BRIEFINGS IN BIOINFORMATICS. VOL 11. NO 1. 127–141
Advance Access published on 29 September 2009
doi:10.1093/bib/bbp035

Advances in metaheuristics for gene selection and classification of microarray data

Béatrice Duval and Jin-Kao Hao

Submitted: 5th May 2009; Received (in revised form): 12th July 2009

Abstract

Gene selection aims at identifying a (small) subset of informative genes from the initial data in order to obtain high predictive accuracy for classification. Gene selection can be considered as a combinatorial search problem and thus be conveniently handled with optimization methods. In this article, we summarize some recent developments of using metaheuristic-based methods within an embedded approach for gene selection. In particular, we put forward the importance and usefulness of integrating problem-specific knowledge into the search operators of such a method. To illustrate the point, we explain how ranking coefficients of a linear classifier such as support vector machine (SVM) can be profitably used to reinforce the search efficiency of Local Search and Evolutionary Search metaheuristic algorithms for gene selection and classification.

Keywords: *microarray data analysis; gene selection; classification; local search; genetic algorithm; memetic algorithm*

INTRODUCTION

DNA microarray technology is a revolutionary method enabling the measurement of expression levels of thousands of genes in a single experiment under diverse experimental conditions. Since its invention, this technology has proved to be a valuable tool for many biological and medical applications [1].

Microarray data analysis can be carried out according to at least two different and complementary perspectives. On the one hand, data clustering (nonsupervised classification) aims to identify groups of genes, or groups of experimental conditions, that exhibit similar expression patterns. In such a context, bi-clustering is particularly interesting since it allows the simultaneous identification of groups of genes that show similar expression patterns across specific groups of experimental conditions (samples) [2, 3].

On the other hand, researchers on cancer studies are interested in categorical phenotypes like cancer occurrences, specific tumor subtypes or cancer survivals, which naturally leads to supervised classification of the data. Supervised classification (also called class prediction or class discrimination) aims to assign samples to predefined categories. This article will focus on this specific task.

The aim of supervised classification is 2-fold. The first is to build accurate classifiers that enable the reliable discrimination between different phenotype classes. Machine learning [4] and pattern recognition studies [5] offer a great number of algorithms that can be adapted for classification of microarray datasets. We can cite, for example, k-nearest neighbor (kNN) classifiers, neural networks and support vector machines (SVM).

The biologists are not only interested in accurate predictive tools, they also need to identify biomarkers of diseases, i.e. a small set of relevant genes that leads to the correct discrimination between different biological states. This second purpose of supervised classification can be achieved by classifiers that provide understandable results and indicate which genes contribute to the discrimination.

Corresponding author: Jin-Kao Hao, University of Angers, 2 Boulevard Lavoisier, 49045 Angers Cedex 01, France. Tel: +33-241-735076; Fax: +33-241-735073, E-mail: hao@info.univ-angers.fr

Béatrice Duval is an associate professor in Computer Science. Her research field concerns data mining and machine learning, and for some years, she has been working on applications in bioinformatics.

Jin-Kao Hao is a Computer Science professor. His research lies in the design of effective heuristic and metaheuristic algorithms for solving large-scale combinatorial search problems. He is interested in various application areas including bioinformatics.

For microarray data, understandable results can only be possible if the very high dimensionality of the data is reduced by a process of *gene selection*. This is a special case of attribute selection, a well-known problem in machine learning (see [6, 7] for a general introduction to this topic). A microarray data analysis typically entails several thousands of genes whereas only a few dozen of samples are available. When faced with such dimensions, most learning algorithms may exploit chance patterns and elaborate models that perform well on training data but poorly on new data. This risk of overfitting must be reduced by selecting a number of genes comparable with the number of samples. Moreover, the selection of a smaller number of attributes requires less-computational efforts for model learning and enables a better understanding of the process that underlies the data.

Depending on how the selection process is combined with the classification process, attribute selection methods belong to one of the following three categories: filter methods, wrapper methods and embedded methods [8].

Filter methods achieve attribute selection independently of the classification model. They consider only the data and rely on a ranking of the attributes according to their correlation with the class label with the aim to identify genes that are differentially expressed in the different phenotypes considered. Many filter methods are univariate since they evaluate each attribute independently of the others, by means of statistical measures like *t*-test, BW ratio [9] or information theoretic criteria like mutual information.

