainty encountered when objects belong to many fuzzy equivalence classes results in a reduced total dependency.

FRQUICKREDUCT(C,D)
C, the set of all conditional features;
D, the set of decision features.

(1) $R \leftarrow \{\}; \gamma'_{best} = 0$
(2) **do**
(3) $T \leftarrow R$
(4) $\gamma'_{prev} = \gamma'_{best}$
(5) $\forall x \in (C - R)$
(6) **if** $\gamma'_{R \cup \{x\}}(D) > \gamma'_T(D)$
(7) $T \leftarrow R \cup \{x\}$
(8) $\gamma'_{best} = \gamma'_T(D)$
(9) $R \leftarrow T$
(10) **until** $\gamma'_{best} == \gamma'_{prev}$
(11) **return** R

Figure 12 The fuzzy-rough QUICKREDUCT algorithm

A possible way of combatting this would be to determine the degree of dependency of a set of decision features D upon the full feature set and use this as the denominator rather than $|U|$ (for normalization), allowing γ' to reach 1. With these issues in mind, a new QUICKREDUCT algorithm has been developed as given in Figure 12. It employs the new dependency function γ' to choose which features to add to the current reduct candidate in the same way as the original QUICKREDUCT process (see Figure 4). The algorithm terminates when the addition of any remaining feature does not increase the dependency (such a criterion could be used with the original QUICKREDUCT algorithm).

As the new degree of dependency measure is non-monotonic, it is possible that the QUICKREDUCT style search terminates having reached only a local optimum. The global optimum may lie elsewhere in the search space. This motivates the adoption of alternative search mechanisms. However, the algorithm as presented in Figure 12 is still highly useful in locating good subsets quickly.

Note that an intuitive understanding of the algorithm implies that, for a dimensionality of n, $(n^2 + n)/2$ evaluations of the dependency function may be performed for the worst-case dataset. However, as FRFS is used for dimensionality reduction prior to any involvement of the system which will employ those features belonging to the resultant reduct, this operation has no negative impact upon the run-time efficiency of the system.

It is also possible to reverse the search; that is, start with the full set of features and incrementally remove the least informative features. This process continues until no more features can be removed without reducing the total number of discernible objects in the dataset. Again, this tends not to be applied to larger datasets as the cost of evaluating these larger feature subsets is too great.

Application to the Example Dataset

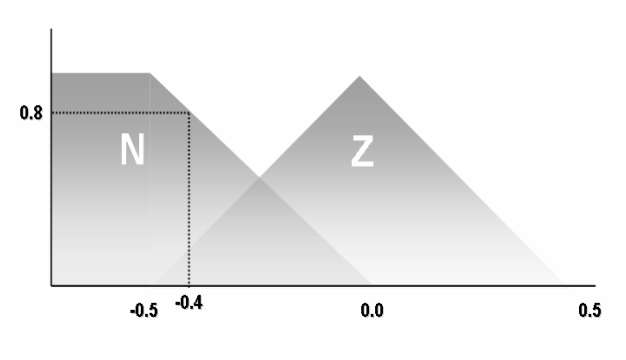

Figure 13 Fuzzifications for conditional features

Using Table 3 and the fuzzy sets defined in Figure 13 (for all conditional attributes), and setting $A=\{a\}$, $B=\{b\}$, $C=\{c\}$ and $Q=\{q\}$, the following equivalence classes are obtained:

$U/A = \{N_a, Z_a\}$
$U/B = \{N_b, Z_b\}$
$U/C = \{N_c, Z_c\}$
$U/Q = \{\{1,3,6\}, \{2,4,5\}\}$

The first step is to calculate the lower approximations of the sets A, B and C, using Equation 20. To clarify the calculations involved, Table 4 contains the membership degrees of objects to fuzzy equivalence classes.

$x \in U$	a		b		c		q	
	N_a	Z_a	N_b	Z_b	N_c	Z_c	{1,3,6}	{2,4,5}
1	0.8	0.2	0.6	0.4	1.0	0.0	1.0	0.0
2	0.8	0.2	0.0	0.6	0.2	0.8	0.0	1.0
3	0.6	0.4	0.8	0.2	0.6	0.4	1.0	0.0
4	0.0	0.4	0.6	0.4	0.0	1.0	0.0	1.0
5	0.0	0.6	0.6	0.4	0.0	1.0	0.0	1.0
6	0.0	0.6	0.0	1.0	0.0	1.0	1.0	0.0

Table 4 Membership values of objects to corresponding fuzzy sets

For simplicity, only A will be considered here; that is, using A to approximate Q. For the first decision equivalence class $X = \{1,3,6\}$, $\mu_{\underline{A}\{1,3,6\}}(x)$ needs to be calculated:

$$\mu_{\underline{A}\{1,3,6\}}(x) = \sup_{F \in U/A} \min(\mu_F(x), \inf_{y \in U} \max\{1 - \mu_F(y), \mu_{\{1,3,6\}}(y)\})$$

Considering the first fuzzy equivalence class of A, N_a:

$$\min(\mu_{N_a}(x), \inf_{y \in U} \max\{1 - \mu_{N_a}(y), \mu_{\{1,3,6\}}(y)\})$$

For object 2 this can be calculated as follows. From Table 4 it can be seen that the membership of object 2 to the fuzzy equivalence class N_a, $\mu_{N_a}(2)$, is 0.8. The remainder of the calculation involves finding the smallest of the following values:

$$\max(1 - \mu_{N_a}(1), \mu_{\{1,3,6\}}(1)) = \max(0.2,1.0) = 1.0$$
$$\max(1 - \mu_{N_a}(2), \mu_{\{1,3,6\}}(2)) = \max(0.2,0.0) = 0.2$$
$$\max(1 - \mu_{N_a}(3), \mu_{\{1,3,6\}}(3)) = \max(0.4,1.0) = 1.0$$
$$\max(1 - \mu_{N_a}(4), \mu_{\{1,3,6\}}(4)) = \max(1.0,0.0) = 1.0$$
$$\max(1 - \mu_{N_a}(5), \mu_{\{1,3,6\}}(5)) = \max(1.0,0.0) = 1.0$$
$$\max(1 - \mu_{N_a}(6), \mu_{\{1,3,6\}}(6)) = \max(1.0,1.0) = 1.0$$

From the calculations above, the smallest value is 0.2, hence:

$$\min(\mu_{N_a}(x), \inf_{y \in U} \max\{1 - \mu_{N_a}(y), \mu_{\{1,3,6\}}(y)\}) = \min(0.8, \inf\{1,0.2,1,1,1,1\})$$
$$= 0.2$$

Similarly for Z_a

$$\min(\mu_{Z_a}(x), \inf_{y \in U} \max\{1 - \mu_{Z_a}(y), \mu_{\{1,3,6\}}(y)\}) = \min(0.2, \inf\{1,0.8,1,0.6,0.4,1\})$$
$$= 0.2$$

Thus,

$$\mu_{\underline{A}\{1,3,6\}}(2) = 0.2$$

Calculating the A-lower approximation of $X = \{1,3,6\}$ for every object gives

$$\mu_{\underline{A}\{1,3,6\}}(1) = 0.2, \quad \mu_{\underline{A}\{1,3,6\}}(2) = 0.2$$
$$\mu_{\underline{A}\{1,3,6\}}(3) = 0.4, \quad \mu_{\underline{A}\{1,3,6\}}(4) = 0.4$$
$$\mu_{\underline{A}\{1,3,6\}}(5) = 0.4, \quad \mu_{\underline{A}\{1,3,6\}}(6) = 0.4$$

The corresponding values for $X = \{2,4,5\}$ can also be determined:

$$\mu_{\underline{A}\{2,4,5\}}(1) = 0.2, \quad \mu_{\underline{A}\{2,4,5\}}(2) = 0.2$$
$$\mu_{\underline{A}\{2,4,5\}}(3) = 0.4, \quad \mu_{\underline{A}\{2,4,5\}}(4) = 0.4$$
$$\mu_{\underline{A}\{2,4,5\}}(5) = 0.4, \quad \mu_{\underline{A}\{2,4,5\}}(6) = 0.4$$

