Pattern Recognition Letters 30 (2009) 525–534

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Different metaheuristic strategies to solve the feature selection problem

Silvia Casado Yusta *

Department of Applied Economics, University of Burgos, Fac C. Económicas y Empresariales, C/Parralillos s/n, Burgos 09001, Spain

ARTICLE INFO

Article history:
Received 17 December 2007
Received in revised form 29 September 2008
Available online 10 December 2008

Communicated by L. Heutte

Keywords:
Feature selection
Floating search
Genetic Algorithm
GRASP
Tabu Search
Memetic Algorithm

ABSTRACT

This paper investigates feature subset selection for dimensionality reduction in machine learning. We provide a brief overview of the feature subset selection techniques that are commonly used in machine learning. Different metaheuristic strategies are proposed to solve the feature selection problem – GRASP, Tabu Search and Memetic Algorithm. These three strategies are compared with a Genetic Algorithm (which is the metaheuristic strategy most frequently used to solve this problem) and with other typical feature selection methods, such as Sequential Forward Floating Selection (SFFS) and Sequential Backward Floating Selection (SBFS). The results show that, in general, GRASP and Tabu Search obtain significantly better results than the other methods.

1. Introduction

Data preprocessing is an indispensable step in effective data analysis. It involves the preparation of data for data mining and machine learning, and aims to turn data into business intelligence or knowledge. Feature or variable selection is a preprocessing technique commonly used for high-dimensional data. This entails selecting a subset or list of attributes or variables that are used to construct models describing data.

Recently, there has been an immense increase in the amount of data collected and stored regarding almost every area of human endeavour. These collections of massive amounts of data have created opportunities to analyze and predict processes of interest. In today's fierce competitive business environment, firms need to rapidly turn these data into significant insights regarding their customers, markets, products and processes to guide their management strategies and investments to gain competitive advantages.

It is known that raw data sets by themselves do not provide much information. For this reason, raw data must first be processed to extract patterns or useful knowledge. To this end, the development of effective and efficient methods for deriving knowledge from these data is becoming increasingly important (Fayyad et al., 1996).

A typical real-world dataset consists of as many features as are deemed necessary. This is constrained by the following: (1) knowledge concerning the domain of interest and, in turn, knowledge concerning the essential features that capture knowledge in this domain; (2) the availability of these essential features; (3) the resources available to collect these available essential features; (4) the resources available to store, maintain, and retrieve these features. Given these constraints, it is clear that not all the features that are stored in the resulting data set are necessary or sufficient to learn the concept of interest. Assuming that all the necessary features are present in these data, feature selection is the problem of choosing a small subset of features that ideally are necessary and sufficient to describe the target concept (Kira and Rendell, 1992).

Basically, the object of the feature selection problem is to find a subset of variables that can be used to carry out the classification task in a optimum way. One usually starts with a given set of features and then attempts to derive an optimal subset of features leading to high classification performance.

The term feature selection refers to algorithms that output a subset of the input feature set. In supervised learning we have a set of examples characterized by the same attributes as the instances and another attribute corresponding to the class they belong to. Using this set of examples we can create and generalize a rule or set of rules that allows us to classify the instance set with the greatest possible precision. The research on feature selection dates back to the early 1960s (for an overview and biographical notes see Devijver and Kittler (1982) or Siedlecki and Sklansky

* Tel.: +34 947 25 93 80; fax: +34 947 25 80 13.
E-mail address: scasado@ubu.es

0167-8655/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2008.11.012

(1988)). Extensive research into feature selection has been carried out over the past four decades. Many studies on feature selection are related to medicine and biology, such as those by Sierra et al. (2001), Ganster et al. (2001), Inza et al. (2000), Lee et al. (2003), Shy and Suganthan (2003), Tamoto et al. (2004) and Baek et al. (2008).

The search for a variable subset is a NP-hard problem (Kohavi, 1995; Cover and Van Campenhout, 1997; Cotta et al., 2004). Therefore, the optimal solution cannot be guaranteed to be acquired except when performing an exhaustive search in the solution space. The use of metaheuristic techniques allows us to obtain reasonably good solutions without having to explore the whole solution space. The quality of "heuristic" solutions varies depending on the methods used. As found in other optimization problems, metaheuristic techniques have proved to be superior methodologies. In real-world applications, people are more interested in obtaining good solutions in a reasonable amount of time rather than being obsessed with optimal solutions. Therefore, we favour metaheuristic methods that are efficient for dealing with real-world applications. The most frequently used metaheuristic strategy applied to the variable selection problem is the Genetic Algorithm (GA) (Siedlecki and Sklansky, 1989; Bala et al., 1996; Jourdan et al., 2001; Oliveira et al., 2003; Inza et al., 2001a,b; Shin and Lee, 2002; Wong and Nandi, 2004). The results obtained show that GA are appropriate methods for this problem.

In the present work, other different metaheuristic strategies are proposed to solve the feature selection problem. Specifically, GRASP (Feo and Resende, 1995), Tabu Search (Glover and Laguna, 1997, 2002) and Memetic Algorithms (Moscato, 2002). We compare our proposed strategies with a GA (this strategy has frequently been used for solving the feature selection problem, in fact, it may be the one most frequently used) and with other typical feature selection methods such as Sequential Forward Floating Search (SFFS) and Sequential Backward Floating Search (SBFS).

In addition, the special case treated in this work involves only using quantitative variables to carry out the classification. The exclusive use of quantitative variables allows better measurement and comparison of their classificatory capacity.

The rest of the paper is organized as follows: Section 2 presents an overview of feature selection methods; in Section 3, the problem is modelled; Section 4 describes the metaheuristic methods proposed: GRASP, Tabu Search and Memetic Algorithm; Section 5 is devoted to computational experiments; and Section 6 summarises the main conclusions.

2. An overview of feature selection methods

Dimensionality reduction is a commonly used step in machine learning, especially when dealing with a high-dimensional feature space. The original feature space is mapped onto a new space of reduced dimensions. Identification of relevant features is extremely important for classification tasks (improving accuracy and reducing computational costs), as well as for understanding the relative significance of features. Dimensionality reduction is usually performed either by selecting a subset of the original dimensions or by constructing new dimensions. Feature selection has often been distinguished from feature extraction in the research literature. Feature selection is aimed at individually finding from among a large group of features the best features that maximize classification ability. More general methods that create new features based on transformations or combinations of the original feature set are termed feature extraction algorithms. Feature extraction has been traditionally viewed as a process to use different weighting schemes to linearly combine features to produce a reduced number of new, ideally uncorrelated, features (as in Principal Component Analysis, PCA, and Linear Discriminant Analysis, LDA) or to combine them non-linearly (as in neural networks).

The first critical issue is whether to use feature selection or feature extraction. There is no clear-cut evidence that one is superior to the other for all types of tasks. They also have their own limitations. Feature extraction methods, such as PCA (Fukunaga and Koontz, 1970) – which use all of the available features and then combine them linearly to generate an, ideally uncorrelated, set of features – can fail if the variability of some of the features is nearly zero, and data assumptions are not met. Similarly, for feature selection, exhaustive search – and most sequential search methods – for high-dimensional datasets are unfeasible. Methods that examine features individually are simple to use, but yield poor feature subsets. On the other hand, those methods that try different feature combinations simply take too much time and run out of computer memory. Finally, features that appear to be important on training data may not fare that well on test data.

