Nature inspired feature selection meta-heuristics

Diao, Ren; Shen, Qiang

Published in:
Artificial Intelligence Review

DOI:
10.1007/s10462-015-9428-8

Publication date:
2015

Citation for published version (APA):
Diao, R., & Shen, Q. (2015). Nature inspired feature selection meta-heuristics. *Artificial Intelligence Review*, *44*(3), 311-340. https://doi.org/10.1007/s10462-015-9428-8

Noname manuscript No.
(will be inserted by the editor)

Nature Inspired Feature Selection Meta-Heuristics

Ren Diao · Qiang Shen

Received: date / Accepted: date

Abstract Many strategies have been exploited for the task of feature selection, in an effort to identify more compact and better quality feature subsets. A number of evaluation metrics have been developed recently that can judge the quality of a given feature subset as a whole, rather than assessing the qualities of individual features. Effective techniques of stochastic nature have also emerged, allowing good quality solutions to be discovered without resorting to exhaustive search. This paper provides a comprehensive review of the most recent methods for feature selection that originated from nature inspired meta-heuristics, where the more classic approaches such as genetic algorithms and ant colony optimisation are also included for comparison. A good number of the reviewed methodologies have been significantly modified in the present, in order to systematically support generic subset-based evaluators and higher dimensional problems. Such modifications are carried out because the original studies either work exclusively with certain subset evaluators (e.g., rough set-based methods), or are limited to specific problem domains. A total of ten different algorithms are examined, and their mechanisms and work flows are summarised in an unified manner. The performance of the reviewed approaches are compared using high dimensional, real-valued benchmark data sets. The selected feature subsets are also used to build classification models, in an effort to further validate their efficacies.

Keywords Feature selection · Dimensionality reduction · Nature inspired optimisation · Stochastic search

1 Introduction

The main aim of feature selection (FS) is to discover a minimal feature subset from a problem domain while retaining a suitably high accuracy in representing the original data (Dash and Liu, 1997). When analysing data that has a very large number of features (Xing et al, 2001), it is uneasy to identify and extract patterns or rules due to the high inter-dependency

Ren Diao · Qiang Shen
Department of Computer Science, Institute of Mathematics, Physics and Computer Science, Aberystwyth University, UK
E-mail: {rrd09, qqs}@aber.ac.uk

amongst individual features, or the behaviour of combined features, the so-called "curse-of-dimensionality" (Bellman, 1957). Techniques to perform tasks such as text processing, data classification and systems control (Mac Parthaláin et al, 2010a; ming Lee et al, 2001; Shang and Barnes, 2013; Karzynski et al, 2003; Shen and Jensen, 2004) can benefit greatly from FS, once the noisy, irrelevant, redundant or misleading features are removed (Jensen and Shen, 2009a).

Given a data set with N features, the task of FS can be seen as a search for "optimal" feature subsets through the competing 2^N candidates. Optimality is often subjective depending on the problem at hand. A subset that is selected as optimal using one particular evaluation function may not be equivalent to that of a subset selected by another. Various techniques have been developed in the literature to judge the quality of discovered feature subsets, several of which rank the features based on a certain importance measure (Kononenko et al, 1997), e.g., chi-square analysis (Zheng et al, 2004), rough set and fuzzy-rough set-based dependency (Jensen and Shen, 2007, 2008), information gain, and symmetrical uncertainty (Senthamarai Kannan and Ramaraj, 2010).

Recent trends in developing FS methods focus on evaluating a feature subset as a whole, forming an alternative type of approach to the aforementioned. Popular methods include the group-based fuzzy-rough FS (FRFS) (Jensen and Shen, 2009b; Mac Parthaláin et al, 2010b), probabilistic consistency-based FS (PCFS) (Dash and Liu, 2003), and correlation-based FS (CFS) (Hall, 1998). These techniques (together with the individual feature-based methods) are often collectively classified as the filter-based techniques. They are typically used as a preprocessing step, and is independent of any learning algorithm that may be subsequently employed. In contrast, wrapper-based (Hsu et al, 2002; Kohavi and John, 1997) and hybrid algorithms (Zhu et al, 2007) are used in conjunction with a learning or data mining algorithm, which is employed in place of an evaluation metric as used in the filter-based approach.

Independent of the learning mechanism, a common issue that all FS methods need to address is how to search for the "optimal" feature subsets. To this end, an exhaustive method may be used, however it is often impractical for most data sets. Alternatively, hill-climbing-based approaches are exploited where features are added or removed one at a time until there is no further improvement to the current candidate solution. Although generally fast to converge, these methods may lead to the discovery of sub-optimal subsets (both in terms of the evaluation score and the subset size) (Diao and Shen, 2012; Liu and Motoda, 2007). To avoid such short-comings, nature inspired meta-heuristics (NIMs) (Brownlee, 2011; Yang, 2008) such as genetic algorithms (Leardi et al, 1992; Wróblewski, 2001), genetic programming (Muni et al, 2006), simulated annealing (Debuse and Rayward-Smith, 1997), and particle swarm optimisation (Wang et al, 2007) have been utilised with varying degrees of success. Most of these algorithms are originally proposed to solve function optimisation problems and thus, require mapping concepts or internal mechanisms into the field of FS.

This paper presents an up-to-date survey of nature inspired approaches to FS. It establishes unified algorithmic notations to describe the procedures of the reviewed methods, and provides a common ground for comparing the performances of these algorithms, especially for high dimensional data sets, and computationally complex feature subset evaluators. The rest of the paper is structured as follows. Section 2 first formulates the problem domain of FS, and categorises the algorithms by their underlying inspirations. This section also points out a number of common concepts and terminologies that are used by various NIMs. Section 3 summarises the principles of the reviewed algorithms, and details the modifications made to several methods that allow them to work with more general types of evaluation mechanism. Pseudo-codes are also included in this section to unify the representation of the underlying algorithms. Section 4 compares the reviewed algorithms via systematic experimentation,

three feature subset-based evaluators of different characteristics are employed to demonstrate the efficacies of these approaches. Section 5 concludes the paper and suggests several future directions.

2 Background

2.1 Feature Selection

In the context of FS, an information system is a couple (X,Y), where X is a non-empty set of finite objects (the universe), and Y is a non-empty, finite set of features. For decision systems, $Y = \{A \cup Z\}$ where $A = \{a_1, \cdots, a_{|A|}\}$ is the set of input features, and Z is the set of decision features. Features can be either qualitative (discrete-valued) or quantitative (real-valued). The ultimate aim of a feature subset search algorithm is to determine a set of features $B \subseteq A$, which has the highest evaluation score $f(B)$, and minimum subset size $|B|$, such that B may encapsulate the original concept to the maximum extend, and be able to distinguish the training instances into their respective classes. Here $f : \mathbb{B} \rightarrow \mathbb{R}$ is a subset evaluation function, which maps a set of feature subsets onto the set of real numbers (the evaluation scores). Several existing techniques such as CFS, PCFS, or FRFS output a normalised score $f(B) \in [0,1], f(\emptyset) = 0$, where higher scores indicate better quality subsets. For a given data set, multiple subsets may exist that are equally (or almost equally) optimal, when judged by a subset evaluator, i.e., $f(B^p) = f(B^q)$ (or $f(B^p) \simeq f(B^q)$), $B^p \neq B^q$, where p and q denote the indices of two of the possible solutions.

2.2 Taxonomy of Algorithms

Following a taxonomy concerning these NIMs in their base form (Brownlee, 2011), the existing nature inspired FS approaches can be classified into a number of categories, as shown in Fig. 1. Considering that a few categories have not yet attracted sufficient applications in the area of FS, e.g., the immune systems and physical/social algorithms, three major categories are established here in order to improve the organisation of the reviewed methods. The biologically-inspired approaches include the genetic algorithm (GA) (Siedlecki and Sklansky, 1989; Sklansky and Vriesenga, 1996; Yang and Honavar, 1998), genetic programming (GP) (Muni et al, 2006), memetic algorithm (MA) (Yang et al, 2008; Zhu and Ong, 2007), and the clonal selection algorithm (CSA) (Shojaie and Moradi, 2008) from immune systems; the physical, social and stochastic algorithms include harmony search (HS) (Diao and Shen, 2012), simulated annealing (SA) (Ekbal et al, 2011; Meiri and Zahavi, 2006), random search (RS) (Stracuzzi and Utgoff, 2004), scatter search (SS) (Lpez et al, 2006), and tabu search (TS) (Hedar et al, 2008); and the swarm-based techniques include artificial bee colony (ABC) (Palanisamy and S, 2012), ant colony optimisation (ACO) (Chen et al, 2010; Jensen and Shen, 2005; Kabir et al, 2012; Ke et al, 2008), bat algorithm (BA) (Nakamura et al, 2012), bee colony optimisation (BCO), and particle swarm optimisation (PSO), etc. Most of the above mentioned algorithms are described in detail in the following sections.

Fundamentally speaking, putting the underlying analogies aside, nature inspired FS approaches are a collection of techniques of stochastic nature, for the purpose of discovering and improving good candidate solutions. Several recent studies combined these algorithms together, adopting a given algorithm's strong point to complement another's weakness. In so doing, a number of hybrid methods have emerged, including GA-PSO (Atyabi et al, 2012),

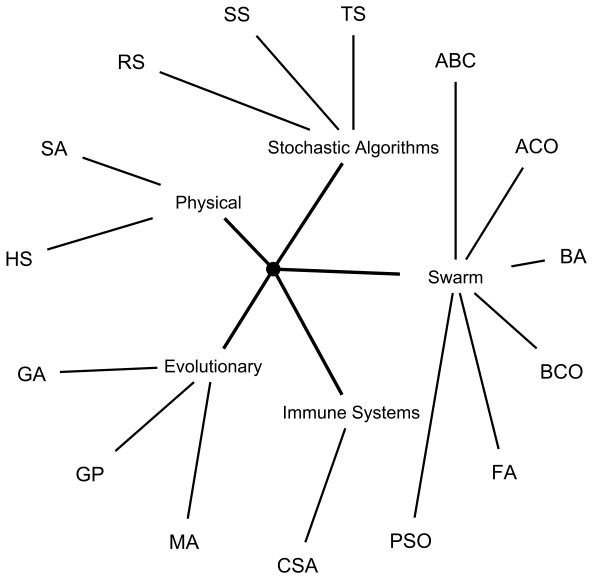

Fig. 1 Taxonomy of Nature Inspired Approaches

ACO-GA (Nemati et al, 2009), ACO-neural networks (Sivagaminathan and Ramakrishnan, 2007), PSO-catfish (Chuang et al, 2011), etc. Moreover, there also exist several approaches that have embedded local search procedures (Kabir et al, 2011; Oh et al, 2004).

2.3 Common Notions and Mechanisms

Table 1 Notions Used in Pseudo-codes

Notion	Meaning
$p^i \in P$	A Population P of individuals p^i
$B^{p^i} \in \mathbb{B}$	Set of candidate feature subsets B^{p^i} maintained by p^i
$\dot{B}$	Current best subset
$\tilde{B}$	A subset of randomly selected features
$b_j^{B^{p^i}} \in \{0,1\}$	Selection state (0: not selected, 1: selected) of the j^{th} feature in B^{p^i}
$f(B)$	Evaluation score of B
g	Current generation/iteration
$g_{\max}$	Maximum number of generations/iterations
r	A random number or a stochastic component
T	A temporary solution

Despite having distinctive characteristics and work flows, the stochastic-based search techniques share many similarities, which are summarised in Table 1. The population-based NIMs employ a group P of individuals p^i, each actively maintains an emerging feature subset

B^{p^i}. Internally, most algorithms represent a given feature subset B^{p^i} in a binary manner, where a string $b^{B^{p^i}}$ of length $|A|$ is used. The j^{th} position of $b^{B^{p^i}}$ is set to 1 ($b_j^{B^{p^i}} = 1$) if its corresponding feature is being selected, i.e., $a_j \in B^{p^i}$, and $b_j^{B^{p^i}} = 0$ if a_j is not selected in the candidate feature subset. The current best solution (amongst the entire population) is represented by $\dot{B}$, and a randomly generated feature subset is denoted by $\tilde{B}$. To simply the representation, random components other than $\tilde{B}$ are denoted by the use of r. For example, $c = r_c, 0 \leq r_c \leq 1$ indicates that the value of a certain parameter c is a randomly generated number drawn from the value range 0 to 1; and $a_r \in A, r \in \{1, \cdots, |A|\}$ denotes a feature randomly picked out of a pool of original features A. These notations will be used extensively in the pseudo codes hereafter, in order to illustrate the work flows of the reviewed NIMs in an unified representation that eases comparison.