It is a coincidence here that $\mu_{\underline{A}\{1,3,6\}}(x) = \mu_{\underline{A}\{2,4,5\}}(x)$ for this example. Using these values, the fuzzy positive region for each object can be calculated via using

$$\mu_{POS_A(Q)}(x) = \sup_{X \in U/Q} \mu_{\underline{A}X}(x)$$

This results in:

$$\mu_{POS_A(Q)}(1) = 0.2, \quad \mu_{POS_A(Q)}(2) = 0.2$$
$$\mu_{POS_A(Q)}(3) = 0.4, \quad \mu_{POS_A(Q)}(4) = 0.4$$
$$\mu_{POS_A(Q)}(5) = 0.4, \quad \mu_{POS_A(Q)}(6) = 0.4$$

The next step is to determine the degree of dependency of Q on A:

$$\gamma'_A(Q) = \frac{\sum_{x \in U} \mu_{POS_A(Q)}(x)}{|U|} = \frac{2}{6}$$

Calculating for B and C gives:

$$\gamma'_B(Q) = \frac{2.4}{6}, \quad \gamma'_C(Q) = \frac{1.6}{6}$$

From this it can be seen that attribute b will cause the greatest increase in dependency degree. This attribute is chosen and added to the potential reduct. The process iterates and the two dependency degrees calculated are

$$\gamma'_{\{a,b\}}(Q) = \frac{3.4}{6}, \quad \gamma'_{\{b,c\}}(Q) = \frac{3.2}{6}$$

Adding attribute a to the reduct candidate causes the larger increase of dependency, so the new candidate becomes $\{a,b\}$. Lastly, attribute c is added to the potential reduct:

$$\gamma'_{\{a,b,c\}}(Q) = \frac{3.4}{6}$$

As this causes no increase in dependency, the algorithm stops and outputs the reduct $\{a,b\}$. The dataset can now be reduced to only those attributes appearing in the reduct. When crisp RSAR is performed on this dataset (after using the same fuzzy sets to discretize the real-valued attributes), the reduct generated is $\{a,b,c\}$, i.e. the full conditional attribute set (Jensen & Shen, 2004b). Unlike crisp RSAR, the true minimal reduct was found using the information on degrees of membership. It is clear from this example alone that the information lost by using crisp RSAR can be important when trying to discover the smallest reduct from a dataset.

Fuzzy Entropy-Guided FRFS

From previous experimentation with crisp rough sets and entropy (Jensen & Shen, 2004b) it was observed that entropy-based methods often found smaller reducts than those based on the dependency function. This provided the motivation for a new fuzzy-rough technique using fuzzy entropy to guide search (Kosko, 1986; Mac Parthláin et al., 2006), in order to locate optimal fuzzy-rough subsets.

Fuzzy Entropy

Again, let $I = (U,$ A) be a decision system, where U is a non-empty set of finite objects. $A = \{C \cup D\}$ is a non-empty finite set of attributes, where C is the set of input features and D is the set of classes. An attribute $a \in A$ has corresponding fuzzy subsets $F_1, F_2, \ldots, F_n$. The fuzzy entropy for a fuzzy subset F_i can be defined as being:

Equation 25

$$H(D \mid F_i) = \sum_{Q \in U/D} -p(Q \mid F_i) \log_2 p(Q \mid F_i)$$

where $p(Q \mid F_i)$ is the relative frequency of the fuzzy subset F_i of attribute a with respect to the decision Q, and is defined:

Equation 26

$$p(Q \mid F_i) = \frac{|Q \cap F_i|}{|F_i|}$$

The cardinality of a fuzzy set is denoted by $|\,.\,|$. Based on these definitions, the fuzzy entropy for an attribute subset R is defined as follows:

$$E(D \mid R) = \sum_{F_i \in U/R} \frac{|F_i|}{\sum_{Y_i \in U/R} |Y_i|} . H(D \mid F_i)$$

This fuzzy entropy can be used to gauge the utility of attribute subsets in a similar way to that of the fuzzy-rough measure. However, the fuzzy entropy measure

decreases with increasing subset utility, whereas the fuzzy-rough dependency measure increases. With these definitions, a new feature selection mechanism can be constructed that uses fuzzy entropy to guide the search for the best fuzzy-rough feature subset.

Fuzzy Entropy-based QUICKREDUCT

FREQUICKREDUCT(*C,D*)
C, the set of all conditional features;
D, the set of decision features.

(1) $T \leftarrow \{\}; \gamma'_{prev} = 0$
(2) **do**
(3) $R \leftarrow T$
(4) $\gamma'_{prev} = \gamma'_{T}(D)$
(5) $\forall x \in (C - R)$
(6) **if** $E(D \mid R \cup \{x\}) < E(D \mid T)$
(7) $T \leftarrow R \cup \{x\}$
(8) **until** $\gamma'_{T}(D) \leq \gamma'_{prev}$
(9) **return** R

Figure 14 The fuzzy-rough entropy-based QUICKREDUCT algorithm

Figure 14 shows a fuzzy-rough entropy-based QUICKREDUCT algorithm based on the previously described fuzzy-rough algorithm in Figure 12. FREQUICKREDUCT is similar to the fuzzy-rough algorithm but uses the entropy value of a data subset to guide the feature selection process. If the fuzzy entropy value of the current reduct candidate is smaller than the previous, then this reduct is retained and used in the next iteration of the loop. It is important to point out that the reduct is evaluated by examining its entropy value, termination only occurs when the addition of any remaining features results in a decrease in the dependency function value (γ'_{prev}). The fuzzy-entropy value therefore is not used as a termination criterion.

The algorithm begins with an empty subset R and with γ'_{prev} initialised to zero. The do-until loop works by examining the entropy value of a subset and incrementally adding one conditional feature at a time, until the dependency function value begins to fall to a value that is lower or equal to that of the last subset. For each iteration, a conditional feature that has not already been evaluated will be temporarily added to the subset R. The entropy of the subset currently being examined (5) is then evaluated and compared with the entropy of T, (the previous subset). If the entropy value of the current subset is lower (6), then the attribute added in (5) is retained as part of the new reduct T (*7*). The loop continues to evaluate in the above manner by adding conditional features, until the dependency value of the current reduct candidate ($\gamma'_R(D)$) falls to a value lower than or equal to that of the previously evaluated reduct candidate.

Application to the Example Dataset

Employing the same setup as before, but with $D = \{e\}$ the following partitions are obtained:

$U/A = \{N_a, Z_a\}$
$U/B = \{N_b, Z_b\}$
$U/C = \{N_c, Z_c\}$
$U/D = \{\{1,3,6\}, \{2,4,5\}\} = \{Q_1, Q_2\}$

The algorithm begins with an empty subset, and considers the addition of individual features. The attribute that results in the greatest decrease in fuzzy entropy will ultimately be added to the reduct candidate. For attribute a, the fuzzy entropy is calculated as follows ($A = \{a\}$):

$$E(A) = \frac{|N_a|}{|N_a + Z_a|} H(N_a) + \frac{|Z_a|}{|N_a + Z_a|} H(Z_a)$$

For the first part of the summation, the value $H(N_a)$ must be determined. This is achieved in the following way:

$$\begin{aligned} H(N_a) &= \sum_{Q \in U/D} -p(Q \mid N_a) \log_2 p(Q \mid N_{ai}) \\ &= -p(Q_1 \mid N_a) \log_2 p(Q_1 \mid N_a) - p(Q_2 \mid N_a) \log_2 p(Q_2 \mid N_a) \end{aligned}$$

The required probabilities are $p(Q_1 \mid N_a) = 0.6363637$, $p(Q_2 \mid N_a) = 0.3636363$. Hence, $H(N_a) = 0.94566023$. In a similar way, $H(Z_a)$ can be calculated, giving a value of 1.0. To determine the fuzzy entropy for a, the values $\frac{|N_a|}{|N_a + Z_a|}$ and $\frac{|Z_a|}{|N_a + Z_a|}$ must also be determined. This is achieved through the standard fuzzy cardinality, resulting in a fuzzy entropy value of:

$$\begin{aligned} E(A) &= (0.47826084 \times H(N_a)) + (0.5217391 \times H(Z_a)) \\ &= (0.47826084 \times 0.94566023) + (0.5217391 \times 1.0) \\ &= 0.9740114 \end{aligned}$$

Repeating this process for the remaining attributes gives:

$$\begin{aligned} E(B) &= 0.99629750 \\ E(C) &= 0.99999994 \end{aligned}$$

From this it can be seen that attribute a will cause the greatest decrease in fuzzy entropy. This attribute is chosen and added to the potential reduct, $R \leftarrow R \cup \{a\}$. This subset is then evaluated using the fuzzy-rough dependency measure, resulting in $\gamma_R(D) = 0.3333333$. The previous dependency value is 0 (the algorithm started with the empty set), hence the search continues. The process iterates and the two fuzzy entropy values calculated are

$E(\{a, b\}) = 0.7878490$

$E(\{a, c\}) = 0.9506136$

Adding attribute b to the reduct candidate causes the larger decrease of fuzzy entropy, so the new candidate becomes $\{a,b\}$. The resulting dependency value for this, $\gamma_{\{a,b\}}(D)$, is 0.56666666. This is, again, larger than the previous dependency value, and so search continues. Lastly, attribute c is added to the potential reduct:

$E(\{a, b, c\}) = 0.7412282$
$(\gamma_{\{a,b,c\}}(D) = 0.56666666)$

As this causes no increase in dependency, the algorithm stops and outputs the reduct $\{a,b\}$. The dataset can now be reduced to only those attributes appearing in the reduct.