As mentioned, this work is only concerned with the problem of feature selection.

Feature selection (so-called variable selection) has become the focus of much research on applications for which datasets with tens or hundred of thousands of variables are available. Feature selection problems are found in many machine learning tasks, including classification, regression, time-series prediction, etc.

There has been a resurgence of interest in applying feature selection methods due to the large numbers of features encountered, for example, in the following types of problems:

(1) Applications where data taken by multiple sensors are combined. Jain and Vailaya (1996), for instance, have merged both colour and shape features to provide improved retrieval accuracy for a trademark image database.
(2) Integration of multiple models, where the parameters from different mathematical models are pooled for the purpose of classification, as in (Solberg and Jain, 1995).
(3) Data mining applications, where the goal is to discover the hidden relationships from among a large number of features, as in (Punch et al., 1993).

An appropriate feature selection can enhance the effectiveness of an inference model. Liu and Motoda (1998) indicated that the effects of feature selection are as follows: (1) to improve performance (speed of learning, predictive accuracy, or simplicity of rules); (2) to visualize the data for model selection; and (3) to reduce dimensionality and remove noise. Although feature selection offers many advantages, it may run the risk of decreasing accuracy or over-fitting. Thus, an important consideration is how to achieve and maintain the expected classification performance and avoid these risks.

Feature subset selection is an important area in the field of machine learning and pattern recognition and many approaches have been proposed. The most important are described below.

The universal algorithms of feature selection are often divided into three modalities: wrappers, filters and embedded (Guyon and Elisseeff, 2003; Kohavi and John, 1997). If the feature selection method works in conjunction with the learning algorithm it is using a wrapper approach (Devijver and Kittler, 1982; Yang and Honavar, 1997). The wrapper approach performs selection taking into account the classifier as a black box and ranking the subset of features by their predictive power. The filter approach (Kira and Rendell, 1992; Anderson and Matessa, 1992; Lanzi, 1997) selects the features using a preprocessing step that ignores the learning algorithm. The main disadvantage of this procedure is that it ignores the effect of the subset of features in the learning algorithm.

Obviously, exhaustive search under these two approaches can be performed, if the number of variables is not too large. However,

the problem is known to be NP-hard (Kohavi, 1995; Cover and Van Campenhout, 1997; Cotta et al., 2004) and the search quickly becomes computationally intractable. The approaches may incur high computational costs when the number of variables is too large. Embedded methods, in contrast to wrapper approaches, select features while taking into account the classifier design.

2.1. Wrappers approach

A wrappers model consists of two phases (Liu and Motoda, 1998):

Phase 1 – feature subset selection, which selects the best subset using the accuracy of the classifier (on the training data) as a criterion.
Phase 2 – learning and testing, where a classifier is learned from the training data with the best feature subset, and is tested on the test data.

The wrappers approach consists of using the prediction performance of a given learning machine to assess the relative usefulness of subsets of variables. It performs selection taking into account the classifier as a black box and ranking the subset of features by their predictive power. Although the wrappers approach is often criticized to be a "brute force" method which involves massive computation time, some researchers have different opinions. In this regard, Reunanen (2003) indicated that coarse search strategies may alleviate the problem of over-fitting.

Due to the fact that a full search requires 2^n different evaluations, "forward" selection or "backward" elimination methods are used. These methods are also referred to as "sequential" methods. These are the most commonly used methods for performing feature selection. Sequential Forward Selection, SFS (Whitney, 1971) starts with the empty set and variables are progressively incorporated into larger and larger subsets, whereas in Sequential Backward Selection, SBS (Marill and Green, 1963), one starts with the set of all the variables and progressively eliminates the least promising ones. Both these methods suffer from the so-called "nesting effect". This means that in the case of the backward elimination method the discarded features cannot be re-selected, whereas in the case of the forward selection method once the features are selected they cannot be discarded later. The result is that the methods are only suboptimal. In addition, forward selection and backward elimination require high computation time when the variable is very large (Kohavi and John, 1997).

A definite improvement can be obtained by combining SFS and SBS to avoid the nesting effect. The "plus-*l*-take-away-*r*" method was proposed to prevent the "nesting" effect.

This method (Stearns, 1976) consists of applying SFS l times followed by r steps of SBS, repeating this fixed cycle of forward and backward selection until the required number of features is reached. In this case, features that have been previously added can be removed in posterior steps thus avoiding the nesting effect. This method allows "fixed backtracking" defined by the values of l or r depending on whether the search is top-down or bottom-up. Even though the problem of nested features can be partially overcome with this procedure, another problem arises: there is no theoretical guide to determine the appropriate value of $l$and r to obtain good enough solutions with a moderate amount of computation.

The idea behind the plus-*l*-take-away-*r* method aimed at counteracting the nesting effect, can be more efficiently implemented by considering the conditional inclusion and exclusion of features controlled by the value of the criterion itself.

Pudil et al. (1994a) introduced the concept of "floating feature search" and two "floating" selection methods, SFFS (Sequential Forward Floating Selection) and SBFS (Sequential Backward Floating Selection).

The floating search methods are related to the plus-*l*-take-away-*r* algorithm, but in contrast to the latter, the number of forward and backtracking steps is dynamically controlled instead of being fixed beforehand.

The SFFS and SBFS methods are characterized by the changing number of features included or eliminated at different stages of the procedure; According to Jain and Zongker (1997), SFFS and SBFS are probably the most effective feature selection techniques.

Basically, in the case of forward search, the algorithm starts with a null feature set and, for each step, the best feature that satisfies some criterion function is included in the current feature set, i.e. one step of the sequential forward selection is performed. The algorithm also verifies the possibility of improvement of the criterion if some feature is excluded. In this case, the worst feature (in relation to the criterion) is eliminated from the set, that is, it one sequential backward selection step is performed. Therefore, the SFFS proceeds by dynamically increasing and decreasing the number of features until the desired number of features is reached. SFBS works similarly, but starts with the full feature set and performs the search until the desired dimension is reached, using SBS and SFS steps. Although neither of these methods can always guarantee obtaining the best subset of features, their performance is very good compared to other search methods.

The particular advantage of the floating search methods over the plus-*l*-take-away-*r* methods is that they can make more than one sweep through feature set subsets to achieve good performance. In practice, searching with dynamic backtracking has very robust performance and if one had to choose between feature set search methods then the floating search procedures would be the first choice (Pudil et al., 1994b). In addition, the floating methods are at least as good as the best sequential methods.

Although the floating methods are considered to be more intelligent, they are still suboptimal. The drawback of these sequential floating methods is that they are still likely to become trapped in a local optimal solution even if the criterion function is monotonic and the scale of the problem is quite small.

2.2. Filters approach

The filters approach is based on the intrinsic properties of the data, rather than being biased toward a particular classifier. The essence of filters is to seek the relevant features and eliminate the irrelevant ones. A filters model of feature selection also consists of two phases (Liu and Motoda, 1998):

Phase 1 – feature selection using measures such as information, distance, dependence, or consistency; no classifier is engaged in this phase.
Phase 2 – this is the same as in the wrappers model, where a classifier is learned on the training data with the selected features, and tested on the test data.