2.3.1 Random Initialisation

One of the key advantages of nature inspired approaches is the insensitivity of initial states. The population at the start of the search (often referred to as the initial population) is generally a randomly generated pool. In stochastic FS, a random subset $\tilde{B}$ can be constructed by randomly setting r bits, where r itself may be a pre-determined size, or random: $r \in \{1, \cdots, |A|\}$.

$$\textbf{for } i = 1 \textbf{ to } |P| \textbf{ do } B^{p^i} = \tilde{B}$$

2.3.2 Solution Adjustment

The candidate solutions are modified constantly during the search process. The most common adjustment procedure is the random addition or removal of r_m number of features, such as the mutation operator used by many evolutionary algorithms. r_m may be pre-defined, or dynamically determined according to certain states of the algorithm. If a binary representation b^B is used, this adjustment may be achieved by randomly flipping r_m bits:

$$\textbf{for } i = 1 \textbf{ to } r_m \textbf{ do } b_r^B = \neg b_r^B,\ a_r \in A$$

Several swarm-based algorithms (Karaboga and Akay, 2009) exploit the notion of movement, from a given candidate solution B^{p^i} towards another possibly better quality feature subset, say B^{p^j}, aiming to eventually reach the true global best solution. For conventional numerical optimisation problems, this process derives new values for the function variables according to pre-defined formulae, and new solution vectors may be constructed which are interpolated in between the source and target vectors. However for FS problems, this is less applicable, since binary values are generally employed, and the variables represent independent features. In the literature, movement is implemented by first determining the distance between the two subsets:

$$d(B^{p^i}, B^{p^j}) = |B^{p^i} \oplus B^{p^j}| \tag{1}$$

which is equal to the number of bit differences. The amount of movement v, $v \in [0, v_{\max}]$, or the number of bits that B^{p^i} should copy from B^{p^j}, is then calculated with regards to the absolute distance, as demonstrated in Algorithm 1. Note that for algorithms such as PSO and FA, the feature subset being improved generally moves towards the current best solution $\dot{B}$.

if $v \leq d(B^{p^i}, B^{p^j})$ **then**
  for $j = 1$ **to** v **do**
    $b_r^{B^{p^i}} = \neg b_r^{B^{p^i}},\ a_r \in B^{p^i} \oplus B^{p^j}$
else
  $B^{p^i} = B^{p^j}$
  for $j = 1$ **to** $(v - d(B^{p^i}, B^{p^j}))$ **do**
    $b_r^{B^{p^i}} = \neg b_r^{B^{p^i}},\ a_r \notin B^{p^j}$

Algorithm 1: Move B^{p^i} towards B^{p^j} by a distance v

2.3.3 Subset Quality Comparison

FS is essentially a bi-objective optimisation task. A good quality feature subset should both achieve a high score in terms of evaluation (of its fit for purpose), and maintain a low cardinality. Having an ordered solution space enables higher quality solutions to be discovered. Algorithms such as CSA and HS use a scheme where two given candidate solutions are compared first based on evaluation scores, and the cardinalities of the subsets are then used as a tie breaker:

$$B^{p^i} > B^{p^j} \Leftrightarrow f(B^{p^i}) > f(B^{p^j}) \vee f(B^{p^i}) == f(B^{p^j}) \wedge |B^{p^i}| < |B^{p^j}| \tag{2}$$

Algorithms including FF, PSO, and SA require a single numerical difference between candidate solutions, so that the internal parameters can be calculated. In this case, evaluation score and subset size are integrated together via weighted aggregation, in order to reflect the influence of both the evaluation score $f(B)$ and subset size (normalised using $\frac{|B|}{|A|}$) of a given feature subset B. The weighting parameters α and β may be equal or biased:

$$B^{p^i} > B^{p^j} \Leftrightarrow \alpha f(B^{p^i}) + \beta \frac{|B^{p^i}|}{|A|} > \alpha f(B^{p^j}) + \beta \frac{|B^{p^j}|}{|A|} \tag{3}$$

Alternative aggregation methods may also be employed of course. Note that multi-objective evolutionary algorithms (Emmanouilidis et al, 2000; Freitas, 2008; Srinivasan and Ramakrishnan, 2011) have also been exploited to facilitate simultaneous optimisation of both criteria, but they are outside within the scope of this paper.

2.3.4 Current Best Solution Tracking

Due to the stochastic behaviour of the search algorithms concerned, it is often necessary to keep a record of the best quality feature subset $\dot{B}$ discovered so far, as the algorithm may explore other (possibly sub-optimal) solution regions later on. The procedure of updating $\dot{B}$ invokes the previously mentioned comparison process (Eqn. 2), and the quality of the current best solution is compared against those of all of the emerging subsets B^{p^i}, which are being maintained by the individuals $p^i \in P$ at every iteration:

```
for i = 1 to |P| do
    if B^{p^i} > Ḃ then Ḃ = B^{p^i}
```

Algorithm 2: Update current best solution $\dot{B}$

2.3.5 Local Search

Algorithm 3 details one of the local search procedures (Aha and Bankert, 1996), commonly used by techniques such as MA (Yang et al, 2008; Zhu and Ong, 2007) and hybrid search methods (Kabir et al, 2011; Oh et al, 2004). It is a greedy mechanism that evaluates all unselected features, and adds the most informative candidate (the feature that provides the most improvement in evaluation score) to the current feature subset. The hill climbing (HC) algorithm works in a similar fashion, which continues to select features until the score cannot be improved further. This mechanism can also be used backward to eliminate the least important feature from a subset.

```
repeat
    f' = f(B)
    t = −1
    for i = 1 to |A|, a_i ∉ B do
        b_i^B = 1
        if f(B) > f' then
            f' = f(B)
            t = i
        b_i^B = 0
    if t ≠ −1 then b_i^B = 1
until t == −1
```

Algorithm 3: Local search B

3 Nature Inspired FS Meta-heuristics

This section introduces the reviewed NIMs, organised by their underlying concepts. Pseudo-codes are given to better illustrate the procedures of these algorithms.

3.1 Biologically-Inspired Algorithms

3.1.1 Genetic Algorithm

Genetic algorithm (GA) mimics the process of natural evolution by simulating events such as inheritance, mutation, selection, and crossover. A considerable amount of investigation (Siedlecki and Sklansky, 1989; Sklansky and Vriesenga, 1996) has been carried out in order to explore the feasibility of applying GA to FS, much of this has been summarised and compared in the literature (Freitas, 2008). In GA, a feature subset is generally represented by a binary string called a chromosome. A population P of such chromosomes is randomly initialised and maintained, and those with higher fitness values are propagated into the later generations.

The reproduction process is typically achieved by the use of two operators: crossover and mutation. As shown in Algorithm 4, the standard one-point crossover operator exchanges and recombines a pair of parent chromosomes, B^p and B^q. It first locates a certain crossover point r_c along the length of the binary string, and then generates two children with all features beyond r_c swapped between the two parents. The mutation operator produces a modified subset by randomly adding or removing features from the original subset. By allowing the survival and reproduction of the fittest chromosomes, the algorithm effectively optimises the quality of the selected feature subset. The parameters c and m that control the rate of crossover and mutation require careful consideration, in order to allow the chromosomes to sufficiently explore the solution space, and to prevent premature convergence towards a local optimal subset.

$p^i \in P$, $i = 1$ **to** $|P|$ group of chromosomes
$B^{p^i} \in \mathbb{B}$, $i = 1$ **to** $|P|$ subsets associated with p^i
c, crossover rate
m, mutation rate
Randomly initialise P
for $g = 1$ **to** $g_{\max}$ **do**
    for $i = 1$ **to** $|P|$ **do**
        $T^i = B^{p^i}$ with a probability of $\frac{f(B^{p^i})}{\sum_{\forall B^{p^i} \in \mathbb{B}} f(B^{p^i})}$
        $T^{i+1} = B^{p^j}$ with a probability of $\frac{f(B^{p^j})}{\sum_{\forall B^{p^j} \in \mathbb{B}} f(B^{p^j})}$
        if $T^i == T^{i+1}$ **then**
            $b_r^{T^i} = \neg b_r^{T^i}, a_r \in A$
        else
            `// Crossover`
            if $r < c$ **then**
                $r_c \in \{1, ..., |A|/2\}$
                for $k = 1$ **to** r_c **do**
                    $T_k^{i+1} = B_k^{p^i}$, $T_k^i = B_k^{p^j}$
            `// Mutation`
            for $j = 1$ **to** $|A|$ **do**
                $0 \leq r_m \leq 1$
                if $r_m < m$ **then** $b_j^{T^i} = \neg b_j^{T^i}$
                if $r_m < m$ **then** $b_j^{T^{i+1}} = \neg b_j^{T^{i+1}}$
        $i = i + 2$
    for $i = 1$ **to** P **do** $B^{p^i} = T^i$
    Update $\dot{B}$

Algorithm 4: Genetic Algorithm

A GA-based FS algorithm is simple in concept, it may be implemented to achieve a great efficiency and obtain quality feature subsets. Being a randomised algorithm, however, there is no guarantee that a top quality feature subset (if not the global best solution) can be found in a reasonable or fixed amount of time. Its optimisation response time and solution quality are not constant. These drawbacks may limit GA's potential in the more demanding scenarios, such as on-line steaming FS (Wu et al, 2013). It is also challenging to identify a suitable set

of required parameter values, since the problem domain generally has very little linkage to the evolutionary concepts of GA.

3.1.2 Memetic Algorithm

Memetic algorithm (Hart et al, 2004; Ong et al, 2007) (MA) signifies several recent advances in evolutionary computation. It is commonly used to refer to any population-based evolutionary approach with a separate individual learning process (a local improvement procedure), and alternatively being referred to as hybrid GA, parallel GA, or genetic local search in the literature (Chen et al, 2011). When applied to the FS domain, the key research question is how the local search should be implemented. Such an approach typically follows a similar improvement process as Algorithm 3, except that the features being considered for addition are from a randomly selected subset, rather than from the complete set of the original features (Yang et al, 2008).

p^i, $i = 1$ **to** $|P|$ group of chromosomes
$B^{p^i} \in \mathbb{B}$, $i = 1$ **to** $|P|$ subsets associated with p^i
s, number of best individuals kept for reproduction
c, crossover rate
m, mutation rate
for $i = 1$ **to** $|P|$ **do**
    $B^{p^i} =$ Local search$(\tilde{B})$
for $g = 1$ **to** $g_{\max}$ **do**
    for $i = 1$ **to** s **do**
        $T^i = B^{p^i}$ with a probability of $\frac{f(B^{p^i})}{\sum_{\forall B^{p^i} \in \mathbb{B}} f(B^{p^i})}$
        $T^{i+1} = B^{p^j}$ with a probability of $\frac{f(B^{p^j})}{\sum_{\forall B^{p^j} \in \mathbb{B}} f(B^{p^j})}$
        if $T^i == T^{i+1}$ **then**
            $b_r^{T^i} = \neg b_r^{T^i}, a_r \in A$
        else
            if $r < c$ **then**
                $r_c \in \{1, ..., |A|/2\}$
                for $k = 1$ **to** r_c **do**
                    $T_k^{i+1} = B_k^{p^i}, T_k^i = B_k^{p^j}$
            for $j = 1$ **to** $|A|$ **do**
                $0 \leq r_m \leq 1$
                if $r_m < m$ **then** $b_j^{T^i} = \neg b_j^{T^i}$
                if $r_m < m$ **then** $b_j^{T^{i+1}} = \neg b_j^{T^{i+1}}$
        $i = i + 2$
    Sort $\mathbb{B}$
    for $i = 1$ **to** s **do** $B^{p^{|P|-i}} =$ Local search(T^i)
    Update $\dot{B}$

Algorithm 5: Memetic Algorithm

An alternative local improvement process (Zhu and Ong, 2007) suggests that a ranking of features should be computed first, and the solutions may then be improved by adding

or removing features based on the ranking information. Note that this requires the subset evaluator employed to be able to handle feature ranking, unless the information can be retrieved elsewhere. It has also been proposed that local search may be performed on an elite subset of population (Yusta, 2009), as shown in Algorithm 5, and the worse solutions may be substituted by the locally improved child solutions. The local search mechanism alters (adding or removing) one feature that provides the greatest increase in terms of evaluation score.