Tolerance Rough Sets

Another way of attempting to handle the problem of real-valued data is to introduce a measure of similarity of feature values and define the lower and upper approximations based on these similarity measures. Such lower and upper approximations define tolerance rough sets (Skowron & Stepaniuk, 1996). By relaxing the transitivity constraint of equivalence classes, a further degree of flexibility (with regard to indiscernibility) is introduced. In traditional rough sets, objects are grouped into equivalence classes if their attribute values are equal. This requirement might be too strict for real-world data, where values might differ only as a result of noise.

Similarity Measures

For the tolerance-based approach, suitable similarity relations must be defined for each attribute, although the same definition can be used for all features if applicable. A standard measure for this purpose, given in (Stepaniuk, 1998), is:

Equation 27

$$SIM_a(x, y) = 1 - \frac{|a(x) - a(y)|}{|a_{\max} - a_{\min}|}$$

where a is the attribute under consideration, and $a_{\max}$ and $a_{\min}$ denote the maximum and minimum values respectively for this attribute. When considering more than one attribute, the defined similarities must be combined to provide a measure of the overall similarity of objects. For a subset of features, P, this can be achieved in many ways; two commonly adopted approaches are:

Equation 28

$$(x, y) \in SIM_{P,\tau} \text{ iff } \prod_{a \in P} SIM_a(x, y) \geq \tau$$

Equation 29

$$(x, y) \in SIM_{P,\tau} \text{ iff } \frac{\sum_{a \in P} SIM_a(x,y)}{|P|} \geq \tau$$

where $\tau \in [0,1]$ is a global similarity threshold: τ determines the required level of similarity for inclusion within tolerance classes. It can be seen that this framework allows for the specific case of traditional rough sets by defining a suitable similarity measure (e.g. equality of feature values and Equation 28) and threshold ($\tau = 1$). Further similarity relations are investigated in (Nguyen & Skowron, 1997a).

Tolerance classes generated by the similarity relation for an object *x* are defined as:

Equation 30

$$SIM_{P,\tau}(x) = \{y \in U \mid (x,y) \in SIM_{P,\tau}\}$$

Approximations and Dependency

Lower and upper approximations are then defined in a similar way to traditional rough set theory:

Equation 31

$$\underline{P_\tau} X = \{x \mid SIM_{P,\tau}(x) \subseteq X\}$$

Equation 32

$$\overline{P_\tau} X = \{x \mid SIM_{P,\tau}(x) \cap X \neq \emptyset\}$$

Positive region and dependency functions then become:

Equation 33

$$POS_{P,\tau}(Q) = \bigcup_{X \in U/\overline{Q}} \underline{P_\tau} X$$

Equation 34

$$\gamma_{P,\tau}(Q) = \frac{|POS_{P,\tau}(Q)|}{|U|}$$

From these definitions, attribute reduction methods can be constructed that use the tolerance-based degree of dependency, $\gamma_{P,\tau}(Q)$, to gauge the significance of feature subsets. For example, the fuzzy-rough QUICKREDUCT algorithm (Figure 12) can be adapted to perform feature selection based on the tolerance rough set-based measure. The resulting algorithm can be found in Figure 15.

TOLQUICKREDUCT(C,D,τ)
C, the set of all conditional features;
D, the set of decision features;
τ, similarity threshold.

(1) $R \leftarrow \{\}; \gamma^{\tau}_{best} = 0$
(2) **do**
(3) $T \leftarrow R$
(4) $\gamma^{\tau}_{prev} = \gamma^{\tau}_{best}$
(5) $\forall x \in (C - R)$
(6) **if** $\gamma_{R \cup \{x\},\tau}(D) > \gamma_{T,\tau}(D)$
(7) $T \leftarrow R \cup \{x\}$
(8) $\gamma^{\tau}_{best} = \gamma_{T,\tau}(D)$
(9) $R \leftarrow T$
(10) **until** $\gamma^{\tau}_{best} == \gamma^{\tau}_{prev}$
(11) **return** R

Figure 15 Tolerance QUICKREDUCT algorithm

Application to the Example Dataset

To illustrate the operation of the tolerance QUICKREDUCT algorithm, it is applied to the example data given in Table 3. For this example, the similarity measure is the same as that given in Equation 27 and Equation 28 for all conditional attributes, with $\tau = 0.7$. This choice of threshold permits attribute values to differ to a limited extent, allowing close values to be considered as identical. For the decision feature, τ is set to 1 (i.e. objects must have identical values to appear in the same tolerance class) as the decision value is nominal. Setting $A = \{a\}$, $B = \{b\}$, $C = \{c\}$ and $Q = \{q\}$, the following tolerance classes are obtained:

$U/SIM_{A,\tau} = \{\{1,2,3\},\{4,5,6\}\}$
$U/SIM_{B,\tau} = \{\{1,3,4,5\},\{2\},\{6\}\}$
$U/SIM_{C,\tau} = \{\{1\},\{2,4,5,6\},\{3\}\}$
$U/SIM_{Q,\tau} = \{\{1,3,6\}, \{2,4,5\}\}$
$U/SIM_{\{a,b\},\tau} = \{\{1,3\}, \{2\}, \{4,5\}, \{4,5,6\}, \{5,6\}\}$
$U/SIM_{\{b,c\},\tau} = \{\{1,3\}, \{2,6\}, \{4,5,6\}, \{2,4,5,6\}\}$
$U/SIM_{\{a,b,c\},\tau} = \{\{1,3\}, \{2\},\{4,5,6\}\}$

It can be seen here that some objects belong to more than one tolerance class. This is due to the additional flexibility of employing similarity measures rather than strict equivalence.

Based on these partitions, the degree of dependency can be calculated for attribute subsets, providing an evaluation of their significance. The tolerance QUICKREDUCT algorithm considers the addition of attributes to the currently stored best subset (initially the empty set) and selects the feature that results in the highest increase of

this value. Considering attribute b the lower approximations of the decision classes are calculated as follows:

$$\underline{B_\tau}\{1,3,6\} = \{x \mid SIM_{B,\tau}(x) \subseteq \{1,3,6\}\} = \{6\}$$
$$\underline{B_\tau}\{2,4,5\} = \{x \mid SIM_{B,\tau}(x) \subseteq \{1,3,6\}\} = \{2\}$$

Hence, the positive region can be constructed:

$$\begin{aligned} POS_{B,\tau}(Q) &= \bigcup_{X \in U/Q} \underline{B_\tau} X \\ &= \underline{B_\tau}\{1,3,6\} \cup \underline{B_\tau}\{2,4,5\} \\ &= \{2,6\} \end{aligned}$$

And the resulting degree of dependency is:

$$\begin{aligned} \gamma_{B,\tau}(Q) &= \frac{|\, POS_{B,\tau}(Q)\,|}{|\,U\,|} \\ &= \frac{|\,\{2,6\}\,|}{|\,\{1,2,3,4,5,6\}\,|} = \frac{2}{6} \end{aligned}$$

For the other conditional features in the dataset, the corresponding dependency degrees are:

$$\gamma_{A,\tau}(Q) = \frac{|\,\{\varnothing\}\,|}{|\,\{1,2,3,4,5,6\}\,|} = \frac{0}{6}$$
$$\gamma_{C,\tau}(Q) = \frac{|\,\{1,3\}\,|}{|\,\{1,2,3,4,5,6\}\,|} = \frac{2}{6}$$