In addition to being based on the intrinsic properties of the data, the filters approach has other characteristics, as follows:

1. Measuring information gains, distance, dependence, or consistency is usually cheaper than measuring classifier accuracy, so a filters method can produce a subset faster, all things being equal.
2. Because of the simplicity of the measures and low time complexity, a filters method can handle larger amount of data than a classifier can; so in the case where a classifier cannot directly be learned from the large amount of data, it can be used to

reduce data dimensionality so that the classifier can be learned from such data. However, there is a danger that the features selected by a filters model may not allow a learning algorithm to fully exploit its potential.

ReliefF is a popular filter algorithm in microarray classification problems because of its simplicity. One example is DNA microarray data (Golub et al., 1999; Ramaswamy et al., 2001), where the number of features (genes) is typically on the order of thousands or even tens of thousands, whereas the number of the useful genes is believed to be in the range of tens and hundreds. Due to the high cost of collecting a large amount of patient data, there are usually only tens or at most hundreds of samples for training.

Other methods or algorithms used in the literature for the feature selection problem include the following:

- The Branch-and-Bound (BB) feature selection algorithm, proposed by Narendra and Fukunaga (1977), can be used to find the optimal subset of features. One drawback is that this method requires the feature selection criterion function to be monotonic. This means that the addition of new features to a feature subset can never decrease the value of the criterion function. Unfortunately, the monotonic condition is seldom satisfied. Nevertheless, its computational cost is prohibitive for large feature spaces: in the worst case, it performs an exhaustive search and its time complexity is exponential according to the dimension of the feature space. As pointed out by Jain and Zongker (1997) the BB method is still impractical for problems with very large feature sets.
- The Max–Min (MM) method proposed by Backer and Shipper is a computationally efficient method (Backer and Shipper, 1977). It only evaluates the individual and pairwise merits of features. The results achieved with this method are invariably rather unsatisfactory. The experiments performed by many researchers show that this method gives the poorest results (Kittler, 1978; Jain and Zongker, 1997).
- Siedlecki and Sklansky (1989) introduced the use of GA for feature selection. In a GA approach, a given feature subset is represented as a binary string (a "chromosome") of length n, with a zero or one in position i denoting the absence or presence of feature i in the set. Note that n is the total number of available features. A population of chromosomes is maintained. Each chromosome is evaluated to determine its "fitness," which determines how likely the chromosome is to survive and breed into the next generation. New chromosomes are created from old chromosomes by the following processes: (1) crossover, where parts of two different parent chromosomes are mixed to create offspring and (2) mutation, where the bits of a single parent are randomly disturbed to create a child. Other works applying the GA approach to the feature selection problem include those of Bala et al. (1996), Jourdan et al. (2001), Oliveira et al. (2003), Inza et al. (2001a,b), Shin and Lee (2002), and Wong and Nandi (2004).

Despite some progress, the feature selection techniques available for large feature sets are not yet completely satisfactory. They are either computationally feasible but far from optimal, or they are optimal or almost optimal but cannot cope with the computational complexity of feature selection problems of realistic size. Further research is needed to develop more powerful methods for feature selection, with the aim of providing very good results and being computationally more efficient. In the present work, three metaheuristic strategies are proposed to solve the feature selection problem: GRASP, Tabu Search and Memetic Algorithm.

3. The feature selection problem

We can formulate the problem of selecting the subset of features with superior classificatory performance as follows: Let V be the original set of features with cardinality m. Let d represent the desired number of features in the selected subset, X, $X \subseteq V$. Let the feature selection criterion function for the set X be represented by $J(X)$. Without any loss of generality, let us consider a high value of J to indicate a better feature subset. Formally, the problem of feature selection is to find a subset $X \subseteq V$ such that $|X| = d$ and

$$J(X) = \max_{Z \subseteq V, |Z|=d} J(Z).$$

In this case, the criterion function for assessing the "goodness" of a feature subset is computed as follows: Let C be a set of cases. For each case we know the class to which it belongs (only two classes are considered). We make the following partition in C: $C = C_1 \cup C_2$, where C_1 and C_2 have approximately the same number of cases and the same proportions of cases in both classes. For each case in C_2 we calculate the Euclidean distance to every case in C_1 and the class with the closest case is assigned. The value of $J(X)$ is the percentage of hits in the assigned classes. That is, the number of times that the assigned class is the same as the real class.

An exhaustive approach to this problem would require examining all $\binom{m}{d}$ possible d-subsets of the feature set V. The number of possibilities grows exponentially, making exhaustive search impractical for even moderate values of m.

As mentioned, this is an NP-hard problem (Kohavi, 1995; Cover and Van Campenhout, 1997; Cotta et al., 2004) and therefore it is appropriate to use heuristic and metaheuristic techniques.

4. Description of the algorithms

4.1. Description of the GRASP algorithm

GRASP (Feo and Resende, 1995) is an iterative process, where each iteration consists of two phases: construction and local search. A feasible solution is built in the construction phase, and then its neighbourhood is explored by the local search. The result is the best solution found over all iterations. The operating scheme is as follows:

Repeat
- *Construction phase*
- *Local search*

Till a stop criterion is satisfied

The construction phase of GRASP is an iterative process where, at each iteration, the elements $v_j \in V/X$, $j = 1, \ldots, m$ – which are the elements that do not belong to the solution – are evaluated by a greedy function g that estimates the value of including it in the partial solution. They are ordered by their estimated value in a list called the restricted candidate list (RLC) and one of them is randomly chosen and included in the solution. The size of the RLC is limited by the parameter α. Only the elements $v_j \in V/X$, $j = 1, \ldots, m$, whose g values $g(v_j) \leqslant \alpha \cdot g_{max} + (1 - \alpha) \cdot g_{min}$ are placed in the RLC where g_{max} is the maximum value of g obtained for all elements in V/X and g_{min} is the minimum value of g obtained for all elements in V/X. In addition, the α parameter is used to control the degree of randomness of the procedure. The greater the value of α, the greater the degree of randomness. If $\alpha = 1$, the procedure is totally random, because the RCL would be made up of all the variables not included in the solution. If $\alpha = 0$, the RLC would only be made up of the variable corresponding to g_{min}. In this work $\alpha = 0.85$ is considered.

Let X be the solution that is going to be built; the constructive procedure is as follows:

1. *Make* $X=\emptyset$
2. *Calculate* $g(v_j)$, $j = 1 \ldots, m$
3. *While* $|X| < d$ *make*:
   - *Determine* $g_{min} = min\{g(v_j),\ j = 1, \ldots, m\}$ *and* $g_{max} = max\{g(v_j),\ j = 1, \ldots, m\}$
   - *Build* $RLC = \{v_j g(v_j) \leqslant \alpha \cdot g_{max} + (1 - \alpha) \cdot g_{min}\}$
   - *Select* $v_{j^*} \in RLC$ *randomly and make* $S = S \cup \{v_{j^*}\}$

In this case, the greedy function, g, guiding the entry of variables into the solution, is the "in-group variability" (IGV). For each feature v_j defined on the n cases under consideration, that is $v_j = \{v_j^1, v_j^2, \ldots, v_j^n\}$, the "in-group variability" is defined as follows:

$$IGV(v_j) = \sum_{i=1}^{n} (v_j^i - \mu_{cl(i)})^2$$

where $\mu_{cl(i)}$ is the mean of the variable v_j in the cases of the class the individual i belongs to.