Although proved beneficial in a majority of scenarios, the presence of greedy mechanisms may have a negative impact on the quality of the feature subset returned, since the natural stochastic evolution of the chromosomes may be disrupted by excessive executions of local adjustments. This is because the variable values (i.e., features) are discrete rather than continuous. It is also difficult to identify the most suitable local search mechanism, in addition to the configuration of parameter values.

3.1.3 Clonal Selection Algorithm

Clonal selection algorithm (CSA) (de Castro and Von Zuben, 2002; Haktanirlar Ulutas and Kulturel-Konak, 2011) is inspired by the adaptive immune response behaviour to an antigenic stimulus. It exploits the fact that only the antibodies are selected to proliferate. The original algorithm involves a maturation process for the selected biological cells, which improves the affinity to the selective antibodies. A simplified implementation of CSA-based FS (Shojaie and Moradi, 2008) has been proposed. It enables both the selection of important features, and the optimisation of parameters for the end classifiers (which are implemented using the support vector machines). Although the original method integrates the two tasks together, it is easily modifiable to support generic FS, as shown in Algorithm 6. An adaptive CSA has also been adopted for network fault FS (Zhang et al, 2009).

$p^i \in P$, $i = 1$ **to** $|P|$ group of antibodies
$0 \leq f(B) \leq 1$, normalised subset evaluation score
c, maximum number of clones per antibody
m, maximum number of bits per mutation
s, maximum number of random cells
$T \in \mathbb{T}$, a set of temporary feature subsets
Randomly initialise P
for $g = 1$ **to** $g_{\max}$ **do**
    $\mathbb{T} = \emptyset$
    for $i = 1$ **to** $|P|$ **do**
        $c_i = ce^{f(B^{p^i}) - f(\dot{B})}$
        for $j = 1$ **to** c_i **do**
            $T = B^{p^i}$
            Flip $m(1 - c_i/c)$ random bits of T
            $\mathbb{T} = \mathbb{T} \cup \{T\}$
    for $i = 1$ **to** s **do** $\mathbb{T} = \mathbb{T} \cup \{\tilde{B}\}$
    Sort $\mathbb{T}$
    while $|\mathbb{T}| > |P|$ **do** $\mathbb{T} = \mathbb{T} \setminus \{T^{|\mathbb{T}|}\}$
    $\mathbb{B} = \mathbb{T}$
    Update $\dot{B}$

Algorithm 6: Clonal Selection Algorithm

The initial population is filled with randomly generated antibodies at start. At every iteration, clones are created for each individual. The better evaluation score an antibody achieves, the more clones are constructed. The maximum number of allowed clones can be configured by a parameter c, with an exponential function: $ce^{f(B^{p^i})-f(\dot{B})}$ used to calculate the amount of copies required. The clones are then mutated by flipping bits randomly, and added to the population. Again, the better the original antibody is, the less bit alteration will occur. The population is further joined by a set of antibodies which are randomly selected out of the existing population. The trim process then removes the worse antibodies in order to maintain the size of the group $|P|$. The current best solution is updated per iteration, and this process repeats until the maximum number of iterations is reached.

Despite being a much simplified version of CSA, the CSA-based FS technique still needs to produce clones of the good candidate feature subsets (antibodies), at every iteration. The exponential costs of generating, mutating, and evaluating such clones may become a significant overhead, especially for higher dimensional data sets.

3.2 Physical, Social, and Stochastic Algorithms

3.2.1 Harmony Search

Harmony search (HS) (Geem, 2010; Lee and Geem, 2005) is a recently developed meta-heuristic algorithm that mimics the improvisation process of musical players. In HS, a possible solution vector is referred to as a harmony H. A harmony memory $\mathbb{H}$ holds a selection of played harmonies, which can be more concretely represented by a two dimensional matrix. The number of rows (harmonies) are predefined and bounded by the size of harmony memory $|\mathbb{H}|$. Each column is dedicated to one musician. It stores the good notes previously played by the musician, and provides the pool of playable notes (referred to hereafter as the note domain N_i for p^i) for future improvisations.

Table 2 Harmony Encoded Feature Subsets

	p^1	p^2	p^3	p^4	p^5	p^6	Represented Subset B
H^1	a_2	a_1	a_3	a_4	a_7	a_{10}	$\{a_1, a_2, a_3, a_4, a_7, a_{10}\}$
H^2	a_2	a_2	a_2	a_3	a_9	a_-	$\{a_2, a_3, a_9\}$
H^3	a_2	a_-	a_2	$a_3 \rightarrow a_6$	a_9	a_4	$\{a_2, a_4, a_6, a_9\}$

When applied to FS (Diao and Shen, 2012), a musician is best described as an independent FS expert, and the available features translate to notes. Each musician may vote for one feature to be included in the emerging harmony (feature subset), indicating which features are being nominated. The set of the original features A forms the pool of notes shared by all musicians. Multiple musicians are allowed to choose the same feature, or they may opt to pick none at all. The fitness function naturally becomes a feature subset evaluator that analyses and merits each of the new subsets found during the search process. The HS-based FS described here employs a distinctive subset representation, which is different from any other approaches reviewed in this paper (but traditional binary string-based representation has also been examined (Diao and Shen, 2010)). Table 2 depicts the following three example harmonies. H^1 denotes a subset of 6 distinctive features. H^2 shows a duplication of choices

from the first three musicians, and a discarded note (represented by $a_{\_}$) from p^6, representing a reduced subset $B_2 = \{a_2, a_3, a_9\}$. In HS, a musician may sometimes select randomly from all available features (rather than from its note domain). This mutation-type of event is controlled by the harmony memory considering rate δ. H^3 signifies the feature subset $B_3 = \{a_2, a_6, a_4, a_9\}$, where $a_3 \rightarrow a_6$ indicates that p^4 originally voted for a_3, but it is forced to change its choice to a_6 due to δ activation.

$p^i \in P$, $i = 1$ **to** $|P|$, group of musicians
$H^j \in \mathbb{H}$, $j = 1$ **to** $|\mathbb{H}|$, harmony memory
$N_i = \bigcup_{j=1}^{|\mathbb{H}|} H_i^j$,he note domain of p^i
δ, harmony memory considering rate
T, a temporary set of feature subsets
for $g = 1$ **to** $g_{\max}$ **do**
    $T = \emptyset$
    for $i = 1$ **to** $|P|$ **do**
        if $r_\delta < \delta$ **then**
            $T = T \cup \{r_a\}$, $1 \leq r_a \leq |A|$
        else
            $T = T \cup \{N_{ir}\}$, $1 \leq r \leq |\mathbb{H}|$
    if $f(T) \geq \min\{f(H)\}, H \in \mathbb{H}$ **then**
        $\mathbb{H} = \mathbb{H} \cup \{T\}$
        $\mathbb{H} = \mathbb{H} \setminus \{\arg\min_{H \in \mathbb{H}} f(H)\}$

Algorithm 7: Harmony Search

The search process of HS is simple. At every iteration, each musician p^i picks one feature from their respective note domain N_i to form a new temporary subset T. The emerging solution replaces the worst harmony in the harmony memory if it achieves a higher evaluation score (or an equal score but with smaller cardinality), otherwise it gets discarded completely. The harmony memory considering rate δ enables a musician to pick a random feature similar to the mutation operator in GA, effectively encourages exploration.

Dynamic parameter tuning and iterative solution refinement techniques have been proposed (Diao and Shen, 2012), which aim to lessen the effort spent in parameter configuration, and to locate more compact subsets. A recent study has also investigated additional self-adjusting mechanisms for HS-based dynamic FS (Zheng et al, 2014), including: restricted feature domain, harmony memory consolidation, and pitch adjustment. Since HS imposes only limited mathematical requirements and is insensitive to initial value settings. Importantly, it has a novel stochastic derivative (for discrete problems such as FS) based on musician's experience, rather than gradient (for continuous variables) in differential calculus.

3.2.2 Simulated Annealing

Simulated annealing (SA) (Debuse and Rayward-Smith, 1997) is a generic probabilistic meta-heuristic for locating an approximation to the global optimum of a given complex function. It is inspired by the annealing process in metallurgy, a technique involving repeatedly heating and cooling a certain material in a controlled environment, in order to increase the size of the crystals, and reduce the defects, both of which depend on the thermodynamic free energy of the material.

```
g ∈ {g_min, g_max}, range of temperatures
ρ, perturbation percentage
c, cooling rate
e, equilibrium count
e_max, maximum number of successful perturbations
while g > g_min do
    B = Ḃ
    r_t = ⌈ρ|A|⌉
    Flip r_t random bits of B
    d = f(Ḃ) − f(B)
    if d ≤ 0 ∨ r < e^{−d/g} then
        Ḃ = B
        e = e + 1
    if e == e_max then
        e = 0, g = gc, ρ = ρc
```

Algorithm 8: Simulated Annealing

The adaptation of SA for FS (Ekbal et al, 2011; Meiri and Zahavi, 2006) requires an adjustment to the underlying computational algorithm, which ensures the SA keeping a record of the current best solution. Unlike most other population-based NIMs, SA-based FS maintains and improves only one single feature subset throughout the search process. As shown in Algorithm 8, the algorithms checks whether it has reached "thermal equilibrium" at a given energy state, by keeping a count e that increments each time SA finds a better quality feature subset. Once encountered sufficient improvements, SA makes a transition in the energy state, where both the temperature g, and the perturbation (mutation) percentage ρ are adjusted, by a so-called cooling rate c. The equilibrium count e is also reset. Such a transition generally indicates that a smaller number of features are adjusted during mutation, allowing fine tuning to be achieved towards termination.

The major criticism of SA: not able to always return the best solution found throughout the search process, is addressed in this application to FS. Only having to improve and maintain a single candidate solution has an obvious advantage of high efficiency. However, this also makes SA-based FS more prune to obtaining local minimal feature subsets, without careful consideration of the settings for the starting temperature and cooling rate.

3.2.3 Tabu Search

Tabu search (TS) is a local search-based meta-heuristic designed to avoid the pitfalls of typical greedy search procedures. It aims to investigate the solution space that would otherwise be left unexplored. Once a local optimum is reached, upward moves and those worsening the solutions are allowed. Simultaneously, the last moves are marked as tabu during the following iterations to avoid cycling.