Following the tolerance QUICKREDUCT algorithm, attribute b is added to the reduct candidate ($R = \{b\}$) and the search continues. The algorithm makes an arbitrary choice here between attributes b and c as they produce equally high degrees of dependency (although they generate different positive regions). As attribute b was considered before attribute c, it is selected. The algorithm continues by evaluating subsets containing this attribute in combination with the remaining individual attributes from the dataset.

$$\gamma_{\{a,b\},\tau}(Q) = \frac{|\,\{1,2,3,4,5\}\,|}{|\,\{1,2,3,4,5,6\}\,|} = \frac{5}{6}$$
$$\gamma_{\{b,c\},\tau}(Q) = \frac{|\,\{1,3\}\,|}{|\,\{1,2,3,4,5,6\}\,|} = \frac{2}{6}$$

The subset $\{a,b\}$ is chosen as this results in a higher dependency degree than $\{b\}$. The algorithm then evaluates the combination of this subset with the remaining attributes (in this example only one attribute, c, remains):

$$\gamma_{\{a,b,c\},\tau}(Q) = \frac{|\{1,2,3\}|}{|\{1,2,3,4,5,6\}|} = \frac{3}{6}$$

As this value is less than that for subset $\{a,b\}$, the algorithm terminates and outputs the reduct $\{a,b\}$. This is the same subset as that found by the fuzzy-rough method. However, for tolerance-based FS techniques, a suitable similarity measure must be defined for *all* attributes that are considered and an appropriate value for τ must be determined.

ALTERNATIVE SEARCH MECHANISMS

GA-based Approaches

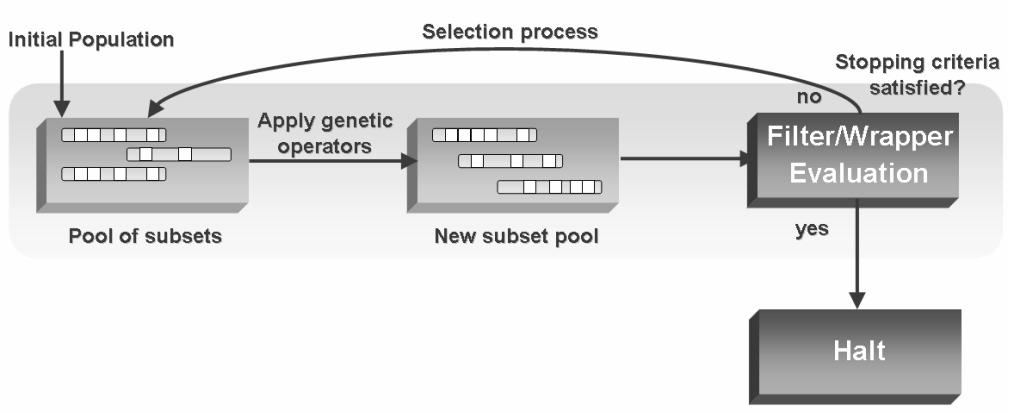

Figure 16 Feature selection with genetic algorithms

Genetic Algorithms (GAs) (Holland, 1975) are generally quite effective for rapid search of large, nonlinear and poorly understood spaces. Unlike classical feature selection strategies where one solution is optimized, a population of solutions can be modified at the same time (Siedlecki & Sklansky, 1989; Kudo & Skalansky, 2000). This can result in several optimal (or close-to-optimal) feature subsets as output.

A feature subset is typically represented by a binary string with length equal to the number of features present in the dataset. A zero or one in the *j*th position in the chromosome denotes the absence or presence of the *j*th feature in this particular subset. The general process for feature selection using GAs can be seen in Figure 16.

An initial population of chromosomes is created; the size of the population and how they are created are important issues. From this pool of feature subsets, the typical genetic operators (crossover and mutation) are applied. Again, the choice of which types of crossover and mutation used must be carefully considered, as well as their probabilities of application. This generates a new feature subset pool which may be evaluated in two different ways. If a filter approach is adopted, the fitness of individuals is calculated using a suitable criterion function. This function evaluates the goodness of a feature subset; a larger value indicates a better subset. Such a criterion function could be Shannon's entropy measure (Quinlan, 1993) or the dependency function from rough set theory (Pawlak, 1991).

For the wrapper approach, chromosomes are evaluated by inducing a classifier based on the feature subset, and obtaining the classification accuracy (or an estimate of it) on the data (Smith & Bull, 2003). To guide the search toward minimal feature subsets, the subset size is also incorporated into the fitness function of both filter and wrapper methods. Indeed, other factors may be included that are of interest, such as the cost of measurement for each feature etc. GAs may also learn rules directly, and in the process perform feature selection (Cordón et al., 1999; Jin, 2000; Xiong & Litz, 2002).

A suitable stopping criterion must be chosen. This is typically achieved by limiting the number of generations that take place or by setting some threshold which must be exceeded by the fitness function. If the stopping criterion is not satisfied, then individuals are selected from the current subset pool and the process described above repeats.

As with all feature selection approaches, GAs can get caught in local minima, missing a dataset's true minimal feature subset. Also, the fitness evaluation can be very costly as there are many generations of many feature subsets that must be evaluated. This is particularly a problem for wrapper approaches where classifiers are induced and evaluated for each chromosome.

The approaches reported in (Bjorvand & Komorowski, 1997; Wróblewski, 1995) 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 *i*th 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 to FS 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 each bit string's fitness, 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 mainly on the population size.

Simulated Annealing-based

Annealing is the process by which a substance is heated (usually melted) and cooled slowly in order to toughen and reduce brittleness. For example, this process is used for a metal to reach a configuration of minimum energy (a perfect, regular crystal). If the metal is annealed too quickly, this perfect organisation is unable to be achieved throughout the substance. Parts of the material will be regular, but these will be separated by boundaries where fractures are most likely to occur.

Simulated Annealing (SA) (Kirkpatrick et al., 1983) is a stochastic optimization technique that is based on the computational imitation of this process of annealing. It is concerned with the change of energy (cost) of a system. In each algorithmic step, an "atom" (a feature subset in FS) is given a small random displacement and the resulting change of energy, ΔE, is calculated. If $\Delta E \leq 0$, this new state is allowed and the process continues. However if $\Delta E > 0$, the probability that this new state is accepted is:

Equation 35

$$P(\Delta E) = e^{-\frac{\Delta E}{T}}$$

As the temperature, T, is lowered, the probability of accepting a state with a positive change in energy reduces. In other words, the willingness to accept a bad move decreases. The conversion of a combinatorial optimization problem into the SA framework involves the following:

1) *Concise configuration description*. The representation of the problem to be solved should be defined in a way that allows solutions to be constructed easily and evaluated quickly.
2) *Random move generator*. A suitable random transformation of the current state must be defined. Typically the changes allowed are small to limit the extent of search to the vicinity of the currently considered best solution. If this is not limited, the search degenerates to a random unguided exploration of the search space.
3) *Cost function definition.* The cost function (i.e. the calculation of the state's energy) should effectively combine the various criteria that are to be optimized for the problem. This function should be defined in such a way that smaller function values indicate better solutions.
4) *Suitable annealing schedule*. As with the real-world annealing process, problems are encountered if the initial temperature is too low, or if annealing takes place too quickly. Hence, an annealing schedule must be defined that avoids these pitfalls. The schedule is usually determined experimentally.

To convert the feature selection task into this framework, a suitable representation must be used. Here, the states will be feature subsets. The random moves can be produced by randomly mutating the current state with a low probability. This may also remove features from a given feature subset, allowing the search to progress both forwards and backwards. The cost function must take into account both the evaluated subset "goodness" (by a filter evaluation function or a wrapper classifier accuracy) and also the subset size. The annealing schedule can be determined by experiment, although a good estimate may be $T(0) = |C|$ and $T(t+1) = \alpha * T(t)$, with $\alpha \geq 0.85$. Here t is the number of iterations and α determines the rate of cooling.