The lower the value of IGV (v_j) the greater the value of including a feature in the partial solution.

This construction procedure is executed several times generating different solutions and only the best solution is selected to be used as the initial solution for the local search phase.

A local search is then performed to attempt to improve each constructed solution. This works by successively replacing the current solution by a better one from its neighbourhood until no more better solutions are found. Let X be a solution and the neighbourhood ($N(X)$) of this solution is defined as follows:

$$N(X) = \{X'/X' = X \cup \{v_{j'}\} - \{v_j\}, \forall v_j \in X, v_{j'} \notin X\}.$$

The local search procedure can be described as follows:

Read initial solution X
Repeat
- *Make previous_value* $= J(X)$
- *Search* $J(X^*) = max\{J(X')/X' \in N(X)\}$
- *If* $J(X^*) > J(X)$ *then make* $X = X^*$
  till $J(X^*) \leqslant$ *previous_value*

Thus, the procedure ends when no exchange provides a better solution.

4.2. Description of Tabu Search

Tabu Search (TS) is a strategy proposed by Glover (1989, 1990). This procedure explores the solution space beyond the local optimum. 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. Recent and comprehensive tutorials on Tabu Search that include all types of applications can be found in (Glover and Laguna, 1997, 2002).

In the present work, a simple Tabu Search has been designed that uses the neighbouring moves employed in the Local Search procedure previously described (Section 4.1). These moves consist in exchanging an element that is in solution X for an outside element at each step. In order to avoid repetitive looping when a move is performed, consisting in exchanging v_j from X for $v_{j'}$ from V/X, element v_j is prevented from returning to X for a certain number of iterations. We define

vector_tabu (v_j) = the number of the iterations in which element v_j leaves X.

Some 'tabu' moves can be permitted under specific conditions ("aspiration criterion"), for example, to improve the best solution found. The Tabu Search method is described next, where X is the current solution and X^* the best solution. The *Tabu_Tenure* parameter indicates the number of iterations during which an element is not allowed to return to X. After different tests, *Tabu_Tenure* was set as d.

Tabu Search Procedure

Read initial solution X
Do *vector_tabu*$(v_j) = -$*Tabu_Tenure*, $j = 1, \ldots, m$; *niter* $= 0$, *iter_better* $= 0$ *and* $X^* = X$
Repeat
- *niter* $=$ *niter* $+ 1$
- *Calculate* $\Delta_{jj'} = J(X \cup \{v_{j'}\} - \{v_j\})$
- *Determine* $\Delta_{j^*j'^*} = max\{\Delta_{jj'}/\forall v_j \in X, v_{j'} \notin X$ *verifying*: *niter* $>$ *vector_tabu*$(v_j) +$ *Tabu_Tenure* *or* $\Delta_{jj'} > J(X^*)$ ('*aspiration criterion*')}
- *DoX* $= X \cup \{v_{j'^*}\} - \{v_{j*}\}$ *and vector_tabu* $(v_{j^*}) =$ *niter*
- *If* $J(X) > J(X^*)$ *then do*: $X^* = X$, $J^* = J$ *and iter_better* $=$ *niter*;

until niter $>$ *iter_better* $+ 2 \cdot m$

That is, this procedure terminates when $2 \cdot m$ iterations have taken place without improvement.

4.3. Description of the Memetic Algorithm

Memetic Algorithms (MA) are population-based methods which combine local search procedures with crossing or mutating operators; due to their structure some researchers have called them Hybrid Genetic Algorithms, Parallel Genetic Algorithms (PGAs) or Genetic Local Search methods. The method is gaining wide acceptance particularly for the well-known problems of combinatorial optimization. A recent tutorial can be found in (Moscato, 2002; Moscato and Cotta, 2003). Similarly, information on Memetic Algorithms can be found at the Memetic Algorithms' Home Page: http://densis.fee.unicamp.br/~moscato/memetic_home.html.

The MA works by generating an initial population of random solutions to the problem to be solved and improving these solutions via local search. Prior to this the problem has to be formulated in vectorial or chromosome notation. In a second step, the fitness value of every chromosome (potential solution) – according to an objective function that classifies the solutions from the best to the worst – is computed and the subset of the *nsel* best solutions from the initial population is selected. In the third step, the MA applies the crossover and mutation operators, where information from the former solution is exchanged and mutated. The obtained solutions ("children" solutions) are then improved with a local search procedure. Finally, if a child solution improves the worst solution in population, this solution is replaced.

For this problem binary representations of the solutions are used. Thus, for example,

$$X = (1, 0, 0, 0, 1, 0, 1, 0, 0, 1)$$

represents a solution that contains variables 1, 5, 7 and 10.

An essential element in MA functioning is the crossover operator. Important similarities between highly fitted strings can help guide the search. Thus, a schema is defined as a similarity template that describes a subset of strings with similarities at certain string positions. Good schemata are hidden in parent chromosomes and the crossover operator is an efficient way to extract them in order to produce an evolved population from the parent population. In this study, the crossover operator is the well-known "one point crossover", that is, let X and X' be a pair of solutions

$X = (x_1, x_2, \ldots, x_m)$ and $X' = (x'_1, x'_2, \ldots, x'_m)$

and a 'crossover point' is randomly generated between 1 and $m - 1$, (pto_cruce), in such a way that the new child solutions are defined as

$$X^* = (x_1, x_2, \ldots, x_{pto\_cruce}, x'_{pto\_cruce+1}, \ldots, x'_m),$$

$$X^{**} = (x'_1, x'_2, \ldots x'_{pto\_cruce}, x_{pto\_cruce+1}, \ldots, x_m).$$

The mutation operator is responsible of maintaining diversity. Each of the elements of each new offspring can mutate with a probability of p_mut. To decide if a specific element is changed, a random value is generated in (0,1); if this value is lower than p_mut, then the element is changed, and this change consists of taking out this element and randomly selecting a new element which is not in the solution.

The functioning of the Memetic Algorithm can be described as follows:

Table 1
Comparison of Genetic Algorithm, SFFS, SBFS and the proposed methods.