A TS-based FS method (Hedar et al, 2008), as shown in Algorithm 9, has been proposed to deal with reduction problems in conjunction with the use of rough set theory. It employs a binary representation for the feature subsets. It also maintains a tabu list τ that holds a record of the most recently evaluated solutions, so that the algorithm can avoid being trapped in a previously explored region, and is refrained from generating solutions of very low quality. The tabu list is commonly initialised to contain two feature subsets: an empty subset $\emptyset$, and a set containing all available features A.

```
ℂ, candidate solutions ordered by quality
τ, tabu list
l, number of trials
T ∈ 𝕋, |𝕋| = l, neighbouring solutions
Q, number of occurrences of features in 𝕋
k ≤ k_max, number of iterations without improvements
ℂ = ∅, Ḃ = ∅, B = ∅, τ = {∅} ∪ {A}
for i = 1 to |A| do ℂ = ℂ ∪ {{a_1}}
while |B| < |A| do Local search (B)
for g = 1 to g_max do
    while k < k_max do
        for j = 1 to l do
            T = B
            Flip j random bits of T
            if T ∈ τ then
                j = j − 1
            else
                𝕋 = 𝕋 ∪ {T}
        B = arg max_{T∈𝕋} f(T)
        τ = τ ∪ {B}
        if f(B) > f(Ḃ) then
            // Shaking
            Ḃ = B
            for ∀C ∈ ℂ do
                if B ∩ C ∉ τ ∧ f(B ∩ C) ≥ f(Ḃ) then
                    Ḃ = B ∩ C
        else
            k = k + 1
        // Diversification
        for i = 1 to |A| do
            r_Q ∈ {1, ··· , |A|}
            if r_Q > Q_i then B = B ∪ {a_i}
```

Algorithm 9: Tabu Search

The approach starts by ranking the individual features according to their evaluation scores, and invokes a procedure to generate l new trial solutions that are neighbours to a given candidate solution, with a hamming distance of up to l features. The algorithm continues to generate new trial at each iteration, until no improvement has been observed for a predefined number of iterations. It then initiates two mechanisms to further "mutate" a given candidate solution: shaking and diversification. Shaking is essentially a greedy backward local search, each of the selected features are examined one by one, in order to check whether its removal produces a higher quality solution, or a subset of the same evaluation score with a reduced size. The diversification procedure attempts to generate a new candidate solution, which contains features chosen with probability inversely proportional to their number of appearances in the trial solutions. The process continues until the maximum number of iterations has been reached.

The greedy mechanisms employed by TS are very beneficial to quickly locating potentially better solutions, but they may lead to locally optimal feature subsets (see Section 4). TS also adopts a trial generation procedure similar to the cloning process of CSA, although the cost is not exponential, it may be a significant overhead for high dimensional problems.

3.3 Swarm Algorithms

3.3.1 Artificial Bee Colony

The artificial bee colony (Karaboga and Basturk, 2007) (ABC) algorithm is inspired by the intelligent behaviour of honey bee swarm when searching for promising food source. Because ABC is a relatively new algorithm, and much of its original concept has been modified and/or omitted in the existing FS adaptations (Palanisamy and S, 2012; Suguna and Thanushkodi, 2011), a brief description of the original method is given here first. In this algorithm, a colony of artificial bees is divided into three groups: employed bees, onlookers and scouts. The positions of food sources represent possible solutions to the optimisation problem. The nectar amount of a food source corresponds to the quality of its associated solution. The number of employed bees determines the amount of solutions to be simultaneously explored (and maintained).

After an initial, random distribution of the food source positions is generated, an employed bee attempts to locate a neighbouring food source and evaluates its nectar amount. If the quality of the nearby source is greater, the employed bee will point to the newer position, otherwise the previous food source is preserved. An onlooker watches the employed bees dance at the hive, sharing information of the discovered sources, and independently selects a food source to visit (following the same neighbourhood investigation procedure). The better food sources are recorded in place of the previously found locations. Employed bees abandon the unvisited food sources and become scouts, who perform random search for new solutions. This process repeat until a predefined set of requirements is met such as the maximum number of iterations.

$p^i \in P$, $i = 1$ **to** $|P|$, the group of employed bees
$q^i \in Q$, $i = 1$ **to** $|P|$, the group of onlooker bees
T^i, a temporary subset for p^i
for $g = 1$ **to** $g_{\max}$ **do**
    for $i = 1$ **to** $|P|$ **do**
        if $p^i.visited == false$ **then**
            $B^{p^i} = \dot{B}$
        else
            if $f(neighbour(B^{p^i})) > f(B^{p^i})$ **then**
                $B^{p^i} = neighbour(B^{p^i})$
    for $i = 1$ **to** $|P|$ **do**
        select p^j **with a probability of** $\frac{f(B^{p^j})}{\sum_{j=1}^{|P|} f(B^{p^j})}$
        $p^j.visited = true$
        if $f(neighbour(B^{p^j})) > f(B^{q^i})$ **then**
            $B^{q^i} = neighbour(B^{p^j})$
        else
            $B^{q^i} = B^{p^j}$
        Update $\dot{B}$

Algorithm 10: Artificial Bee Colony Optimisation

The rough set-based ABC FS method (Suguna and Thanushkodi, 2011) first groups the instances by the decision attributes, and applies a greedy local search to find the reduced feature sets (for each class). ABC is then used to choose a random number of features out of each set, and to combine the chosen features into the final feature subset. Due to the presence of the initial local search, this approach only requires a population size as big as the number of classes. A more general approach (Palanisamy and S, 2012) considers features as food sources. It configures the population of both employed bees and onlookers to be equal to the number of features. Each employed bee is allocated one feature at the start, and may be merged with others following the decisions of an onlooker, forming feature subsets in the process. The merge of B^{p^i} and B^{p^j} only happens if $r(f(B^{p^i}) - f(B^{p^j})) > 0$, $r \in [0,1]$.

In order to make the approach more scalable for data sets with a large number of features, an alternative method is described in Algorithm 10 that fits more closely to the original ABC algorithm. It uses a predefined population size independent of the number of features, which is initialised with randomly formed subsets $\tilde{B}$. Both the employed bees and onlookers employ the same neighbourhood investigation procedure, and accept a neighbouring solution if it is better than the subset it is currently examining. An onlooker q^i picks a particular employed bee p^j with probability:

$$prob_{B^{p^j}} = \frac{f(B^{p^j})}{\sum_{j=1}^{|P|} f(B^{p^j})} \tag{4}$$

which is in proportion to the evaluation score of its current feature subset $f(B^{p^j})$, and marks p^j as visited. At the end of the neighbourhood inspecting procedure, any employed bee that is unvisited generates and evaluates a new random subset $\tilde{B}$, as its current solution is very likely to be of a lower quality. The process repeats until $g_{\max}$ number of iterations is reached. The current best solution $\dot{B}$ which has been updated at every iteration, is returned as the final result. A solution adjustment procedure similar to the move operation, previously described in Algorithm 1, is also employed with promising results. In particular, it allows an onlooker to generate a neighbouring solution by moving its current subset towards that of the inspecting employed bee.

3.3.2 Ant Colony Optimisation

The ant colony optimisation (ACO) algorithm (Dorigo and Sttzle, 2010) is originally proposed for solving hard combinatorial optimisation problems. It is based on the behaviour of ants seeking an optimal path between their colony and a source of food. The approach uses a group of simple agents called ants that communicate indirectly via pheromone trails, and probabilistically constructs solutions to the problem being solved. Several adaptations of the ACO algorithm have been used in the FS problem domain, a number of which focus on rough set (Chen et al, 2010; Ke et al, 2008) and fuzzy-rough set-based (Jensen and Shen, 2005) subset evaluators, while a more general approach also exists in the literature (Kabir et al, 2012).

In ACO-based FS algorithms, features are represented as the nodes in a fully connected bi-directional graph, a candidate feature subset B is therefore a path that connects the selected features. Two sets of hints are available to the ants: the heuristic information η and the pheromone values τ. η is a pre-constructed matrix of size $|A|^2$, where A is the set of original features. cell $\eta_{jk} = \eta_{kj}$ stores the evaluation score of the feature subset $\{a_j, a_k\}$, and signifies the quality of the path between a_j and a_k. τ is another matrix of same size that stores the pheromone values deposited by the ants, it is initially filled with a constant value τ_0.

$p^i \in P$, $i = 1$ **to** $|P|$, group of ants
B^{p^i}, current edges (feature subset) traversed by p^i
$\eta_{jk} = \eta_{kj}$, $j,k = 1$ **to** $|A|$, heuristic information
$\tau_{jk} = \tau_{kj}$, $j,k = 1$ **to** $|A|$, pheromone values
τ_0, initial pheromone
ρ, evaporation rate
Initialise Parameters
for $j = 1$ **to** $|A| - 1$ **do**
    for $k = j + 1$ **to** $|A|$ **do**
        $\eta_{jk} = f(T)$
        $\tau_{jk} = \tau_0$
for $g = 1$ **to** $g_{\max}$ **do**
    for $j = 1$ **to** $|A| - 1$, $k = j + 1$ **to** $|A|$ **do**
        $\tau_{jk} = \rho\,\tau_{jk}$
    normalise τ
    for $i = 1$ **to** $|P|$ **do**
        $a_c = a_r, 1 \leq r \leq |A|$
        $B^{p^i} = \{a_c\}$
        while $|B^{p^i}| < |A|$ **do**
            select $a_l \notin B^{p^i}$ **with probability** ${\tau_{lc}}^{\alpha}{\eta_{lc}}^{\beta}$
            if $f(B^{p^i} \cup \{a_l\}) < f(B^{p^i})$ **break**
            $B^{p^i} = B^{p^i} \cup \{a_l\}$
            $\tau_{lc} = (1 - f(B^{p^i}))/2 + f(B^{p^i})\tau_{lc}$
            $a_c = a_l$
    for $i = 1$ **to** $|P|$ **do**
        for $j = 1$ **to** $|A| - 1$, $k = j + 1$ **to** $|A|$ **do**
            $\tau_{jk} = \tau_{jk} + f(B^{p^i})$
    Update $\dot{B}$

Algorithm 11: Ant Colony Optimisation

During every iteration, each ant beings from a random feature, an edge connecting the previous node a_c and an unvisited feature a_l is determined with probability:

$$prob_l = \frac{{\tau_{lc}}^{\alpha}{\eta_{lc}}^{\beta}}{\sum_{a_l \notin B} {\tau_{lc}}^{\alpha}{\eta_{lc}}^{\beta}} \tag{5}$$

where α and β are predefined parameters. Following the path construction process, an active (on-line) update of τ is performed, according to rules such as:

$$\tau_{jk} = m\tau_{jk} + n\frac{1}{|B|} \tag{6}$$

where m and n are predefined weights (Jensen and Shen, 2005; Ke et al, 2008). A passive (off-line) update (Jensen and Shen, 2005; Ke et al, 2008) may also be performed once the whole path (feature subset) B is established:

$$\tau_{jk} = \begin{cases} \rho\,\tau_{jk} + f(B), & j \in B \wedge k \in B \\ \rho\,\tau_{jk}, & \text{otherwise} \end{cases} \tag{7}$$

where ρ is the evaporation rate. For feature subset evaluators with a pre-determined, maximum evaluation score, e.g., rough and fuzzy-rough set-based evaluators that have a score range of

0 to 1, such information may be used to stop an ant from traversing towards further nodes, once the highest possible fitness value is obtained. For a generic evaluation technique, an ant may stop if $f(B) > f(\dot{B})$, or $f(B \cup \{a_l\}) < f(B)$, where l is the feature about to be included.

As ACO requires a pre-constructed heuristic information matrix, an $O(|A|^2)$ number of subset evaluations is necessary to calculate the pair-wise feature dependency, which may become prohibitive for large data sets or high complexity subset evaluators. To combat this, a starting set of essential features may be calculated in advance, such as the core for rough set and fuzzy-rough set-based techniques (Chen et al, 2010). This may significantly reduce this computational overhead, thereby eliminating the need to consider such features while traversing the graph. In addition, a normalisation process for τ has been proposed (Ke et al, 2008) to avoid search stagnation caused by extreme relative differences of pheromone trails.

3.3.3 Firefly Algorithm

The firefly algorithm (FA) (Yang, 2008) is a meta-heuristic inspired by the flashing behaviour of fireflies, which acts as a signal system to attract others. This approach has several underlying assumptions: 1) all fireflies are uni-sexual, so that an individual will be attracted to all others; 2) the brightness of a firefly is proportional to its fitness value but can decrease when observed over distance; and 3) a random exploration is performed if no brighter fireflies can be seen. It has been shown that FA degenerates into the particle swarm optimisation algorithm with specific parameter settings.