SAFS(T_0, T_{min}, α, L_k)
T_0, the initial temperature;
T_{min}, the minimum allowed temperature;.
α, the extent of temperature decrease;
L_k, the extent of local search

(1) $R \leftarrow$ genInitSol()
(2) **while** $T(t) > T_{\min}$
(3) **for** i=1,…, L_k
(4) $S \leftarrow$ genSol(R)
(5) $\Delta E = \text{cost}(S)$
(6) **if** $\Delta E \leq 0$
(7) $M \leftarrow S$
(8) **else if** $P(\Delta E) > randNumber()$
(9) $M \leftarrow S$
(10) $R \leftarrow M$
(11) $T(t+1) = \alpha * T(t)$
(12) **output** R

Figure 17 Simulated annealing-based feature selection

The SA-based feature selection algorithm can be seen in Figure 17. This differs slightly from the general SA algorithm in that there is a measure of local search employed at each iteration, governed by the parameter L_k. An initial solution is created, from which the next states are derived by random mutations and evaluated. The best state is remembered and used for processing in the next cycle. The chosen state may not actually be the best state encountered in this loop, due to the probability $P(\Delta E)$ that a state is chosen randomly (which will decrease over time). The temperature is decreased according to the annealing schedule and the algorithm continues until the lowest allowed temperature has been exceeded.

Problems with this approach include how to define the annealing schedule correctly. If α is too high, the temperature will decrease slowly, allowing more frequent jumps to higher energy states, slowing convergence. However, if α is too low, the temperature decreases too quickly and the system will converge to local minima (equivalent to brittleness in the case of metal annealing). Also, the cost function definition is critical - there must be a balancing of the importance assigned to the different evaluation criteria involved. Biasing one over another will have the effect of directing search toward solutions that optimize that criterion only.

SimRSAR employs a simulated annealing-based feature selection mechanism to locate rough set reducts (Jensen & Shen, 2004b). The states are feature subsets, with random state mutations set to changing three features (either adding or removing them). The cost function attempts to maximize the rough set dependency (γ) whilst minimizing the subset cardinality. The cost of subset R is defined as:

Equation 36

$$\text{cost}(R) = \left[\frac{\gamma_C(D) - \gamma_R(D)}{\gamma_C(D)} \right]^a + \left[\frac{|R|}{|C|} \right]^b$$

where *a* and *b* are defined in order to weight the contributions of dependency and subset size to the overall cost measure.

Ant Colony Optimization-based

Swarm Intelligence (SI) is the property of a system whereby the collective behaviours of simple agents interacting locally with their environment cause coherent functional global patterns to emerge (Bonabeau, 1999). SI provides a basis with which it is possible to explore collective (or distributed) problem solving without centralized control or the provision of a global model. One area of interest in SI is Particle Swarm Optimization (Kennedy & Eberhart, 1995), a population-based stochastic optimization technique. Here, the system is initialised with a population of random solutions, called particles. Optima are searched for by updating generations, with particles moving through the parameter space towards the current local and global optimum particles. At each time step, the velocities of all particles are changed depending on the current optima.

Ant Colony Optimization (ACO) (Bonabeau, 1999) is another area of interest within SI. In nature, it can be observed that real ants are capable of finding the shortest route between a food source and their nest without the use of visual information and hence possess no global world model, adapting to changes in the environment. The deposition of pheromone is the main factor in enabling real ants to find the shortest routes over a period of time. Each ant probabilistically prefers to follow a direction rich in this chemical. The pheromone decays over time, resulting in much less pheromone on less popular paths. Given that over time the shortest route will have the higher rate of ant traversal, this path will be reinforced and the others diminished until all ants follow the same, shortest path (the "system" has converged to a single solution). It is also possible that there are many equally short paths. In this situation, the rates of ant traversal over the short paths will be roughly the same, resulting in these paths being maintained while others are ignored. Additionally, if a sudden change to the environment occurs (e.g. a large obstacle appears on the shortest path), the ACO system can respond to this and will eventually converge to a new solution. Based on this idea, artificial ants can be deployed to solve complex optimization problems via the use of artificial pheromone deposition.

ACO is particularly attractive for feature selection as there seems to be no heuristic that can guide search to the optimal minimal subset every time. Additionally, it can be the case that ants discover the best feature combinations as they proceed throughout the search space

ACO Framework

An ACO algorithm can be applied to any combinatorial problem as far as it is possible to define:

1) *Appropriate problem representation*. The problem can be described as a graph with a set of nodes and edges between nodes.
2) *Heuristic desirability (η) of edges*. A suitable heuristic measure of the "goodness" of paths from one node to every other connected node in the graph.
3) *Construction of feasible solutions*. A mechanism must be in place whereby possible solutions are efficiently created. This requires the definition of a suitable traversal stopping criterion to stop path construction when a solution has been reached.
4) *Pheromone updating rule*. A suitable method of updating the pheromone levels on edges is required with a corresponding evaporation rule, typically involving the selection of the *n* best ants and updating the paths they chose.
5) *Probabilistic transition rule*. The rule that determines the probability of an ant traversing from one node in the graph to the next.

Each ant in the artificial colony maintains a memory of its history - remembering the path it has chosen so far in constructing a solution. This history can be used in the evaluation of the resulting created solution and may also contribute to the decision process at each stage of solution construction.

Two types of information are available to ants during their graph traversal, local and global, controlled by the parameters β and α respectively. Local information is obtained through a problem-specific heuristic measure. The extent to which the measure influences an ant's decision to traverse an edge is controlled by the parameter β. This will guide ants towards paths that are likely to result in good solutions. Global knowledge is also available to ants through the deposition of artificial pheromone on the graph edges by their predecessors over time. The impact of this knowledge on an ant's traversal decision is determined by the parameter α. Good paths discovered by past ants will have a higher amount of associated pheromone. How much pheromone is deposited, and when, is dependent on the characteristics of the problem. No other local or global knowledge is available to the ants in the standard ACO model, though the inclusion of such information by extending the ACO framework has been investigated (Bonabeau, 1999).

Feature Selection

The feature selection task may be reformulated into an ACO-suitable problem (Jensen & Shen, 2005; Jensen, 2006). ACO requires a problem to be represented as a graph - here nodes represent features, with the edges between them denoting the choice of the next feature. The search for the optimal feature subset is then an ant traversal through the graph where a minimum number of nodes are visited that satisfies the traversal stopping criterion. Figure 18 illustrates this setup - the ant is currently at node *a* and has a choice of which feature to add next to its path (dotted lines). It chooses feature *b* next based on the transition rule, then *c* and then *d*. Upon arrival at *d*, the current subset $\{a,b,c,d\}$ is determined to satisfy the traversal stopping criteria (e.g. a suitably high classification accuracy has been achieved with this subset, assuming that the selected features are used to classify certain objects). The ant terminates its traversal and outputs this feature subset as a candidate for data reduction.

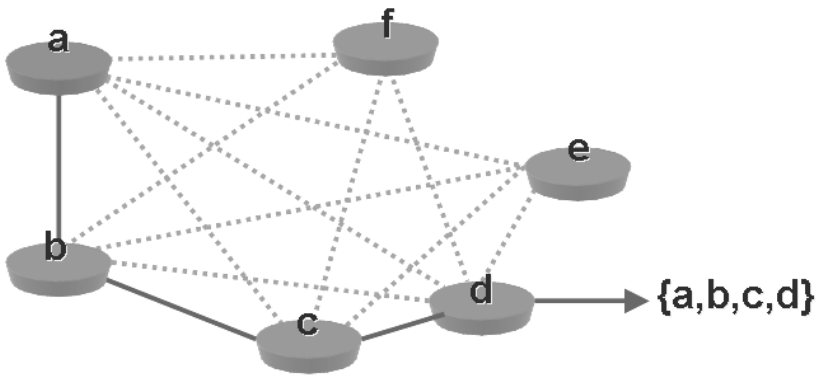

Figure 18 ACO problem representation for feature selection

A suitable heuristic desirability of traversing between features could be any subset evaluation function - for example, an entropy-based measure (Quinlan, 1993) or the fuzzy-rough set dependency measure. Depending on how optimality is defined for the particular application, the pheromone may be updated accordingly. For instance, subset minimality and ``goodness'' are two key factors so the pheromone update should be proportional to ``goodness'' and inversely proportional to size. How "goodness" is determined will also depend on the application. In some cases, this may be a heuristic evaluation of the subset, in others it may be based on the resulting classification accuracy of a classifier produced using the subset.