Database	*D*	GRASP	Tabu Search	Memetic Algorithm	Genetic Algorithm	SFFS	SBFS
Spambase	3	*J_mean:* ***0.821***	***0.821***	0.78	0.729	0.723	0.723
		St_d: 0.0481	0.0481	0.0545	0.040	0.0442	0.0442
	4	*J_mean:* ***0.824***	***0.824***	0.791	0.766	0.713	0.713
		St_d: 0.0447	0.0447	0.0545	0.0305	0.0452	0.0452
	5	*J_mean:* ***0.86***	0.825	0.796	0.756	0.717	0.717
		St_d: 0.0387	0.0455	0.0350	0.043	0.0444	0.0444
	6	*J_mean:* ***0.862***	0.833	0.813	0.78	0.755	0.755
		St_d: 0.04391	0.0397	0.0502	0.0226	0.0494	0.0494
	7	*J_mean:* ***0.864***	0.829	0.808	0.786	0.741	0.741
		St_d: 0.0445	0.0414	0.0434	0.0394	0.0670	0.0670
Waveform	3	*J_mean:* ***0.802***	***0.802***	0.788	0.768	0.77	0.761
		St_d: 0.0656	0.0656	0.0367	0.0367	0.0561	0.0490
	4	*J_mean:* ***0.82***	***0.82***	0.807	0.771	0.787	0.787
		St_d: 0.0343	0.0343	0.054	0.041	0.0507	0.0507
	5	*J_mean:* ***0.815***	***0.815***	0.785	0.755	0.767	0.767
		St_d: 0.0403	0.0403	0.0383	0.0535	0.0427	0.0427
	6	*J_mean:* ***0.832***	***0.832***	0.786	0.769	0.797	0.797
		St_d: 0.051	0.051	0.0341	0.0397	0.0611	0.0611
	7	*J_mean:* ***0.84***	***0.84***	0.774	0.766	0.798	0.798
		St_d: 0.0380	0.0380	0.0457	0.0389	0.0637	0.0637
Ionosfera	3	*J_mean:* ***0.905***	***0.905***	0.885	0.835	0.861	0.864
		St_d: 0.0092	0.0092	0.0134	0.0104	0.0235	0.0224
	4	*J_mean:* 0.907	***0.912***	0.9	0.843	0.897	0.897
		St_d: 0.011	0.0081	0.0205	0.0127	0.0106	0.0102
	5	*J_mean:* 0.906	***0.914***	0.8832	0.8539	0.9	0.89
		St_d: 0.016	0.0046	0.0079	0.0135	0.0647	0.0089
	6	*J_mean:* ***0.914***	0.898	0.902	0.875	0.894	0.9
		St_d: 0.0046	0.0130	0.0173	0.0119	0.0169	0.0111
	7	*J_mean:* 0.89	***0.905***	0.88	0.877	0.893	0.896
		St_d: 0.019	0.0186	0.0058	0.0111	0.0181	0.0089
Vehicle	3	*J_mean:* ***0.893***	0.87	0.867	0.795	0.855	0.848
		St_d: 0.0202	0.0130	0.021	0.0569	0.0179	0.0148
	4	*J_mean:* ***0.885***	***0.885***	0.861	0.819	***0.885***	0.856
		St_d: 0.0212	0.0212	0.023	0.0785	0.0083	0.0106
	5	*J_mean:* ***0.896***	***0.896***	0.87	0.8283	***0.896***	0.859
		St_d: 0.0203	0.0203	0.0684	0.0163	0.0158	0.0191
	6	*J_mean:* 0.895	***0.933***	0.91	0.854	0.885	0.878
		St_d: 0.0085	0.0165	0.0183	0.024	0.0127	0.0127
	7	*J_mean:* ***0.926***	***0.926***	0.915	0.894	0.91	0.91
		St_d: 0.0157	0.0157	0.0856	0.0093	0.0157	0.0171
Wincosin	3	*J_mean:* 0.915	***0.938***	0.909	0.893	0.9	0.9
		St_d: 0.0145	0.0078	0.0129	0.059	0.0574	0.0574
	4	*J_mean:* ***0.937***	***0.937***	0.93	0.9	0.9	0.9
		St_d: 0.0079	0.0079	0.0125	0.060	0.0579	0.0579
	5	*J_mean:* ***0.938***	0.911	0.905	0.906	0.906	0.904
		St_d: 0.0081	0.014	0.0085	0.044	0.0519	0.0595
	6	*J_mean:* ***0.926***	0.907	0.925	0.908	0.91	0.908
		St_d: 0.0127	0.0128	0.0117	0.053	0.0453	0.0303
	7	*J_mean:* ***0.934***	***0.937***	0.924	0.9	0.917	0.9
		St_d: 0.0125	0.0085	0.0162	0.073	0.03972	0.0377
German	3	*J_mean:* 0.926	***0.929***	0.922	0.922	0.86	0.851
		St_d: 0.0081	0.0066	0.0113	0.0077	0.0684	0.068
	4	*J_mean:* 0.926	***0.93***	0.928	0.926	0.863	0.854
		St_d: 0.070	0.0073	0.0086	0.0068	0.0456	0.0456
	5	*J_mean:* ***0.929***	0.926	0.928	0.927	0.856	0.856
		St_d: 0.0082	0.0076	0.0053	0.00701	0.0472	0.0472
	6	*J_mean:* ***0.928***	0.926	0.927	0.927	0.866	0.86
		St_d: 0.0082	0.0081	0.007	0.0006	0.0685	0.0562
	7	*J_mean:* ***0.927***	***0.927***	***0.927***	***0.927***	0.871	0.86
		St_d: 0.0068	0.0104	0.0038	0.0064	0.0633	0.0522

1. *Generate an initial population of random solutions of size npop*
2. *Improve these solutions via local search.*
3. *Repeat*:
   (a) *Select the subset of nsel elements of the population with the highest fitness value.*
   (b) *Crossreprodution*: *cross a pair of these solutions(parents)to give rise to new solutions(children). From each pair of parents a new pair of child solutions is generated.*
   (c) *Mutation*: *the "child" solutions can change some of their elements with a small probability p_mut.*
   (d) *Improve child solutions with local search.*
   (e) *Population replacement*: *substitute the worst solutions of the population with the new child solutions.*
   Until reaching a number of n_iter iterations.
4. *Choose the solution with the highest fitness value as the final solution*

Each of the iterations in which the sequence of the selection of "parents", crossreproduction, mutation and improved solutions is computed is called "generation". The Memetic Algorithm finishes when a number n_iter of "generations" is reached.

In this work *npop* = 10; in each generation four parents are selected, and the value of pmut is 0.10.

5. Computational results

In order to check and compare the efficacy of the different metaheuristic strategies previously described, a series of experiments was run with different problems.

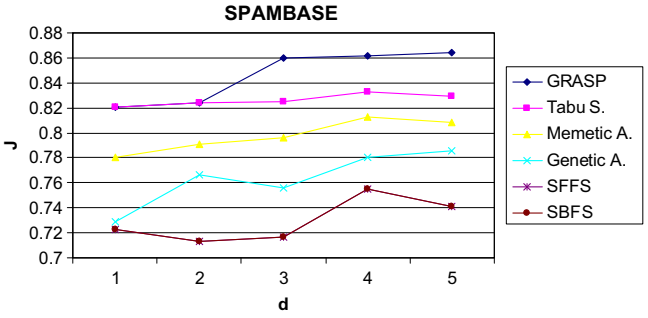

Fig. 1. Objective function values, *J*, obtained for the various techniques for the Spam database (note that the values of SFFS and SBFS are equal).

5.1. Previous test

In this section, we compare the three metaheuristic strategies described in Section 4 with a Genetic Algorithm and the SFFS and SBFS methods.

As mentioned, GAs have often been used to solve the feature selection problem, and may even be the most frequently used strategy for solving this problem. The GA used in this case is very similar to the MA described in Section 4.3. Specifically, it follows the same general scheme of the Memetic Algorithm but does not perform steps 2 and 3-d of the MA and which consists in improving the solutions via local search (in step 2 after generating the initial population and in step 3-d after the mutation).

We compare metaheuristic strategies with floating search methods (Pudil et al., 1994a) because though neither of these methods (SFFS and SBFS) can always guarantee providing the best subset of features, their performance has been found to be very good compared with other search methods. As mentioned, in practice, searching with dynamic backtracking has very robust performance and, if one had to choose between feature set search methods then the floating search procedures would be the first choice. Finally, it should be emphasised that the floating methods are at least as good as the best sequential methods.