$p^i \in P$, $i = 1$ **to** $|P|$, group of fireflies
$q^i \in Q$, $i = 1$ **to** $|Q|$, $|Q| = |P|$, temporary group of fireflies
γ, light absorption coefficient
I_{ij}, observed brightness of p^j by p^i
Initialise Parameters
Random Initialisation
for $g = 1$ **to** $g_{\max}$ **do**
    $Q = \emptyset$
    for $i = 1$ **to** $|P|$ **do**
        select p^j, $j = \arg\max_j I_{ij}$, $j \neq i$
        $d_{ij} = d(B^{p^i}, B^{p^j})$
        $p^{i'} =$ **move** p^i **towards** p^j **by** $d_{ij} e^{-\gamma {d_{ij}}^2}$
        if $f(B^{p^{i'}}) > f(B^{p^i})$ **then** $B^{q^i} = B^{p^{i'}}$
    $P = Q$
    Update $\dot{B}$

Algorithm 12: Firefly Algorithm

FA has been successfully applied to addressing rough set-based FS problems (Banati and Bajaj, 2011), via the use of a population size equal to the number of features. Each individual's feature subset is initialised with one of the original features: $B^{p^i} = \{a_i\}$. The brightness I_i of p^i is determined by the rough set dependency score of its associated feature only. The best mating partner p^j for a firefly p^i should satisfy: 1) $I_j > I_i$, 2) the distance $f^* - f(B^{p^i} \cup B^{p^j})$ is minimal for all $p^j \in P, j \neq i$, and 3) $f(B^{p^i} \cup B^{p^j}) > f(B^{p^j})$. The two subsets then merge together and the process repeats for all fireflies until the maximum rough set dependency score is achieved. This implementation removes all stochastic components

from the base FA algorithm, and delivers compact rough set reducts in a manner similar to that of a greedy local search.

An alternative FA-based FS approach can be developed. Briefly, it makes better use of the stochastic elements proposed by the original, base FA algorithm. As illustrated in Algorithm 12, such an approach supports the use of any subset-based evaluator, and shares similar intentions and modifications as those of the improved ABC algorithm explained in 3.3.1. A population P of pre-defined fireflies p^i are initialised with random subsets. The brightness of p^i when observed by p^j is calculated using:

$$I_{ij} = f(B^{p^j}) \cdot e^{-\gamma d(B^{p^i}, B^{p^j})^2} \tag{8}$$

where γ is a pre-defined parameter termed the "absorption coefficient". This implements the original idea in which the attractiveness of a firefly decreases as the distance between it and its mating partner increases. In FS terms, this means that the larger $|B^{p^i} \oplus B^{p^j}|$ (the total number of mismatched features) is, the less bright it will be perceived. A subset B^{p^i} is moved towards its best mating partner with a distance of:

$$d_{ij} = d(B^{p^i}, B^{p^j}) \cdot e^{-\gamma d(B^{p^i}, B^{p^j})^2} \tag{9}$$

The resulting subset $B^{p^{i'}}$ replaces the previous B^{p^i} if it is a better solution, otherwise the original subset is maintained. At every iteration, the current best solution $\dot{B}$ is updated and returned once the process reaches maximum number of iterations.

3.3.4 Particle Swarm Optimisation

Particle swarm optimisation (PSO) (AlRashidi and El-Hawary, 2009) is a method that optimises a problem by exploiting a population of particles P, also referred to as the swarm, which move around in the solution space with simulated positions and velocities. The movement of a given particle is not only influenced by its own current best position, but also guided towards the currently known group-wise best position in the search space. The individual current best solution is constantly updated when the individual particles locate better positions. The previously introduced FA is very closely related to PSO, having many similar underlying principles, especially with regards to the concept of particle movements. However, the fireflies in FA are only attracted and move towards locally observed best mating partners.

When applied to FS, the velocity v_i which represents the number of features to be altered, of a given candidate feature subset B^{p^i} is calculated by:

$$v_i = w_g v_i + c_1 r_{d_1} d(\dot{B}^{p^i}, B^{p^i}) + c_2 r_{d_2} d(\dot{B}^{p^i}, B^{p^i}) \tag{10}$$

where w_g is a gradually decreasing inertia weight, and c_1 and c_2 are the acceleration constants giving weights to the current individual best and the group-wise best solution, respectively. The outcome of the velocity calculation, is further randomised via the use of random numbers $0 \leq r_{d_1}, r_{d_2} \leq 1$. It has been suggested in the literature (Wang et al, 2007) that the velocity should be regulated by a pre-defined value $v_{\max}$, since the amount of features being modified can potentially become very large. Finally, once the number of features to be modified is determined, the new candidate subset is calculated following Algorithm 1.

There exists significant debate surrounding the velocity calculation (Chuang et al, 2011; Liu et al, 2011; Wang et al, 2007). This reflects the discrepancy between the intended usage

$p^i \in P$, $i = 1$ **to** $|P|$, group of particles
$\dot{B}^{p^i}$, local best subset found by p^i.
c_1, c_2, acceleration constants towards $\dot{B}^{p^i}$ and $\dot{B}$
$w_g \in [w_{\min}, w_{\max}]$, gradually decreasing inertia weight
$v_i \in [1, v_{\max}]$, current and the maximum velocity
Initialise Parameters
Random Initialisation
for $g = 1$ **to** $g_{\max}$ **do**
    Update $\dot{B}$, $\dot{B}^{p^i}$
    for $i = 1$ **to** P **do**
        $0 \leq r_{d_1}, r_{d_2} \leq 1$
        $v_i = w_g v_i + c_1 r_{d_1} d(\dot{B}^{p^i}, B^{p^i}) + c_2 r_{d_2} d(\dot{B}^{p^i}, B^{p^i})$
        Move B^{p^i} towards $\dot{B}^{p^i}$ by v_i
    $w_g = w_{\min} + (1 - g/g_{\max})(w_{\max} - w_{\min})$

Algorithm 13: Particle Swarm Optimisation

of "movement" proposed in the base PSO algorithm, and its actual implementation in PSO-based FS. For continuous-valued optimisation, the notions such as velocity and movement are intuitive to understand, which is used to locate a possible intermediate, interpreted solution between two solution vectors (points) in a continuous space. Yet in FS, the features are discrete-valued, and the current binary representation does not allow straightforward, meaningful interpretation between two feature subsets. Therefore, PSO- and FA-based FS methods may potentially benefit from the integer-valued representation as used in HS-based FS.

4 Experimentation

A series of systematic tests are carried out using a selection of feature subset evaluators, including CFS (Hall, 1998), PCFS (Dash and Liu, 2003), and FRFS (Jensen and Shen, 2009b), which differ in terms of computational complexity and characteristics. For instance, CFS is the most lightweight method. It addresses the problem of FS through a correlation-based approach, and identifies features that are highly correlated with the class, yet uncorrelated with each other (Hall, 1998). PCFS is an FS approach that attempts to identify a group of features that are inconsistent, and removes irrelevant features in the process (Dash and Liu, 2003). FRFS, similar to most rough set-based methods, exploits rough set notions such as the lower and upper approximations of a given concept, and is able to identify very compact subsets of features that can fully discern the training objects into their respective classes. Note that FRFS is relatively high in terms of computational complexity, and finding the minimal sized solution of full discernibility (a minimal fuzzy-rough reduct (Jensen and Shen, 2009b)) remains as significant research.

In total 12 real-valued UCI benchmark data sets (Frank and Asuncion, 2010) are used, in order to demonstrate the capabilities of the reviewed approaches. Several data sets are high in dimension and hence, present reasonable challenges to FS. Lower dimensional problems (e.g., *cleveland* and *heart*) are also included to see whether the feature subsets selected by the algorithm are consistent. Table 3 provides a summary of these data sets.

Stratified 10-fold cross-validation (10-FCV) is employed for data validation, where a given data set is partitioned into 10 subsets. Of these 10 subsets, nine are used to form one

Table 3 Data Set Information

Data Set	Feature	Instance	Class	C4.5 (%)	NB (%)
arrhythmia	280	452	16	65.97	61.40
cleveland	14	297	5	51.89	55.36
handwritten	257	1593	10	75.74	86.21
heart	14	270	2	77.56	84.00
ionosphere	35	230	2	86.22	83.57
libras	91	360	15	68.24	63.635
multifeat	650	2000	10	94.54	95.30
ozone	73	2534	2	92.70	67.66
secom	591	1567	2	89.56	30.04
sonar	61	208	2	73.59	67.85
water	39	390	3	81.08	85.40
waveform	41	699	2	75.49	79.99

training fold. The FS methods are employed to identify quality subsets, which are then used to build classification models. A single subset is retained as the testing data, so that the built classifiers can be compared using the same unseen data. This process is then repeated 10 times (the number of folds). The advantage of 10-FCV over random sub-sampling is that all objects are used for both training and testing, and each object is tested only once per fold. The stratification of the data prior to its division into different folds ensures that each class label has equal representation in all folds, thereby helping to alleviate bias/variance problems (Bengio and Grandvalet, 2004). In the experiment, unless stated otherwise, 10-FCV is executed 10 times (10×10-FCV) in order to reveal the impact of the stochastic nature of the approaches employed. The differences in performance of the various methods are statistically compared using Paired T-Test with two-tailed $P = 0.01$.

Note that since 10×10-FCV is imposed, each of the figures displayed in the following tables is an averaged result of 100 search outputs (per data set per algorithm). Obviously, the searches are carried out using the same fold of the data set each time, so that their results (and the final averaged figures) are directly comparable.

4.1 Feature Selection Results

Tables 4 details the results collected with three different subset evaluators and all of the reviewed search algorithms. As the time complexity of a subset evaluation using CFS is very low, the maximum search iterations/generations for this set of experiment is set to a very large value ($g_{\max}$ = 50,000), in order to allow all algorithms to fully converge. Based on the figures shown in this table, GA, HS, and MA deliver very similar results, and work well for lower dimensional data sets such as *cleveland*, *heart*, *ionosphere*, and *water*. Algorithms such as SA and TS demonstrate very good performance for the most complex data sets: *arrhythmia*, *multifeat*, and *secom*. ABC, ACO, and FF are not particularly competitive in identifying feature subsets with the highest evaluation scores. However, they are relatively good at producing very compact feature subsets, with acceptable evaluation quality.

For results obtained with PCFS, TS fails to find subsets with the best evaluation scores for most of the data sets, which forms a sharp contrast to its strong performance in the previous set of experiment. However, it still identifies the best solutions for *multifeat* and *waveform*, which are two of the higher dimensional problems. CSA, GA, and HS demonstrate their capabilities in finding good quality and compact feature subsets for 7/12 data sets. GA is the

Table 4 Feature selection results using CFS, PCFS, and FRFS, showing both the evaluation scores (left) and the sizes (right) of the selected feature subsets. Bold figures indicate the highest evaluation scores, shaded cells signify overall most compact best solutions.

CFS

Data Set	ABC		ACO		CSA		FF		GA		HS		MA		PSO		SA		TS	
arrhythmia	0.349	10.5	0.369	15.5	**0.467**	**24.6**	0.394	27.7	0.263	61.4	**0.466**	**26.8**	0.275	58.4	0.280	16.4	**0.466**	**23.5**	0.467	25.1
cleveland	0.271	5.4	0.239	6.4	0.271	5.4	0.269	5.0	**0.274**	**6.7**	**0.274**	**6.7**	**0.274**	**6.7**	**0.274**	**6.7**	0.266	4.4	**0.274**	**6.6**
handwritten	0.435	92.0	0.444	31.5	**0.527**	**78.0**	0.484	58.0	0.513	97.9	0.525	93.7	0.467	120	0.473	122	**0.526**	**68.2**	**0.527**	**82.3**
heart	0.337	5.3	0.313	6.3	0.337	5.3	0.334	5.0	**0.338**	**6.4**	**0.338**	**6.4**	**0.338**	**6.4**	**0.338**	**6.4**	0.333	4.7	0.335	5.3
ionosphere	0.523	9.6	0.511	9.5	0.538	9.4	0.528	9.2	**0.539**	**10.4**	**0.539**	**10.3**	**0.539**	**10.4**	0.535	10.9	0.536	8.4	0.537	9.3
libras	0.577	23.5	0.570	15.6	0.610	23.0	0.589	21.7	**0.611**	**31.4**	**0.611**	**28.7**	0.607	33.7	0.595	26.6	0.607	19.1	**0.611**	**25.0**
multifeat	0.824	235	0.802	26.9	0.923	106	0.836	88.4	0.879	258	0.919	141	0.836	318	0.849	357	**0.926**	**76.0**	**0.926**	**80.0**
ozone	0.100	14.0	0.102	10.7	0.112	14.0	0.106	13.0	**0.114**	**21.9**	**0.114**	**21.4**	**0.113**	**23.3**	0.110	25.3	0.107	9.5	**0.114**	**19.8**
secom	0.024	2.9	0.085	14.2	**0.101**	**14.5**	0.045	15.7	0.008	97.0	0.064	22.5	0.025	95.6	0.018	20.8	**0.101**	**14.1**	**0.100**	**14.3**
sonar	0.330	11.4	0.316	17.7	**0.360**	**16.7**	0.339	12.5	**0.360**	**17.7**	**0.360**	**17.7**	**0.360**	**17.6**	0.329	11.9	0.358	15.4	0.359	16.6
water	0.417	8.2	0.387	10.3	**0.426**	**10.4**	0.416	7.5	**0.426**	**10.5**	**0.426**	**10.5**	**0.426**	**10.5**	0.417	8.9	0.423	8.4	0.424	9.2
waveform	0.366	11.3	0.361	14.5	0.383	13.1	0.364	10.5	**0.384**	**14.9**	**0.384**	**14.9**	**0.384**	**14.9**	0.372	12.4	0.382	12.6	**0.384**	**14.9**