The heuristic desirability and pheromone factors are combined to form the so-called probabilistic transition rule, denoting the probability of an ant k at feature i choosing to move to feature j at time t:

Equation 37

$$p_{i,j}^{k}(t) = \frac{[\tau_{i,j}(t)]^{\alpha}.[\eta_{i,j}]^{\beta}}{\sum_{l \in J_i^k}[\tau_{i,l}(t)]^{\alpha}.[\eta_{i,l}]^{\beta}}$$

where J_i^k is the set of ant k's unvisited features, $\eta_{i,j}$ is the heuristic desirability of choosing feature j when at feature i and $\tau_{i,j}(t)$ is the amount of virtual pheromone on edge (i,j). The choice of α and β is determined experimentally. Typically, several experiments are performed, varying each parameter and choosing the values that produce the best results.

Selection Process

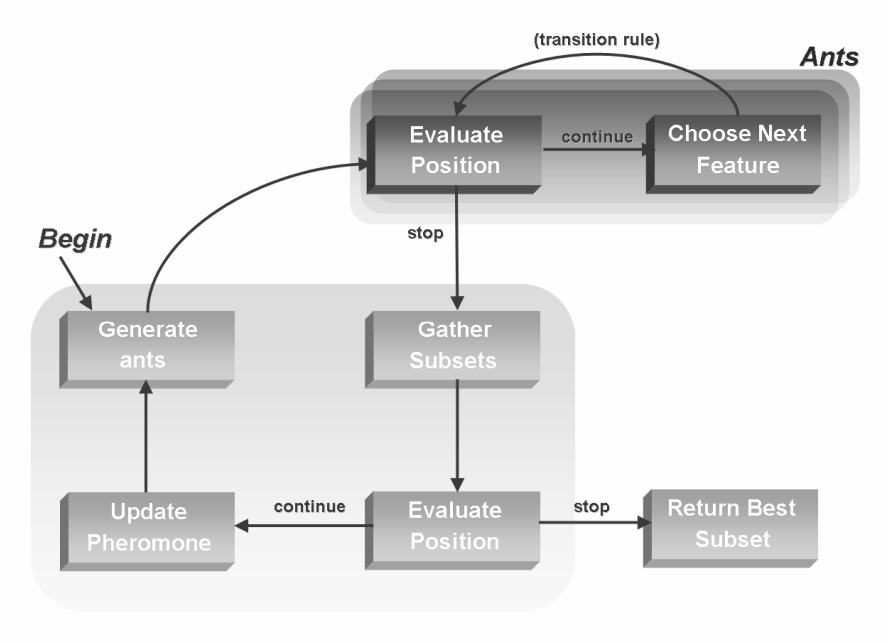

Figure 19 ACO-based feature selection overview

The overall process of ACO feature selection can be seen in Figure 19. It begins by generating a number of ants, *k*, which are then placed randomly on the graph (i.e. each ant starts with one random feature). Alternatively, the number of ants to place on the graph may be set equal to the number of features within the data; each ant starts path construction at a different feature. From these initial positions, they traverse edges probabilistically until a traversal stopping criterion is satisfied. The resulting subsets are gathered and then evaluated. If an optimal subset has been found or the algorithm has executed a certain number of times, then the process halts and outputs the best feature subset encountered. If neither condition holds, then the pheromone is updated, a new set of ants are created and the process iterates once more.

Complexity Analysis

The time complexity of the ant-based approach to feature selection is $O(IAk)$, where I is the number of iterations, A the number of original features, and k the number of ants. In the worst case, each ant selects all the features. As the heuristic is evaluated after each feature is added to the reduct candidate, this will result in A evaluations per ant. After one iteration in this scenario, Ak evaluations will have been performed. After I iterations, the heuristic will be evaluated IAk times.

Pheromone Update

Depending on how optimality is defined for the particular application, the pheromone may be updated accordingly. To tailor this mechanism to find rough set reducts, it is necessary to use the dependency measure as the stopping criterion. This means that an ant will stop building its feature subset when the dependency of the subset reaches the maximum for the dataset (the value 1 for consistent datasets). The dependency function may also be chosen as the heuristic desirability measure, but this is not necessary. In fact, it may be of more use to employ a non-rough set related heuristic for this purpose. By using an alternative measure such as an entropy-based heuristic,

the method may avoid feature combinations that may mislead the rough set-based heuristic.

The pheromone on each edge is updated according to the following formula:

$$\tau_{i,j}(t+1) = (1-\rho).\tau_{i,j}(t) + \Delta\tau_{i,j}(t)$$

where

$$\Delta\tau_{i,j}(t) = \sum_{k=1}^{n} \frac{\gamma_{S^k}(D)}{|S^k|}$$

This is the case if the edge (i,j) has been traversed; $\Delta\tau_{i,j}(t)$ is 0 otherwise. The value ρ is a decay constant used to simulate the evaporation of the pheromone, S^k is the feature subset found by ant k. The pheromone is updated according to both the rough set measure of the goodness of the ant's feature subset and the size of the subset itself. By this definition, all ants update the pheromone. Alternative strategies may be used for this, such as allowing only the ants with the currently best feature subsets to proportionally increase the pheromone.

Results in (Jensen, 2006) show the effectiveness of the ACO-based approach for finding reducts both for crisp rough set reduction and fuzzy-rough selection. This technique regularly located optimal subsets (in terms of the subset size and corresponding dependency degree).

Particle Swarm Optimization-based

Particle swarm optimization (PSO) is an evolutionary computation technique (Kennedy & Eberhart, 1995). The original intent was to graphically simulate the movement of bird flocking behaviour. (Shi & Eberhart, 1998) introduced the concept of inertia weight into the particle swarm optimizer to produce the standard PSO algorithm.

Standard PSO algorithm

PSO is initialized with a population of particles. Each particle is treated as a point in an S-dimensional space. The ith particle is represented as $X_i = (x_{i1}, x_{i2}, ..., x_{iS})$. The best previous position (p_{best}, the position giving the best fitness value) of any particle is $P_i = (p_{i1}, p_{i2}, ..., p_{iS})$. The index of the global best particle is represented by g_{best}. The velocity for particle i is $V_i = (v_{i1}, v_{i2}, ..., v_{iS})$. The particles are manipulated according to the following:

Equation 38

$$v_{id} = w \times v_{id} + c_1 \times rand() \times (p_{id} - x_{id}) + c_2 \times Rand() \times (p_{gd} - x_{id})$$

Equation 39

$$x_{id} = x_{id} + v_{id}$$

where w is the inertia weight. Suitable selection of the inertia weight provides a balance between global and local exploration. If a time varying inertia weight is employed, better performance can be expected (Shi & Eberhart, 1998). The acceleration constants c_1 and c_2 in Equation 38 represent the weighting of the stochastic acceleration terms that pull each particle toward p_{best} and g_{best} positions. Low values allow particles to roam far from target regions before being tugged back, while high values result in abrupt movement toward, or past, target regions. Particles' velocities on each dimension are limited to a maximum velocity V_{max}. If V_{max} is too small, particles may not explore sufficiently beyond locally good regions. If V_{max} is too high, particles might fly past good solutions.

The first part of Equation 38 provides the particles with a memory capability and the ability to explore new search space areas. The second part is the "cognition" part, which represents the private thinking of the particle itself. The third part is the "social" part, which represents the collaboration among the particles. This equation is used to update the particle's velocity, with the position of the particle updated according to Equation 39. The performance of each particle is then measured according to a pre-defined fitness function.

PSO-FS(*C, D*)
C, the set of conditional features;
D, the set of decision features.

(1) $\forall i : X_i \leftarrow$ randomPosition(); $V_i \leftarrow$ randomVelocity()
(2) $fit \leftarrow$ bestFit(X); globalbest $\leftarrow fit$; $p_{best} \leftarrow$ bestPos(X);
(3) $\forall i : P_i \leftarrow X_i$
(4) **while** (stopping criterion not met)
(5) **for** $i = 1,\ldots, S$ *// for each particle*
(6) **if** (fitness(i) > fit) *// local best*
(7) $fit \leftarrow$ fitness(i)
(8) $p_{best} \leftarrow X_i$
(9) **if** (fitness(i) > *globalbest*) *// global best*
(10) *globalbest* $\leftarrow$ fitness(i)
(11) $g_{best} \leftarrow X_i$**;** $R \leftarrow$ getReduct(X_i) *// convert to reduct*
(12) updateVelocity(); updatePosition()
(13) **output** R

Figure 20 PSO-FS algorithm

The PSO algorithm for feature selection can bee seen in Figure 20. Initially, a population of particles is constructed with random positions and velocities on S dimensions in the problem space. For each particle, the fitness function is evaluated. If the current particle's fitness evaluation is better than p_{best}, then this particle becomes the current best, and its position and fitness are stored. Next, the current particle's fitness is compared with the population's overall previous best fitness. If the current value is better than g_{best}, then this is set to the current particle's position, with the global best fitness updated. This position represents the best feature subset

encountered so far, and is thus converted and stored in R. The velocity and position of the particle is then updated according to Equation 38 and Equation 39. This process loops until a stopping criterion is met, usually a sufficiently good fitness or a maximum number of iterations (generations).