Note that the experiments are conducted using the *k*-nearest neighbours classification algorithm.

Six databases from the well-known data repository of the University of California, UCI, (see Murphy and Aha, 1994) have been used for these experiments. The selected databases have enough instances – at least around 10 cases – for every degree of freedom in the training sets. This is recommended to avoid over-fitting.

A short description of these databases follows:

- Spambase database: 57 attributes, 2 classes and 4601 cases.
- Waveform database: 40 attributes with continuous values, 3 classes and 5000 instances. Only the examples from the first two classes have been taken into account, and therefore 3347 instances have been used.
- Ionosphere database: 34 attributes, 2 classes and 351 cases.
- Vehicle database: 18 variables and 4 classes. The instances corresponding to the classes bus and saab have been considered, that is, 435 instances.
- Wisconsin database breast cancer: 30 attributes, 2 classes and 569 cases.
- German-Data-Numeric database: a database with 24 numerical attributes, 2 classes and 1000 cases.

For each database, a 1×10 cross-validation was developed to measure and test the methods used. All the experiments were done

Table 2
Results obtained for each database and strategy.

Database	GRASP	Tabu Search	Memetic Algorithm	Genetic Algorithm	SFFS	SBFS
Spambase	*J_mean: 0.846*	*0.8264*	*0.7976*	*0.7634*	*0.7298*	*0.7298*
	St_d: 0.0190	*0.0041*	*0.0102*	*0.0135*	*0.0176*	*0.0176*
Waveform	*J_mean: 0.821*	*0.821*	*0.788*	*0.765*	*0.7838*	*0.782*
	St_d: 0.0148	*0.0148*	*0.0119*	*0.0063*	*0.0146*	*0.0171*
Ionosphera	*J_mean: 0.904*	*0.906*	*0.890*	*0.856*	*0.889*	*0.8894*
	St_d: 0.0087	*0.0063*	*0.0101*	*0.0187*	*0.0158*	*0.0146*
Vehicle	*J_mean: 0.899*	*0.902*	*0.884*	*0.838*	*0.886*	*0.872*
	St_d: 0.0157	*0.0268*	*0.0257*	*0.0377*	*0.0202*	*0.0248*
Wincosin	*J_mean: 0.93*	*0.926*	*0.918*	*0.901*	*0.906*	*0.902*
	St_d: 0.0096	*0.0155*	*0.0109*	*0.0058*	*0.0071*	*0.0035*
German	*J_mean: 0.927*	*0.927*	*0.926*	*0.925*	*0.863*	*0.856*
	St_d: 0.0013	*0.0018*	*0.0025*	*0.0021*	*0.0057*	*0.0038*

on a Pentium IV 2.4 GHz PC using the BORLAND DELPHI compiler (version 5.0).

Table 1 shows the results obtained with the three metaheuristic strategies proposed in this work – GRASP, Tabu Search and Memetic Algorithm – compared to the results obtained with the Genetic Algorithm and floating search methods.

Different values of d have been considered, specifically from $d = 3$ to $d = 7$. For each database and each value of d, the mean value (*J_mean*) and the standard deviation (*St_d*) of the objective function J in test sets are shown. The best solutions for each database and each value of d are written in bold-italics.

Table 1 shows that GRASP yields the best solutions in 23 cases out of 30 and tabu search in 21 cases. Thus, these two strategies obtain better results than the Memetic Algorithm, Genetic Algorithm, SFFS, and SBFS. In general, among the remaining strategies, MA obtains the best results, although the differences are not significant in most cases.

Fig. 1 shows the solutions obtained with GRASP, Tabu Search, Memetic Algorithm, Genetic Algorithm, SFFS and SBFS for the Spam database.

Fig. 1 very clearly shows how solutions obtained with GRASP, Tabu Search and Memetic Algorithm dominate the solutions obtained by the other methods.

Table 2 shows the mean value, *J_mean*, and standard deviation, *St_d*, for each database and for each strategy, and Table 3 shows the mean value and standard deviation of all the databases in conjunction for each strategy.

The main conclusion derived from Tables 2 and 3 is basically the same as that derived from Table 1; that is, GRASP and Tabu Search obtain similar results overall and obtain the best results compared to MA, GA and floating search methods. The GA obtains the worst results but these are very similar to the results obtained with SFFS and SBFS.

Table 3
Results for all databases in conjunction and for each strategy.

Database	GRASP	Tabu Search	Memetic Algorithm	Genetic Algorithm	SFFS	SBFS
J_mean	0.888	0.885	0.867	0.841	0.843	0.838
St_d	0.0429	0.0467	0.0573	0.0652	0.0666	0.0645

Table 4
t-Test for every database (proposed methods versus GA).

Database	GRASP–Genetic A.		Tabu S.–Genetic A.		Memetic A.–Genetic A.	
	t-Value	Prob.	*t*-Value	Prob.	*t*-Value	Prob.
Spambase	10.534	***0.000***	4.703	***0.000***	6.517	***0.003***
Waveform	8.227	***0.000***	8.227	***0.000***	4.509	0.011
Ionosphera	6.763	***0.000***	3.291	***0.002***	3.492	0.025
Vehicle	6.465	***0.000***	3.679	***0.001***	5.496	***0.005***
Wincosin	3.678	***0.001***	3.143	***0.005***	2.302	0.030
German	1.871	0.135	0.612	0.544	1.5	0.208

Table 5
t-Test for every database (proposed methods versus SFFS).

Database	GRASP–SFFS		Tabu S.–SFFS		Memetic A.–SFFS	
	t-Value	Prob.	*t*-Value	Prob.	*t*-Value	Prob.
Spambase	6.517	***0.000***	15.679	***0.000***	14.409	***0.000***
Waveform	12.461	***0.000***	12.461	***0.000***	0.461	0.668
Ionosphera	1.914	0.128	2.606	0.060	0.141	0.895
Vehicle	3.828	***0.004***	3.802	***0.005***	−0.159	0.882
Wincosin	5.075	***0.005***	2.348	0.079	2.318	0.081
German	23.064	***0.000***	23.579	***0.000***	23.990	0.000

In support of the previous conclusion, Tables 4–6 show the results of the *t*-test (*t*-value and probability) obtained by comparing GRASP, Tabu Search and the Memetic Algorithm with the Genetic Algorithm, SFFS and SBFS for each database.

Finally, Table 7 shows the results of the *t*-test for all the databases in conjunction.

Table 7 shows that for all databases in conjunction the proposed metaheuristic strategies – GRASP, Tabu Search and the Memetic Algorithm – obtain the best results, that is, the differences between these strategies and the Genetic Algorithm, SFFS and SBFS are significant with zero probability.

Tables 4–6 show that the differences between the Memetic Algorithm and the Genetic Algorithm, SFFS and SBFS are not significant for most of the databases. Thus, the Memetic Algorithm obtains worse results than GRASP and Tabu Search. We can also see that for the Ionosphere database GRASP and Tabu Search obtain poor results (no significant differences exist compared to the other strategies) and for the German database the Genetic Algorithm obtains good results (the differences between the Genetic Algorithm and the other metaheuristic strategies are not significant).

5.2. Application: selection of financial ratios to predict corporate failure

The feature selection problem is applied to select financial ratios for predicting corporate failure.