PCFS

Data Set	ABC		ACO		CSA		FF		GA		HS		MA		PSO		SA		TS	
arrhythmia	0.987	121	0.752	8.8	0.988	107	0.977	38.7	**0.989**	**45.6**	**0.989**	**29.1**	**0.989**	**114**	**0.989**	**111**	**0.989**	**29.1**	0.983	21.6
cleveland	**0.781**	**8.0**	0.775	8.8	**0.781**	**7.9**	0.715	6.4	**0.781**	**7.9**	**0.781**	**7.9**	**0.781**	**7.9**	**0.781**	**7.9**	**0.781**	**8.0**	0.738	6.7
handwritten	**1.000**	**27.1**	**1.000**	**26.9**	**1.000**	**22.0**	**1.000**	**23.1**	**1.000**	**41.2**	**1.000**	**70.2**	**1.000**	**40.5**	**1.000**	**24.0**	**1.000**	**83.4**	0.999	18.1
heart	**0.961**	**9.5**	0.955	10.2	**0.961**	**9.5**	0.915	7.2	**0.961**	**9.5**	**0.961**	**9.5**	**0.961**	**9.5**	**0.961**	**9.5**	0.922	8.7	0.947	8.3
ionosphere	**0.996**	**9.8**	0.993	10.0	**0.996**	**7.0**	**0.996**	**8.7**	**0.996**	**10.0**	**0.996**	**6.8**	**0.996**	**10.1**	**0.996**	**8.1**	0.989	15.6	0.991	6.4
libras	**0.971**	**35.2**	0.935	19.0	**0.972**	**17.2**	0.968	24.5	**0.972**	**18.2**	**0.972**	**16.4**	**0.972**	**33.3**	**0.972**	**28.4**	0.961	41.6	0.967	16.0
multifeat	**1.000**	**14.6**	**1.000**	**10.6**	**1.000**	**13.1**	**1.000**	**19.5**	**1.000**	**44.9**	**1.000**	**9.1**	**1.000**	**43.8**	**1.000**	**13.4**	**1.000**	**325**	**1.000**	**6.1**
ozone	0.997	23.0	0.969	14.2	0.999	16.6	0.996	20.1	**1.000**	**21.0**	0.999	18.4	**1.000**	**31.8**	**1.000**	**26.0**	0.994	34.5	0.999	19.1
secom	0.986	211	0.936	2.5	**0.989**	**213**	0.974	150	**0.990**	**198**	0.988	23.7	0.988	314	**0.990**	**256**	0.971	294	0.979	97.3
sonar	**0.993**	**24.8**	0.946	11.6	**0.993**	**11.7**	0.989	15.8	**0.993**	**12.4**	0.991	11.8	**0.993**	**23.7**	**0.993**	**17.6**	0.924	30.3	0.985	11.6
water	**0.994**	**15.8**	0.975	10.8	**0.995**	**9.8**	0.990	10.5	**0.995**	**10.0**	**0.995**	**10.2**	**0.995**	**13.8**	**0.995**	**12.2**	0.927	18.9	0.990	9.3
waveform	0.999	12.4	0.999	11.4	0.999	9.7	0.999	10.9	**1.000**	**11.5**	**1.000**	**11.1**	**1.000**	**14.9**	**1.000**	**12.9**	0.979	20.1	**1.000**	**10.6**

FRFS

Data Set	ABC		ACO		CSA		FF		GA		HS		MA		PSO		SA		TS	
arrhythmia	1.000	40.4	1.000	23.6	1.000	29.0	1.000	29.2	1.000	63.4	1.000	25.1	1.000	63.1	1.000	34.5	1.000	108	1.000	24.9
cleveland	0.929	13.0	0.929	13.0	0.929	13.0	0.854	10.9	0.999	12.9	0.999	12.9	0.999	12.9	0.999	12.9	0.929	11.2	0.989	11.9
handwritten	1.000	26.6	1.000	27.5	1.000	22.0	1.000	22.9	1.000	40.4	0.999	22.5	1.000	40.0	1.000	23.8	1.000	129	0.999	22.0
heart	0.959	12.8	0.960	13.0	0.959	12.8	0.909	10.7	1.000	10.1	1.000	10.0	1.000	10.1	1.000	10.3	0.989	10.6	0.959	9.1
ionosphere	0.993	15.3	0.994	16.6	0.993	14.0	0.992	15.1	1.000	25.7	1.000	25.7	1.000	26.1	1.000	26.5	0.991	14.9	1.000	25.9
libras	0.997	19.7	0.997	21.3	0.998	18.7	0.997	19.5	1.000	29.4	1.000	20.7	1.000	29.9	1.000	23.2	0.999	26.2	0.999	20.4
multifeat	1.000	20.3	1.000	17.6	1.000	18.9	1.000	27.3	1.000	49.8	1.000	15.3	1.000	41.8	1.000	21.0	1.000	323	0.562	6.0
ozone	0.975	36.5	0.962	35.7	0.973	32.9	0.976	34.5	0.982	48.5	0.924	38.5	0.982	48.8	0.982	51.9	0.919	36.1	0.979	33.7
secom	1.000	35.2	1.000	20.5	1.000	37.0	1.000	26.6	1.000	67.5	1.000	15.8	1.000	67.1	1.000	31.7	1.000	296	0.803	7.0
sonar	1.000	13.3	1.000	14.5	1.000	12.9	1.000	13.5	1.000	17.3	1.000	13.0	1.000	16.9	1.000	14.1	1.000	17.8	0.998	11.7
water	0.998	18.0	0.998	19.0	0.998	15.9	0.997	17.8	1.000	20.8	1.000	19.8	1.000	22.0	1.000	22.1	0.981	19.2	1.000	19.9
waveform	1.000	17.0	1.000	18.0	0.999	16.5	1.000	17.0	1.000	19.2	1.000	18.4	1.000	19.4	1.000	19.2	0.996	20.5	1.000	17.4

only algorithm that identifies the overall best solutions for the *ozone* and *secom* data sets. Note that the search outputs from these algorithms can differ significantly. Taking the *secom* data set as an example, the overall best evaluation score is 0.990 (achieved by CSA, GA, and PSO) with around 200 features, while ACO, HS, and TS yield subsets with an average size of only 2.5, 23.7, and 97.3, respectively. The evaluation scores of the resultant solutions are also significantly lower in comparison, indicating that only local minimal solutions are possibly returned. Similar observations are also reflected by the results for the *arrhythmia* data set.

FRFS is a computationally intensive evaluator that requires a considerable amount of time when the number of training objects is very large. Because of the underlying mathematical properties of rough and fuzzy-rough sets (Jensen and Shen, 2009b; Mac Parthaláin et al, 2010b), it is easy to find feature subsets with almost full dependency scores (for the commonly adopted fuzzy t-norms and fuzzy implicators). However, the search for the most compact solutions (fuzzy-rough reducts) is very challenging. These characteristics of FRFS help to compare the size reduction capabilities of the reviewed methods. Note that the evaluation scores are not compared statistically, since subsets with full dependency score can be readily identified. HS performs very well in this set of experiment, mainly owning to the fact that it is tailored to solving FRFS problems in the first place (Diao and Shen, 2010), and that it

Table 5 C4.5 (left) and NB (right) classification accuracies using the feature subsets found with the respective search algorithms via CFS, PCFS, and FRFS. Bold figures indicate best accuracy (per classifier), shaded cells signify higher accuracies are achieved by both classifiers.

CFS

Data Set	ABC		ACO		CSA		FF		GA		HS		MA		PSO		SA		TS	
arrhythmia	63.0	63.2	66.8	67.0	67.1	68.5	66.8	67.0	66.8	66.8	66.9	**68.9**	66.9	67.4	63.5	63.3	**67.4**	**69.0**	67.2	**69.0**
cleveland	55.8	56.3	55.3	56.5	55.8	56.3	55.8	56.6	**56.4**	**56.9**	**56.4**	**56.9**	**56.4**	**56.9**	**56.4**	**56.9**	55.0	55.7	**56.3**	**57.0**
handwritten	75.0	83.7	70.2	72.9	**75.9**	**84.7**	74.3	82.0	75.5	**85.2**	**75.9**	**85.3**	75.4	**85.5**	75.3	**85.3**	**76.0**	83.8	**76.1**	**84.9**
heart	**81.7**	83.0	81.2	84.1	**81.7**	83.0	**82.2**	83.0	80.6	**85.0**	80.6	**85.0**	80.6	**85.0**	80.7	**85.0**	**82.1**	82.3	**81.7**	**82.9**
ionosphere	**88.2**	85.6	87.6	86.3	87.8	**86.8**	**88.0**	86.1	**88.4**	**86.9**	**88.3**	**86.9**	**88.3**	**86.9**	**88.0**	86.4	87.4	**87.0**	87.3	**87.0**
libras	65.6	60.7	62.1	57.3	67.0	61.6	65.5	61.0	67.6	**62.1**	67.3	61.6	**68.2**	**61.8**	66.8	61.4	65.9	61.4	66.9	61.3
multifeat	94.3	95.7	93.4	95.2	**94.9**	**97.1**	92.8	95.7	**94.6**	96.3	**94.9**	96.8	**94.6**	95.8	**94.7**	95.9	**95.1**	**97.2**	**94.9**	**97.2**
ozone	**93.4**	75.8	**93.4**	77.2	93.1	74.8	**93.3**	76.1	**93.3**	73.9	93.2	73.9	**93.4**	73.7	**93.3**	73.5	**93.4**	**78.4**	93.1	74.0
secom	**93.4**	**88.7**	92.5	82.6	92.5	84.2	92.7	75.1	90.7	84.1	92.1	71.2	91.2	**88.7**	92.7	74.2	92.5	84.7	92.4	83.7
sonar	72.3	66.6	73.2	66.3	73.0	66.6	73.4	65.9	73.1	66.6	73.2	66.6	73.3	66.5	72.8	66.3	72.3	66.6	**74.1**	**66.9**
water	81.9	84.9	82.7	**85.8**	82.8	**85.9**	82.3	85.1	**83.4**	**85.9**	83.3	**85.9**	**83.3**	**85.9**	82.1	85.2	83.0	85.6	82.9	85.6
waveform	76.9	79.8	**77.4**	**80.5**	**77.6**	**80.7**	76.8	79.5	**77.5**	80.2	**77.5**	80.2	**77.5**	80.2	76.9	79.7	**77.6**	**80.9**	**77.5**	80.5