Encoding

To apply PSO to rough set reduction, the particle's position is represented as binary bit strings of length N, where N is the total number of attributes. This is the same representation as that used for GA-based feature selection. Therefore, each particle position is an attribute subset.

Representation of Velocity

Each particle's velocity is represented as a positive integer, varying between 1 and V_{max}. It implies how many of the particle's bit should be changed to be the same as that of the global best position, i.e. the velocity of the particle flying toward the best position. The number of different bits between two particles relates to the difference between their positions. For example, P_{gbest} = [1,0,1,1,1,0,1,0,0,1], X_i = [0,1,0,0,1,1,0,1,0,1]. The difference between the global best and the particle's current position is P_{gbest} -X_i = [1,–1,1,1,0,–1,1,–1,0,0]. '1' means that, compared with the best position, this bit (feature) should be selected but it is not, decreasing classification quality. On the other hand, '-1' means that compared with the best position, this bit should not be selected but it is. Both cases will lead to a lower fitness value.

Updating Position

After the updating of velocity, a particle's position will be updated by the new velocity. If the new velocity is V, and the number of different bits between the current particle and g_{best} is xg, there exist two situations while updating the position:

1) $V \leq xg$. In such a situation, randomly change V bits of the particle, which are different from that of g_{best}. The particle will move toward the global best while keeping its exploration ability.

2) $V > xg$. In this case, in addition to changing all the different bits to be the same as that of g_{best}, a further (V-xg) bits should be randomly changed. Hence, after the particle reaches the global best position, it keeps on moving some distance toward other directions, which gives it further exploration ability.

Velocity Limit

In experimentation, the particles' velocity was initially limited to [1, N]. However, it was noticed that in some cases after several generations swarms find good solutions (but not optimal ones), and in the following generations g_{best} remains stationary. Hence, only a sub-optimal solution is located. This indicates that the maximum velocity is too high and particles often 'fly past' the optimal solution. This can be prevented by setting V_{max} as (1/3)*N and so the velocity is limited to the range [1, (1/3)*N]. Once finding a global best position, other particles will adjust their velocities and positions, searching around the best position.

Fitness Function

The following can be used as the fitness function for directing search towards optimal reducts:

$$Fitness = \alpha * \gamma_R(D) + \beta * \frac{|C| - |R|}{|C|}$$

where $\gamma_R(D)$ is the classification quality of condition attribute set *R* relative to decision *D*, |*R*| is the length of the selected feature subset, and |*C*| is the total number of features. α and β are two parameters that correspond to the importance of classification quality and subset length, with $\alpha \in [0,1]$ and $\beta = 1 - \alpha$. Setting a high α value assures that the best position is at least a real rough set reduct. The goal then is to maximize fitness values.

Setting parameters

In the algorithm, the inertia weight decreases along with the iterations according to (Shi & Eberhart, 1998):

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{iter_{\max}} iter$$

where $w_{\max}$ is the initial value of the weighting coefficient, $w_{\min}$ is the final value of the weighting coefficient, $iter_{\max}$ is the maximum number of iterations or generations, and $iter$ is the current iteration or generation number.

Time Complexity

Let *N* be the number of features and *M* the total objects, then the complexity of the fitness function is $O(NM^2)$. The other impact on time is the number of generation iterations; however time is spent mainly evaluating the particles' positions.

Experimentation reported in (Wang et al., 2006) demonstrates the utility of PSO-based rough set feature selection. The method was compared against standard RSAR, EBR, a discernibility-based method, and a GA-based method. It regularly located better reducts, but tended to be more time-consuming than RSAR and EBR due to its non-deterministic nature.

CONCLUSION

This chapter has reviewed several techniques for feature selection based on rough set theory. Current methods tend to concentrate on alternative evaluation functions, employing rough set concepts to gauge subset suitability. These methods can be categorized into two distinct approaches: those that incorporate the degree of dependency measure (or extensions), and those that apply heuristic methods to generated discernibility matrices.

Methods based on traditional rough set theory do not have the ability to effectively manipulate continuous data. For these methods to operate, a discretization step must be carried out beforehand, which can often result in a loss of information. There are two main extensions to RST that handle this and avoid information loss: tolerance rough sets and fuzzy-rough sets. Both approaches replace crisp equivalence classes with alternatives that allow greater flexibility in handling object similarity.

Alternative search mechanisms have also been presented in this chapter, with a particular emphasis on stochastic approaches. These are of particular interest as no perfect heuristic exists that can always guide hill-climbing approaches to optimal subsets. Methods based on GAs, simulated annealing, ant colony optimization and particle swarm optimization were described.

Most of the effort in dependency degree-based feature selection has focussed on the use of lower approximations and positive regions in gauging subset suitability. However, there is still additional information that can be obtained through the use of upper approximations and the resulting boundary regions. The boundary region is of interest as this contains those objects whose concept membership is unknown – objects in the positive region definitely belong to a concept, and objects in the negative region do not belong. It is thought that by incorporating this additional information into the search process, better subsets should be located. Investigations into this are ongoing at the University of Wales, Aberystwyth.

An interesting direction for future work in discernibility matrix-based approaches might be the use of fuzzy propositional satisfiability. Traditionally, attributes appearing in objects are included in the discernibility matrix if their values differ. Based on these entries, a discernibility function is constructed, from which prime implicants are calculated or heuristic methods employed to find reducts. Currently, there is no flexibility in this process – discrete, noise-free data must be used and attributes either appear or are absent in matrix entries. This could be extended by considering fuzzy indiscernibility, where attributes belong to the discernibility matrix with varying degrees of membership based on fuzzy similarity. From this, fuzzy propositional satisfiability (or heuristic methods) could be used to determine fuzzy reducts for the system.

REFERENCES

Bakar, A. A., Sulaiman, M. N., Othman, M., & Selamat, M. H. (2002). Propositional Satisfiability Algorithm to Find Minimal Reducts for Data Mining. *International Journal of Computer Mathematics*, Vol. 79, No. 4, pp. 379–389.

Bazan, J., Skowron, A., & Synak, P. (1994). Dynamic reducts as a tool for extracting laws from decision tables. In *Proceedings of the 8th Symposium on Methodologies for Intelligent Systems*, Lecture Notes in Artificial Intelligence 869, Springer-Verlag, pp. 346–355.

Bazan, J. (1998). A Comparison of Dynamic and non-Dynamic Rough Set Methods for Extracting Laws from Decision Tables. In *Rough Sets in Knowledge Discovery*, Physica-Verlag, Heidelberg, pp. 321–365.

Beynon, M. J. (2000). An investigation of β-reduct selection within the variable precision rough sets model. In *Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing* (RSCTC 2000), pp 114–122.

Beynon, M. J. (2001). Reducts within the Variable Precision Rough Sets Model: A Further Investigation. *European Journal of Operational Research*, Vol. 134, No. 3, pp. 592–605.

Bjorvand, A. T., & Komorowski, J. (1997). Practical applications of genetic algorithms for efficient reduct computation. In *Proceedings of the 15th IMACS World Congress on Scientific Computation*, Modelling and AppliedMathematics, Vol. 4, pp. 601–606.

Bonabeau, E., Dorigo, M., & Theraulez, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press Inc., New York, NY, USA.

Chouchoulas, A., & Shen, Q. (2001). Rough set-aided keyword reduction for text categorisation. *Applied Artificial Intelligence*, Vol. 15, No. 9, pp. 843–873.

Cordón, O., del Jesus, M.J., & Herrera, F. (1999). Evolutionary approaches to the learning of fuzzy rule-based classification systems. In *Evolution of Engineering and Information Systems and Their Applications*. L.C. Jain (Ed.), CRC Press, pp. 107-160.

Dash, M., & Liu, H. (1997). Feature Selection for Classification. *Intelligent Data Analysis*, Vol. 1, No. 3, pp. 131-156.