The discovery of knowledge in business data is an important task capable of providing significant competitive advantages for a business organization by exploiting the potential of large databases. Data mining has been applied to various business domains, such as marketing, finance, banking, manufacturing and telecommunications (Branchman et al., 1996). Classification is one of the important issues in many business applications. Typical examples of business classification problems include credit approval, securities trading, product selection, risk estimation, personnel selection and corporate bankruptcy.

Corporate bankruptcy triggers economic losses for management stockholders, employees, customers and others, together with a great social and economic cost to the nation. Thus, the accurate prediction of bankruptcy is a critical issue in finance.

The financial database used contains information about the value of a series of financial ratios for a set of Spanish firms. Specifically, there are 93 variables with continuous values (financial ratios), 17 108 cases (firms) and two classes (failed/healthy). The failed firms are firms which had failed in the period 2000–2003, whereas healthy firms are firms which went on with their normal activities in 2003.

As seen in the previous test, the GRASP and Tabu Search metaheuristic strategies obtained the best results, so we have only used these strategies. Different values of d have been considered, specifically from $d = 3$ to $d = 7$.

As in the experiments described in Section 5.1, a 1×10 cross-validation has been developed to measure and test the methods used. All the experiments were done on a Pentium IV 2.4 GHz PC using the BORLAND DELPHI compiler (version 5.0).

Table 6
t-Test for every database (proposed methods versus SBFS).

Database	GRASP–SBFS		Tabu S.–SBFS		Memetic A.–SBFS	
	t-Value	Prob.	*t*-Value	Prob.	*t*-Value	Prob.
Spambase	14.982	***0.000***	15.679	***0.000***	14.409	***0.000***
Waveform	14.895	***0.000***	14.895	***0.000***	0.605	0.578
Ionosphera	1.983	0.118	2.398	0.75	0.104	0.922
Vehicle	6.120	***0.004***	5.692	***0.005***	2.826	0.048
Wincosin	6.014	***0.004***	4.774	***0.006***	3.133	0.035
German	46.816	***0.000***	29.750	***0.000***	50.401	***0.000***

Table 7
t-Test for all databases in conjunction.

	t-Value	Prob.
GRASP–Genetic A.	8.302	***0.000***
Tabu Search–Genetic A.	8.395	***0.000***
Memetic A.–Genetic A.	7.147	***0.000***
GRASP–SFFS	6.350	***0.000***
Tabu Search–SFFS	6.998	***0.000***
Memetic A.–SFFS	4.045	***0.000***
GRASP–SBFS	7.410	***0.000***
Tabu Search–SFFS	8.208	***0.000***
Memetic A.–SBFS	5.074	***0.000***

Table 8
Results for the financial database.

d	GRASP	Tabu Search
3	*J_mean: 0.9* *St_d: 0.04*	*0.885* *0.056*
4	*J_mean: 0.897* *St_d: 0.054*	*0.911* *0.028*
5	*J_mean: 0.9* *St_d: 0.0266*	*0.911* *0.028*
6	*J_mean: 0.902* *St_d: 0.0289*	*0.895* *0.0368*
7	*J_mean: 0.902* *St_d: 0.0209*	*0.9* *0.0383*

Table 8 shows the results obtained.

Table 8 shows that by selecting only 3 features out of 93 the percentage of hits is 88.5%, and when 7 are selected the percentage is 90%. The number of features has been reduced considerably, but the degree of accuracy regarding predicting bankruptcy is very acceptable.

6. Conclusions

This work investigates the problem of feature selection. This is an NP-hard problem and so the use of metaheuristic strategies is very appropriate to solve it. Extensive research into feature selection has been carried out over the past four decades. The Genetic Algorithm is the metaheuristic strategy that has been used more times for solving this problem.

In the present work three metaheuristic strategies were proposed to solve the feature selection problem, specifically, GRASP, Tabu Search and the Memetic Algorithm. Computational experiments with different databases were performed to compare GRASP, Tabu Search and the Memetic Algorithm with a Genetic Algorithm and the SFFS and SBFS floating methods. Although neither of these floating methods can always guarantee providing the best subset of features, their performance has been found to be very good compared with other search methods. The results obtained show that GRASP and Tabu Search perform better than the rest of the strategies. Finally the feature selection problem is applied to select financial ratios for predicting corporate failure.

It should be noted that all the metaheuristics strategies compared are the simplest version of the strategies to avoid over-fitting. Thus, a trade-off between optimization and generalization is obtained.

Acknowledgements

The authors are grateful for financial support from the Spanish Ministry of Education and Science, FEDER grants (National R&D Plan, Projects SEJ2005-08923/ECON and SEJ 2004-08176-02-01/ECON) and from the Regional Government of Castilla y León, Spain (Conserjería de Educación, Project BU008A06).