PCFS

Data Set	ABC		ACO		CSA		FF		GA		HS		MA		PSO		SA		TS	
arrhythmia	66.4	63.3	62.6	61.5	66.3	63.5	**66.7**	66.2	66.2	65.9	66.2	66.8	66.3	64.0	66.1	63.9	66.2	67.5	**66.7**	**68.2**
cleveland	54.5	**55.9**	54.0	**56.0**	54.5	**55.9**	**56.4**	**55.8**	54.5	**55.9**	54.5	**55.9**	54.5	**55.8**	54.4	**55.9**	54.5	**55.8**	55.2	**56.0**
handwritten	65.2	68.4	65.6	69.1	63.2	66.2	63.8	66.7	68.1	73.3	69.9	75.9	67.9	72.8	63.7	66.6	70.2	**76.8**	**71.1**	73.5
heart	78.5	**84.4**	78.6	84.2	78.5	**84.4**	**80.2**	83.0	78.5	**84.4**	78.5	**84.4**	78.5	**84.4**	78.5	**84.4**	79.6	83.1	79.5	**84.5**
ionosphere	85.8	79.6	86.0	80.5	85.7	78.8	86.0	79.9	85.4	79.0	85.7	80.1	85.3	80.1	85.5	78.6	85.5	**81.0**	**87.4**	80.3
libras	**66.0**	**61.9**	64.5	58.7	64.4	60.2	65.9	**61.8**	64.1	60.4	65.4	61.5	65.1	61.6	65.3	61.6	**66.1**	61.5	63.7	59.9
multifeat	79.5	82.1	83.8	86.5	78.7	81.0	80.2	82.7	86.0	89.5	81.7	84.3	85.5	89.1	78.5	81.0	**93.4**	**94.9**	89.8	91.4
ozone	93.0	72.5	93.1	75.7	**93.3**	**76.3**	93.1	73.7	93.1	73.4	93.2	75.6	93.1	70.6	93.0	71.8	92.9	69.4	93.2	75.2
secom	90.4	36.2	**93.3**	**90.9**	90.7	37.8	91.4	49.6	90.5	37.0	92.5	70.9	90.4	32.3	90.2	35.0	89.9	31.9	92.1	67.6
sonar	73.5	67.1	73.2	66.5	**74.1**	66.3	**73.9**	66.7	72.4	66.6	73.5	66.6	73.0	66.3	73.7	66.6	72.2	66.9	**73.8**	**67.3**
water	**81.5**	85.7	**81.4**	84.9	81.1	86.3	**81.5**	85.9	**81.7**	85.9	**81.4**	**86.2**	81.7	86.1	**81.7**	**86.3**	80.2	83.2	**81.9**	86.1
waveform	74.6	78.5	75.3	80.0	73.7	78.9	75.0	78.6	75.0	79.1	74.7	79.5	74.3	78.8	74.8	79.1	71.6	75.8	**75.8**	**80.2**

FRFS

Data Set	ABC		ACO		CSA		FF		GA		HS		MA		PSO		SA		TS	
arrhythmia	57.8	58.5	57.1	58.5	58.7	**61.8**	52.3	57.3	58.0	57.2	60.6	60.6	**63.0**	57.2	56.6	58.2	62.1	57.0	59.5	61.5
cleveland	50.4	55.1	50.4	55.1	50.4	55.1	49.9	54.1	49.7	55.1	49.7	55.1	49.7	55.1	49.7	55.1	**55.8**	**56.0**	51.8	53.8
handwritten	65.6	68.2	66.7	69.0	62.0	65.1	63.5	66.1	68.3	71.2	64.3	67.0	67.4	72.0	62.2	64.1	**73.9**	**83.3**	65.7	67.9
heart	77.8	82.6	77.8	**83.0**	77.8	82.6	78.9	**83.0**	78.0	82.4	78.9	82.4	78.5	82.4	78.3	81.7	**79.3**	82.2	78.2	80.7
ionosphere	88.5	83.3	**89.8**	**85.0**	89.1	83.9	88.3	83.7	88.7	83.9	88.5	83.9	88.5	83.7	88.5	83.9	85.7	82.6	86.5	83.9
libras	63.6	56.1	60.8	58.5	59.9	54.6	63.1	58.6	62.9	58.8	63.1	53.6	**66.0**	**60.3**	64.4	57.2	62.1	58.8	63.3	58.9
multifeat	80.6	84.4	83.1	88.3	82.6	86.4	81.8	86.6	87.0	91.1	84.1	88.6	85.8	90.2	83.2	86.7	**93.8**	**95.2**	84.8	86.0
ozone	93.3	**69.9**	93.0	67.6	92.7	**70.3**	93.0	69.4	93.2	67.0	93.1	**69.7**	93.2	68.3	93.1	67.3	92.8	68.5	**93.5**	**69.6**
secom	92.7	71.6	92.9	91.6	92.7	77.4	93.2	69.4	91.6	50.4	93.1	93.1	91.4	43.9	92.7	74.6	90.2	31.3	**93.4**	**93.4**
sonar	75.3	69.2	76.0	73.9	76.2	71.1	71.6	71.5	71.7	67.9	**76.7**	71.2	70.0	67.4	70.7	65.4	70.8	68.4	76.4	**76.4**
water	80.8	83.9	80.0	83.9	78.7	**85.0**	79.7	83.9	79.7	83.9	80.6	84.4	**81.4**	**85.0**	81.3	83.9	79.6	84.2	79.7	84.6
waveform	71.1	74.8	**73.8**	**78.5**	71.0	74.9	70.4	75.8	70.4	75.7	72.9	75.7	71.6	75.6	73.0	76.5	71.8	75.5	72.6	75.9

also embeds mechanisms to actively refine the sizes of the feature subsets during the search. GA, MA, and PSO also deliver competitive performance. Although TS obtains best results for 6/12 data sets, it fails to optimise the FRFS dependency scores for *multifeat* and *secom*, producing sub-optimal solutions.

4.2 Classification Results

The classification algorithms adopted in the experiments include two commonly used techniques: 1) the tree-based C4.5 algorithm (Witten and Frank, 2005) that uses entropy to identify the most informative feature at each level, in order to split the training samples according to their respective classes; and 2) the probabilistic Bayesian classifier with naive independence assumptions (NB) (John and Langley, 1995). Obtaining the views of two different classifiers helps to provide a more comprehensive understanding of the qualities of the selected feature subsets. Table 5 shows the accuracies of the classifiers, trained using the

same 10-FCV folds as those used to perform FS. The quality of the underlying subsets have already been discussed in the previous section, the accuracies of the full data sets (without FS) is given in Table 3.

For features selected by CFS, the worst solutions (with an averaged score of 0.024 and size of 2.9) that are found by ABC, actually result in the best classification performance for both tested classifiers for the *secom* data set. This shows that for filter-based evaluators, a solution that achieves the highest evaluation score does not necessarily guarantee the best classifier model, subsequently learnt using such features, since these subsets are selected independent of the end classification algorithms, However, in general, there is a reasonable correlation between subset quality (judged by the CFS evaluator) and the classification accuracy. Feature subsets selected according to CFS also build slightly more accurate models than those constructed based on PCFS and FRFS, and more algorithms are able to find better performing solutions.

For the set of feature subsets selected using PCFS, CSA and TS seem to lead to the best classifiers overall. Each of the remaining algorithms also finds best results for one or more data sets. For the *secom* data set, all of the reviewed algorithms, apart from ACO, select features that do not contribute to good NB classifier models. Several models result in an averaged classification accuracy lower than 40%. A closer investigation reveals that in fact, local best solutions have been selected by these algorithms for a number of cross validation folds, which have a large, negative impact in the final 10-FCV results.

For classifiers built using feature subsets selected by FRFS, algorithms such as ACO, MA, SA, and TS all perform reasonably well. Both tested classifiers also tend to agree more often (than the two previous sets of experiments) in terms of predictive accuracy. The performance of a given classifier, and the evaluation score of its underlying feature subset are also well correlated for FRFS. Note that since FRFS generally helps to find more compact subsets, the resultant classification accuracies are also slightly lower when compared to those obtained by CFS or PCFS.

5 Conclusion

This paper has presented a comparative review of ten different NIMs that have been applied to the problem domain of FS. Existing methods that are based on the classic heuristics, such as ACO, GA, and PSO, are summarised. Several more recent developments, including CSA, ABC, and FA, which are proposed to solve more specific scenarios (or work with fixed types of feature subset evaluator) are introduced and modified considerably. These modifications enable the approaches to work with generic, feature subset-based evaluators and thus, allow their FS results to be compared systematically. Experimental evaluation shows that all reviewed algorithms are capable of finding good quality solutions. SA and TS are particularly powerful in optimising the evaluation scores of the CFS evaluator, and work well with a few high dimensional problem. Algorithms such as CSA, GA, and HS offer more balanced results for all tested subset evaluators, in terms of both evaluation score and subset size. HS also excels in size reduction and produces very compact fuzzy-rough reducts for most of the tested data sets. The selected feature subsets are verified via the use of two classification algorithms: C4.5 and NB, the performance of the resultant models generally agree with the quality measurement by the filter-based evaluators, although, there exist cases where feature subsets with very low evaluation lead to the most accurate classifiers.

Stochastic FS approaches have shown promising results in the area of meta-learning (Vilalta and Drissi, 2002). It is capable of generating diverse feature subset-based classifier

ensembles, and is effective in reducing the amount of redundancy in pre-constructed base classifier learners (Diao et al, to appear). It is worth investigating whether different NIMs can identify feature subsets with distinctive characteristics, so that they can jointly construct even higher quality FS ensembles. It is evident from the experimental results that each NIM may have its its own strength and weakness in dealing with different data sets, but a number of them generally work better in conjunction with the use of feature subset evaluators. It may be beneficial to develop a meta-framework in which suitable algorithms may be dynamically identified, and employed either concurrently or consecutively, in order to form a more intelligent, hybrid approach for FS.

References

Aha DW, Bankert RL (1996) A comparative evaluation of sequential feature selection algorithms. In: Fisher DH, Lenz HJ (eds) Learning from Data: Artificial Intelligence and Statistics V, Lecture Notes in Statistics, Springer-Verlag, New York, USA, pp 199–206

AlRashidi MR, El-Hawary M (2009) A survey of particle swarm optimization applications in electric power systems. IEEE Trans Evol Comput 13(4):913–918, DOI 10.1109/TEVC.2006.880326

Atyabi A, Luerssen M, Fitzgibbon S, Powers D (2012) Evolutionary feature selection and electrode reduction for eeg classification. In: 2012 IEEE Congress on Evolutionary Computation, pp 1 –8, DOI 10.1109/CEC.2012.6256130

Banati H, Bajaj M (2011) Fire fly based feature selection approach. International Journal of Computer Science Issues 8(2):473–479

Bellman R (1957) Dynamic Programming, 1st edn. Princeton University Press, Princeton, NJ, USA

Bengio Y, Grandvalet Y (2004) No Unbiased Estimator of the Variance of K-Fold Cross-Validation. Journal of Machine Learning Research 5:1089–1105

Brownlee J (2011) Clever Algorithms: Nature-Inspired Programming Recipes. Lulu Enterprises Incorporated

de Castro L, Von Zuben F (2002) Learning and optimization using the clonal selection principle. IEEE Trans Evol Comput 6(3):239 –251, DOI 10.1109/TEVC.2002.1011539

Chen X, Ong YS, Lim MH, Tan KC (2011) A multi-facet survey on memetic computation. IEEE Trans Evol Comput 15(5):591–607, DOI 10.1109/TEVC.2011.2132725

Chen Y, Miao D, Wang R (2010) A rough set approach to feature selection based on ant colony optimization. Pattern Recognition Letters 31(3):226 – 233

Chuang LY, Tsai SW, Yang CH (2011) Improved binary particle swarm optimization using catfish effect for feature selection. Expert Systems with Applications 38(10):12,699 – 12,707

Dash M, Liu H (1997) Feature selection for classification. Intelligent Data Analysis 1:131–156

Dash M, Liu H (2003) Consistency-based search in feature selection. Artif Intell 151(1-2):155–176, DOI 10.1016/S0004-3702(03)00079-1

Debuse J, Rayward-Smith V (1997) Feature subset selection within a simulated annealing data mining algorithm. Journal of Intelligent Information Systems 9:57–81, DOI 10.1023/A:1008641220268

Diao R, Shen Q (2010) Two new approaches to feature selection with harmony search. In: IEEE International Conference on Fuzzy Systems, pp 1 –7, DOI 10.1109/FUZZY.2010.5584009

Diao R, Shen Q (2012) Feature selection with harmony search. IEEE Trans Syst, Man, Cybern B 42(6):1509–1523