Dubois, D., & Prade, H. (1992). Putting rough sets and fuzzy sets together. *Intelligent Decision Support*. Kluwer Academic Publishers, Dordrecht, pp. 203–232.

Han, J., Hu, X., Lin, T. Y. (2004). Feature Subset Selection Based on Relative Dependency between Attributes. *Rough Sets and Current Trends in Computing: 4th International Conference, RSCTC 2004*, Uppsala, Sweden, June 1-5, pp. 176–185.

Höhle, U. (1988). Quotients with respect to similarity relations. *Fuzzy Sets and Systems*, Vol. 27, No. 1, pp. 31–44.

Holland, J. (1975). *Adaptation In Natural and Artificial Systems*. The University of Michigan Press, Ann Arbour.

Hoos, H. H., & Stützle, T. (1999). Towards a Characterisation of the Behaviour of Stochastic Local Search Algorithms for SAT. *Artificial Intelligence*, Vol. 112, pp. 213–232.

Jensen, R. (2006). Performing Feature Selection with ACO. To appear in *Swarm Intelligence and Data Mining*, Springer SCI book series.

Jensen, R., & Shen, Q. (2004a). Fuzzy-Rough Attribute Reduction with Application to Web Categorization. *Fuzzy Sets and Systems*, Vol. 141, No. 3, pp. 469-485.

Jensen, R., & Shen, Q. (2004b). Semantics-Preserving Dimensionality Reduction: Rough and Fuzzy-Rough Based Approaches. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 12, pp.1457-1471.

Jensen, R., & Shen, Q. (2005). Fuzzy-Rough Data Reduction with Ant Colony Optimization. *Fuzzy Sets and Systems*, Vol. 149, No. 1, pp. 5–20.

Jensen, R., Shen, Q., & Tuson, A. (2005). Finding Rough Set Reducts with SAT. In *Proceedings of the 10th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, LNAI 3641, pp. 194-203.

Jin, Y. (2000). Fuzzy modeling of high-dimensional systems: complexity reduction and interpretability improvement. *IEEE Transactions on Fuzzy Systems*, Vol. 8, No. 2, pp. 212-221.

Kennedy, J., & Eberhart, R. C. 1995. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, Piscataway, NJ. pp. 1942-1948,.

Koczkodaj, W. W., Orlowski, M., & Marek, V. W. (1998). Myths about Rough Set Theory. *Communications of the ACM*, Vol. 41, No. 11, pp. 102-103.

Kosko, B. (1986). Fuzzy entropy and conditioning. *Information Sciences*, Vol. 40, No. 2, pp. 165–174.

Kryszkiewicz, M. (1994). Maintenance of reducts in the variable precision rough sets model. *ICS Research Report* 31/94, Warsaw University of Technology.

Kudo, M. & Skalansky, J. (2000). Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, Vol. 33, No. 1, pp. 25-41.

Langley, P. (1994). Selection of relevant features in machine learning. In *Proceedings of the AAAI Fall Symposium on Relevance*, pp. 1-5.

Mac Parthaláin, N., Jensen, R., & Shen, Q. (2006). Fuzzy entropy-assisted fuzzy-rough feature selection. To appear in *Proceedings of the 15th International Conference on Fuzzy Systems* (FUZZ-IEEE'06).

Modrzejewski, M. (1993). Feature selection using rough sets theory. In *Proceedings of the 11th International Conference on Machine Learning,* pp. 213–226.

Nguyen, S. H., & Nguyen, H. S. (1996). Some efficient algorithms for rough set methods. In *Proceedings of the Conference of Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pp. 1451–1456.

Nguyen, S. H., & Skowron, A. (1997a). Searching for Relational Patterns in Data. In *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery*, Trondheim, Norway, June 25-27, pp. 265–276.

Nguyen, H. S., & Skowron, A. (1997b). Boolean Reasoning for Feature Extraction Problems. In *Proceedings of the 10th International Symposium on Methodologies for Intelligent Systems*, pp. 117–126.

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

Pawlak, Z. (1991). *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishing, Dordrecht.

Polkowski, L., Lin, T. Y., & Tsumoto, S. (Eds). (2000). Rough Set Methods and Applications: New Developments in Knowledge Discovery in Information Systems, Vol. 56. *Studies in Fuzziness and Soft Computing*, Physica-Verlag, Heidelberg, Germany.

Polkowski, L. (2002). Rough Sets: Mathematical Foundations. *Advances in Soft Computing*. Physica Verlag, Heidelberg, Germany.

Quinlan, J. R. (1993). C4.5: Programs for Machine Learning. *The Morgan Kaufmann Series in Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA.

Shen, Q., & Jensen, R. (2004). Selecting Informative Feature with Fuzzy-Rough Sets and its Application for Complex Systems Monitoring. *Pattern Recognition*, Vol. 37, No. 7, pp. 1351-1363.

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. In Proceedings of the IEEE International Conference on Evolutionary Computation, Anchorage, AK, USA, pp. 69-73.

Siedlecki, W., & Sklansky, J. (1988). On automatic feature selection. *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 2, No. 2, pp. 197-220.

Siedlecki, W., & Sklansky, J. (1989). A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters*, Vol. 10, No. 5, pp. 335-347.

Skowron, A., & Rauszer, C. (1992). The discernibility matrices and functions in information systems. *Intelligent Decision Support*, Kluwer Academic Publishers, Dordrecht, pp. 331–362.

Skowron, A., & Stepaniuk, J. (1996). Tolerance Approximation Spaces. *Fundamenta Informaticae*, Vol. 27, No. 2, pp. 245–253.

Ślezak, D. (1996). Approximate reducts in decision tables. In *Proceedings of the 6th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems* (IPMU ’96), pp. 1159–1164.

Smith, M. G., & Bull, L. (2003). Feature construction and selection using genetic programming and a genetic algorithm. *Proceedings of the 6th European Conference on Genetic Programming (EuroGP 2003)*, pp. 229–237.

Starzyk, J. A., Nelson, D. E., & Sturtz, K. (2000). A Mathematical Foundation for Improved Reduct Generation in Information Systems. *Journal of Knowledge and Information Systems*, Vol. 2, No. 2, pp.131-146.

Stepaniuk, J. (1998). Optimizations of rough set model. *Fundamenta Informaticae*, Vol. 36, Issue 2-3, pp. 265–283.

Thiele, H. (1998). Fuzzy rough sets versus rough fuzzy sets - an interpretation and a comparative study using concepts of modal logics. Technical report no. CI- 30/98, University of Dortmund.

Wang, J., & Wang, J. (2001). Reduction Algorithms Based on Discernibility Matrix: The Ordered Attributes Method. *Journal of Computer Science & Technology*, Vol. 16, No. 6, pp. 489–504.

Wang, X. Y., Yang, J., Teng, X., Xia, W., & Jensen, R. (2006). Feature Selection based on Rough Sets and Particle Swarm Optimization. Accepted for publication in *Pattern Recognition Letters*.

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

Xiong, N. & Litz, L. (2002). Reduction of fuzzy control rules by means of premise learning - method and case study. *Fuzzy Sets and Systems*, Vol. 132, No. 2, pp. 217-231.

Yao, Y. Y. (1998). A Comparative Study of Fuzzy Sets and Rough Sets. *Information Sciences*, Vol. 109, No. 1-4, pp. 21–47.

Zadeh, L. A. (1975). The concept of a linguistic variable and its application to approximate reasoning. *Information Sciences*, Vol. 8, pp. 199–249, 301–357; Vol 9, pp. 43–80.

Zhang, L., & Malik, S. (2002). The Quest for Efficient Boolean Satisfiability Solvers. In *Proceedings of the 18th International Conference on Automated Deduction*, pp. 295–313.

Zhang, M., & Yao, J.T. (2004). A Rough Sets Based Approach to Feature Selection. *Proceedings of the 23rd International Conference of NAFIPS*, Banff, Canada, June 27-30, pp. 434–439.

Zhong, N., Dong, J., & Ohsuga, S. (2001). Using Rough Sets with Heuristics for Feature Selection. *Journal of Intelligent Information Systems*, Vol. 16, No. 3, pp. 199–214.

Ziarko, W. (1993). Variable Precision Rough Set Model. *Journal of Computer and System Sciences*, Vol. 46, No. 1, pp. 39–59.