References

Anderson, J.R., Matessa, M., 1992. Explorations of an incremental, Bayesian algorithm for categorization. Machine Learn. 9, 275–308.
Backer, E., Shipper, J.A., 1977. On the max–min approach for feature ordering and selection. In: Proceedings of the Seminar on Pattern Recognition, Liege.
Baek, H.C., Hwanjo, Y., Kwang-Won, K., Tae, H.K., In Young, K., SunI, K., 2008. Application of irregular and unbalanced data to predict diabetic nephropathy using visualization and feature selection methods. Artif. Intell. Med. 42 (1), 37–53.
Bala, J., Dejong, K., Huang, J., Vafaie, H., Wechsler, H., 1996. Using learning to facilitate the evolution of features for recognizing visual concepts. Evolution. Comput. 4 (3), 297–311.
Branchman, R.J., Khabaza, T., Kloesgen, W., Piaterky-Shpiro, G., Simoudis, E., 1996. Mining business databases. Comm. ACM 39 (11), 42–48.
Cotta, C., Sloper, C., Moscato, P., 2004. Evolutionary search of thresholds for robust feature set selection: Application to the analysis of microarray data. Lect. Notes Comput. Sci. 3005, 21–30.
Cover, T.M., Van Campenhout, J.M., 1997. On the possible orderings in the measurement selection problem. IEEE Trans. Syst. Man Cybernet. 7 (9), 657–661.
Devijver, P., Kittler, J., 1982. Pattern Recognition: A Statistical Approach. Prentice Hall.
Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., 1996. Advances in Knowledge Discovery and Data Mining. AAAI Press, Menlo Park, CA.
Feo, T.A., Resende, M.G.C., 1995. Greedy randomized adaptive search procedures. J. Global Optim. 2, 1–27.
Fukunaga, K., Koontz, W.L.G., 1970. Application of the Karhunen–Loève expansion to feature selection and ordering. IEEE Trans. Comput. 19, 311.
Ganster, H., Pinz, A., Rohrer, R., Wildling, E., Binder, M., y Kittler, H., 2001. Automated melanoma recognition. IEEE Trans. Med. Imag. 20 (3), 233–239.
Glover, F., 1989. Tabu Search: Part I. ORSA J. Comput. 1, 190–206.
Glover, F., 1990. Tabu Search: Part II. ORSA J. Comput. 2, 4–32.
Glover, F., Laguna, M., 1997. Tabu Search. Kluwer Academic Publishers, Boston.
Glover, F., Laguna, M., 2002. Tabu Search. In: Pardalos, P.M., Resende, M.G.C. (Eds.), Handbook of Applied Optimization. Oxford University Press, pp. 194–208.
Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M., Bloomfield, C., Lander, E., 1999. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. Science 286 (5439), 531–537.
Guyon, I., Elisseeff, A., 2003. An introduction to variable and feature selection. J. Mach. Learn Res. 3, 1157–1182.
Inza, I., Larrañaga, P., Etxeberria, R., y Sierra, B., 2000. Feature subset selection by bayesian networks based optimization. Artif. Intell. 123, 157–184.
Inza, I., Merino, M., Larranaga, P., Quiroga, J., Sierra, B., y Girala, M., 2001a. Feature subset selection by genetic algorithms and estimation of distribution algorithms – A case study in the survival of cirrhotic patients treated with TIPS. Artif. Intell. Med. 23 (2), 187–205.
Inza, I., Larranaga, P., y Sierra, B., 2001b. Feature subset selection by Bayesian networks: A comparison with genetic and sequential algorithms. Internat. J. Approximate Reasoning 27 (2), 143–164.
Jain, A.K., Vailaya, A., 1996. Image retrieval using colour and shape. Pattern Recognition 29 (8), 1233–1244.
Jain, A.K., Zongker, D., 1997. Feature selection: Evaluation, application and small sample performance. IEEE Trans. Pattern Anal. Mach. Intell. 19, 153–158.
Jourdan, L., Dhaenens, C., Talbi, E., 2001. A Genetic algorithm for feature subset selection in data-mining for genetics. In: Proc. 4th Metaheuristics Internat. Conf., MIC 2001, pp. 29–34.
Kira, K., Rendell, L.A., 1992. A practical approach to feature selection. In: Proc. 9th Internat. Conf. on Machine Learning, pp. 249–256.
Kittler, J., 1978. Feature set search algorithms. In: Chen, C.H. (Ed.), Pattern Recognition and Signal Processing. Sijthoff and Noordhoff, Alphen aan den Rijn, The Netherlands, pp. 41–60.
Kohavi, R., 1995. Wrappers for Performance Enhancement and Oblivious Decision Graphs. Stanford University, Computer Science Department.
Kohavi, R., John, G., 1997. Wrapper for feature selection. Artif. Intell. 97 (1–2), 273–324.
Lanzi, P., 1997. Fast feature selection with genetic algorithms: A filter approach. In: Proc. IEEE Internat. Conf. on Evolutionary Computation, pp. 537–540.
Lee, S., Yang, J., y Oh, K.W., 2003. Prediction of molecular bioactivity for drug design using a decision tree algorithm. Lect. Notes Artif. Intell. 2843, 344–351.
Liu, H., Motoda, H., 1998. Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic, Boston.
Marill, T., Green, D.M., 1963. On the effectiveness of receptors in recognition system. IEEE Trans. Inform. Theory 9, 11–17.
Moscato, P., 2002. Memetic algorithms. In: Pardalos, P.M., Resende, M.G.C. (Eds.), Handbook of Applied Optimization. Oxford University Press, pp. 157–167.
Moscato, P., Cotta, C., 2003. A Gentle Introduction to Memetic Algorithms, Handbook of Metaheuristics. Kluwer, pp. 105–144.
Murphy, P.M., Aha, D.W., 1994. UCI repository of Machine Learning. University of California, Department of Information and Computer Science, <http://www.ics.uci.edu/~mlearn/MLRepository.html>.

Narendra, P.M., Fukunaga, K., 1977. A Branch and Bound Algorithm for Feature Subset Selection. IEEE Trans. Comput. 26 (9), 917–922.
Oliveira, L.S., Sabourin, R., Bortolozzi, F., y Otros, 2003. A methodology for feature selection using multiobjective genetic algorithms for handwritten digit string recognition. Internat. J. Pattern Recognition Artif. Intell. 17(6), 903–929.
Pudil, P., Novovicova, J., Kittler, J., 1994a. Floating search methods in feature selection. Pattern Recognition Lett. 15, 1119–1125.
Pudil, P., Ferri, F., Novovicova, J., Kittler, J., 1994b. Floating search methods for feature selection with nonmonotonic criterion functions. In: Proc. 12th IEEE Conf. on Pattern Recognition. Jerusalem, Israel, pp. 279–283.
Punch, W.F., Goodman, E.D., Pei, M., Chia-Shun, L., Hovland, P. Enbody, R., 1993. Further research on feature selection and classification using genetic algorithms. In: Proc. 5th Internat. Conf. Genetic Algorithms, pp. 557–564.
Ramaswamy, S., Tamayo, P., Rifkin, R., Mukherjee, S., Yeang, C., Angelo, M., Ladd, C., Reich, M., Latulippe, E., Mesirov, J., Poggio, T., Gerald, W., Loda, M., Lander, E., Golub, T., 2001. Multiclass cancer diagnosis using tumor gene expression signatures. Proc. Natl. Acad. Sci. USA 98 (26), 15149–15154.
Reunanen, J., 2003. Overfitting in making comparisons between variable selection methods. J. Mach. Learn Res. 3, 1371–1382.
Shin, K.S., Lee, Y.J., 2002. A genetic algorithm applications in bankruptcy prediction modelling. Expert Systems Appl. 23 (3), 321–328.
Shy, S., Suganthan, P.N., 2003. Feature analysis and classification of protein secondary structure data. Lect. Notes Comput. Sci. 2714, 1151–1158.
Siedlecki, W., Sklansky, J., 1988. On automatic feature selection. Internat. J. Pattern Recognition Artif. Intell. 2 (2), 197–220.
Siedlecki, W., Sklansky, J., 1989. A note on genetic algorithms for large-scale feature selection. Pattern Recognition Lett. 10, 335–347.
Sierra, B., Lazkano, E., Inza, I., Merino, M., Larrañaga, P., Quiroga, J., 2001. Prototype selection and feature subset selection by estimation of distribution algorithms. A case study in the survival of cirrhotic patients treated with TIPS. Lect. Notes Artif. Intell. 2101, 20–29.
Solberg, A.H.S., Jain, A.K., 1995. A study of the invariance properties of textural features. In: Proc. IGARS Conf., Florence (Italy), pp. 670–672.
Stearns, S.D., 1976. On selecting features for pattern classifiers. In: 3rd Internat. Conf. on Pattern Recognition, Coronado, CA, pp. 71–75.
Tamoto, E., Tada, M., Murakawa, K., Takada, M., Shindo, G., Teramoto, K., Matsunaga, A., Komuro, K., Kanai, M., Kawakami, A., Fujiwara, Y., Kobayashi, N., Shirata, K., Nishimura, N., Okushiba, S.I., Kondo, S., Hamada, J., Yoshiki, T., Moriuchi, T., y Katoh, H., 2004. Gene expression profile changes correlated with tumor progression and lymph node metastasis in esophageal cancer. Clin. Cancer Res. 10 (11), 3629–3638.
Whitney, A.W., 1971. A direct method of nonparametric measurement selection. IEEE Trans. Comput. 20, 1100–1103.
Wong, M.L.D., Nandi, A.K., 2004. Automatic digital modulation recognition using artificial neural network and genetic algorithm. Signal Process. 84 (2), 351–365.
Yang, J., Honavar, V., 1997. Feature subset selection using a genetic algorithm genetic programming 1997. In: Proc. 2nd Annual Conf. Morgan Kaufmann.