Diao R, Chao F, Peng T, Snooke N, Shen Q (to appear) Feature selection inspired classifier ensemble reduction. IEEE Trans Cybern

Dorigo M, Sttzle T (2010) Ant colony optimization: Overview and recent advances. In: Gendreau M, Potvin JY (eds) Handbook of Metaheuristics, International Series in Operations Research and Management Science, vol 146, Springer US, pp 227–263, DOI 10.1007/978-1-4419-1665-5_8

Ekbal A, Saha S, Uryupina O, Poesio M (2011) Multiobjective simulated annealing based approach for feature selection in anaphora resolution. In: Proceedings of the 8th international conference on Anaphora Processing and Applications, Springer-Verlag, Berlin, Heidelberg, pp 47–58

Emmanouilidis C, Hunter A, MacIntyre J (2000) A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator. In: Proceedings of the 2000 Congress on Evolutionary Computation, vol 1, pp 309–316

Frank A, Asuncion A (2010) UCI machine learning repository

Freitas AA (2008) A review of evolutionary algorithms for data mining. In: Maimon O, Rokach L (eds) Soft Computing for Knowledge Discovery and Data Mining, Springer US, pp 79–111, DOI 10.1007/978-0-387-69935-6_4

Geem ZW (ed) (2010) Recent Advances In Harmony Search Algorithm, Studies in Computational Intelligence, vol 270. Springer

Haktanirlar Ulutas B, Kulturel-Konak S (2011) A review of clonal selection algorithm and its applications. Artificial Intelligence Review 36(2):117–138, DOI 10.1007/s10462-011-9206-1

Hall MA (1998) Correlation-based feature subset selection for machine learning. PhD thesis, University of Waikato, Hamilton, New Zealand

Hart W, Krasnogor N, Smith J (eds) (2004) Recent Advances in Memetic Algorithms. Springer, Berlin, Heidelberg, New York

Hedar AR, Wang J, Fukushima M (2008) Tabu search for attribute reduction in rough set theory. Soft Comput 12(9):909–918

Hsu CN, Huang HJ, Schuschel D (2002) The ANNIGMA-wrapper approach to fast feature selection for neural nets. IEEE Trans Syst, Man, Cybern B 32(2):207–212

Jensen R, Shen Q (2005) Fuzzy-rough data reduction with ant colony optimization. Fuzzy Sets and Systems 149:5–20

Jensen R, Shen Q (2007) Fuzzy-rough sets assisted attribute selection. IEEE Trans Fuzzy Syst 15(1):73–89, DOI 10.1109/TFUZZ.2006.889761

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

Jensen R, Shen Q (2009a) Are more features better? a response to attributes reduction using fuzzy rough sets. IEEE Trans Fuzzy Syst 17(6):1456–1458

Jensen R, Shen Q (2009b) New approaches to fuzzy-rough feature selection. IEEE Trans Fuzzy Syst 17(4):824–838, DOI 10.1109/TFUZZ.2008.924209

John G, Langley P (1995) Estimating continuous distributions in bayesian classifiers. In: In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, pp 338–345

Kabir MM, Shahjahan M, Murase K (2011) A new local search based hybrid genetic algorithm for feature selection. Neurocomputing 74(17):2914 – 2928

Kabir MM, Shahjahan M, Murase K (2012) A new hybrid ant colony optimization algorithm for feature selection. Expert Systems with Applications 39(3):3747 – 3763

Karaboga D, Akay B (2009) A survey: algorithms simulating bee swarm intelligence. Artificial Intelligence Review 31(1-4):61–85, DOI 10.1007/s10462-009-9127-4

Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization 39(3):459–471

Karzynski M, Mateos l, Herrero J, Dopazo J (2003) Using a genetic algorithm and a perceptron for feature selection and supervised class learning in dna microarray data. Artificial Intelligence Review 20(1-2):39–51, DOI 10.1023/A:1026032530166

Ke L, Feng Z, Ren Z (2008) An efficient ant colony optimization approach to attribute reduction in rough set theory. Pattern Recognition Letters 29(9):1351 – 1357

Kohavi R, John GH (1997) Wrappers for feature subset selection. Artificial Intelligence 97(1):273–324

Kononenko I, Simec E, Robnik-Sikonja M (1997) Overcoming the myopia of inductive learning algorithms with RELIEFF. Applied Intelligence 7:39–55

Leardi R, Boggia R, Terrile M (1992) Genetic algorithms as a strategy for feature selection. Journal of Chemometrics 6(5):267–281, DOI 10.1002/cem.1180060506

ming Lee H, ming Chen C, ming Chen J, lu Jou Y (2001) An efficient fuzzy classifier with feature selection based on fuzzy entropy. IEEE Trans Syst, Man, Cybern B 31:426–432

Lee KS, Geem ZW (2005) A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Computer Methods in Applied Mechanics and Engineering 194(36-38):3902–3933, DOI 10.1016/j.cma.2004.09.007

Liu H, Motoda H (2007) Computational Methods of Feature Selection (Chapman & Hall/Crc Data Mining and Knowledge Discovery Series). Chapman & Hall/CRC

Liu Y, Wang G, Chen H, Dong H, Zhu X, Wang S (2011) An improved particle swarm optimization for feature selection. Journal of Bionic Engineering 8(2):191 – 200, DOI http://dx.doi.org/10.1016/S1672-6529(11)60020-6

Lpez FG, Torres MG, Batista BM, Prez JAM, Moreno-vega JM (2006) Solving feature subset selection problem by a parallel scatter search. European Journal of Operational Research 169(2):477–489

Mac Parthaláin N, Jensen R, Shen Q, Zwiggelaar R (2010a) Fuzzy-rough approaches for mammographic risk analysis. Intell Data Anal 14(2):225–244

Mac Parthaláin N, Shen Q, Jensen R (2010b) A distance measure approach to exploring the rough set boundary region for attribute reduction. IEEE Trans Knowl Data Eng 22(3):305–317, DOI 10.1109/TKDE.2009.119

Meiri R, Zahavi J (2006) Using simulated annealing to optimize the feature selection problem in marketing applications. European Journal of Operational Research 171(3):842 – 858

Muni D, Pal N, Das J (2006) Genetic programming for simultaneous feature selection and classifier design. IEEE Trans Syst, Man, Cybern B 36(1):106 –117, DOI 10.1109/TSMCB.2005.854499

Nakamura RYM, Pereira LAM, Costa KA, Rodrigues D, Papa JP, Yang XS (2012) Bba: A binary bat algorithm for feature selection. In: 25th SIBGRAPI Conference on Graphics, Patterns and Images, pp 291–297, DOI 10.1109/SIBGRAPI.2012.47

Nemati S, Basiri ME, Ghasem-Aghaee N, Aghdam MH (2009) A novel ACO-GA hybrid algorithm for feature selection in protein function prediction. Expert Systems with Applications 36(10):12,086–12,094

Oh IS, Lee JS, Moon BR (2004) Hybrid genetic algorithms for feature selection. IEEE Trans Pattern Anal Mach Intell 26(11):1424–1437, DOI 10.1109/TPAMI.2004.105

Ong YS, Krasnogor N, Ishibuchi H (2007) Special issue on memetic algorithms. IEEE Trans Syst, Man, Cybern B 37(1):2–5, DOI 10.1109/TSMCB.2006.883274

Palanisamy S, S Kanmani (2012) Artificial bee colony approach for optimizing feature selection. International Journal of Computer Science Issues 9(3):432–438

Senthamarai Kannan S, Ramaraj N (2010) A novel hybrid feature selection via symmetrical uncertainty ranking based local memetic search algorithm. Know-Based Syst 23(6):580–585, DOI 10.1016/j.knosys.2010.03.016

Shang C, Barnes D (2013) Fuzzy-rough feature selection aided support vector machines for mars image classification. Computer Vision and Image Understanding 117(3):202–213, DOI 10.1016/j.cviu.2012.12.002

Shen Q, Jensen R (2004) Selecting informative features with fuzzy-rough sets and its application for complex systems monitoring. Pattern Recognition 37(7):1351–1363

Shojaie S, Moradi M (2008) An evolutionary artificial immune system for feature selection and parameters optimization of support vector machines for ERP assessment in a P300-based GKT. In: International Biomedical Engineering Conference, pp 1 –5, DOI 10.1109/CIBEC.2008.4786065

Siedlecki W, Sklansky J (1989) A note on genetic algorithms for large-scale feature selection. Pattern Recognition Letters 10(5):335 – 347

Sivagaminathan RK, Ramakrishnan S (2007) A hybrid approach for feature subset selection using neural networks and ant colony optimization. Expert Systems with Applications 33(1):49 – 60

Sklansky J, Vriesenga M (1996) Genetic selection and neural modeling of piecewise-linear classifiers. International Journal of Pattern Recognition and Artificial Intelligence 10(05):587–612, http://www.worldscientific.com/doi/pdf/10.1142/S0218001496000360

Srinivasan S, Ramakrishnan S (2011) Evolutionary multi objective optimization for rule mining: a review. Artificial Intelligence Review 36(3):205–248, DOI 10.1007/s10462-011-9212-3

Stracuzzi DJ, Utgoff PE (2004) Randomized variable elimination. Journal of Machine Learning Research 5:1331–1364

Suguna N, Thanushkodi KG (2011) An independent rough set approach hybrid with artificial bee colony algorithm for dimensionality reduction. American Journal of Applied Sciences 8(3):261–266

Vilalta R, Drissi Y (2002) A perspective view and survey of meta-learning. Artificial Intelligence Review 18(2):77–95, DOI 10.1023/A:1019956318069

Wang X, Yang J, Teng X, Xia W, Jensen R (2007) Feature selection based on rough sets and particle swarm optimization. Pattern Recogn Lett 28(4):459–471, DOI 10.1016/j.patrec.2006.09.003

Witten IH, Frank E (2005) Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA

Wróblewski J (2001) Ensembles of classifiers based on approximate reducts. Fundam Inf 47(3-4):351–360

Wu X, Yu K, Ding W, Wang H, Zhu X (2013) Online feature selection with streaming features. IEEE Trans Pattern Anal Mach Intell 35(5):1178–1192, DOI 10.1109/TPAMI.2012.197

Xing EP, Jordan MI, Karp RM (2001) Feature selection for high-dimensional genomic microarray data. In: Proceedings of the Eighteenth International Conference on Machine Learning, Morgan Kaufmann, pp 601–608

Yang CS, Chuang LY, Chen YJ, Yang CH (2008) Feature selection using memetic algorithms. In: Third International Conference on Convergence and Hybrid Information Technology, vol 1, pp 416 –423, DOI 10.1109/ICCIT.2008.81

Yang J, Honavar V (1998) Feature subset selection using a genetic algorithm. Intelligent Systems and their Applications, IEEE 13(2):44 –49, DOI 10.1109/5254.671091

Yang XS (2008) Nature-Inspired Metaheuristic Algorithms. Luniver Press

Yusta SC (2009) Different metaheuristic strategies to solve the feature selection problem. Pattern Recogn Lett 30(5):525–534

Zhang L, Meng X, Wu W, Zhou H (2009) Network fault feature selection based on adaptive immune clonal selection algorithm. In: International Joint Conference on Computational Sciences and Optimization, vol 2, pp 969–973, DOI 10.1109/CSO.2009.342

Zheng Z, Wu X, Srihari R (2004) Feature selection for text categorization on imbalanced data. SIGKDD Explor Newsl 6(1):80–89, DOI 10.1145/1007730.1007741

Zheng L, Diao R, Shen Q (2014) Self-Adjusting Harmony Search-based Feature Selection. Soft Computing, to appear.

Zhu Z, Ong YS (2007) Memetic algorithms for feature selection on microarray data. In: Liu D, Fei S, Hou ZG, Zhang H, Sun C (eds) Advances in Neural Networks, Lecture Notes in Computer Science, vol 4491, Springer Berlin Heidelberg, pp 1327–1335, DOI 10.1007/978-3-540-72383-7_155

Zhu Z, Ong YS, Dash M (2007) Wrapper-filter feature selection algorithm using a memetic framework. IEEE Trans Syst, Man, Cybern B 37(1):70–76