These methods are computationally efficient, but they fail to deal with redundancy between the selected genes. Moreover, they do not take into account complementary genes that have individually a poor predictive performance, but jointly enable a good classification. Finally, it is known that there may be no agreement between the different measures, leading to contradictory rankings [10].

In *wrapper methods*, selection of relevant attributes is performed in interaction with a classifier [6]. The problem is no longer to rank the variables according to their predictive power, but to find a gene subset that achieves the best performance for a particular learning model. To explore the space of attribute subsets, a search algorithm is 'wrapped' around the classification model which is used as a black box to assess the predictive quality of the candidate gene subsets. This problem of subset selection is known to be NP-hard [11], so heuristic methods are often used to select the best possible gene subset.

Wrapper methods explore the possible subsets by deterministic greedy heuristics, like backward elimination and forward selection, or by randomized search algorithms, like local search and genetic algorithms that will be further described in the article. Compared with filter methods, one main advantage of wrappers is that they consider interactions between genes, and consequently are able to deal with redundancy. Their main drawback is the high-computational cost, since a classifier must be trained to assess the quality of each candidate subset.

Embedded methods are similar to wrapper methods in the sense that the search of an optimal subset is performed for a specific learning algorithm, but they are characterized by a deeper interaction between gene selection and classifier construction. Two main different approaches have been devised.

In the first approach, the classifier provides information to guide the exploration of candidate subsets. This kind of interaction is illustrated by Recursive Feature Elimination (RFE) methods, of which SVM-RFE is the most famous illustration [12]. These methods begin with all the genes and recursively eliminate one (or several) genes. To choose which gene can be discarded, a classifier is trained and the gene the removal of which has the least influence on its objective function (the objective or cost function of a classifier evaluates the risk of misclassification. It is approximated by a cost function calculated on the training examples.) is eliminated; the process is iterated with learning of a new classifier at each step.

Penalized methods form another family of embedded methods where attribute selection is integrated in the process of classifier construction by adding to the objective function of the classifier a penalty function that takes into account the number of attributes (complexity of the model). Ma and Huang [13] give a review of penalized methods recently proposed in bioinformatics.

In this article, we explain how metaheuristic methods like local search, genetic and memetic algorithms (MAs) can be designed to deal with gene selection and classification of microarray data. According to the typology of attribute selection methods, the methods presented in this article are clearly embedded ones since they realize a tight interaction between the process of selection and the classification model. Moreover, they belong

to the first kind of embedded methods since the exploration of candidate subsets is guided by information provided by the classifier.

CLASSIFICATION ELEMENTS FOR EMBEDDED METHODS

Before presenting the general framework of metaheuristic search methods for gene selection and classification of microarray data, this section recalls some fundamental elements of supervised classification that are key points in embedded methods. We deal with the important question of performance evaluation of selection methods, which must be treated with care owing to the data dimensionality. We also provide a brief presentation of linear SVM, which is necessary to illustrate how information provided by a classifier can interact with a metaheuristic-based selection process.

Validation protocol for gene selection

The importance of a rigorous estimation of the predictive accuracy of a classification model is a well-known problem in machine learning. When the labeled samples are scarce, which is the case for microarray data, the estimation of prediction accuracy can be realized via cross-validation. In the k-fold cross-validation protocol, the initial dataset D is split into k subsets of approximately the same size $D_1, \ldots, D_k$. The learning algorithm is applied k times to build k classifiers: in step i, the data subset D_i is left out as a test set, the classifier is induced from the training dataset $D - D_i$ and its accuracy Acc_i is estimated on D_i. The accuracy estimate computed by k-fold cross-validation is then the mean of the Acc_i, for $1 \leq i \leq k$. When the number of folds (iterations) is equal to the number of initial samples, the so-called leave-one-out cross-validation (LOCCV) protocol provides an unbiased estimate of the generalization accuracy. Empirical studies [14, 15] have shown that 10-fold cross-validation is a good choice to obtain an almost unbiased estimate, with small variance and reasonable computational time.

When dealing with gene selection, the quality of a selected gene subset is assessed by its capability to lead to an efficient classifier. Therefore, to validate the results of a selection method, we have to use a schema that evaluates both the selection process and the classification model.

In order to avoid the *selection bias* that overestimates predictive accuracy, [16–19] have pointed out that it is important to include gene selection into the cross-validation schema. Selection bias occurs when the accuracy of a model is assessed on samples that play a role in the construction of the model. Therefore, selecting a subset of genes on the entire dataset and then performing cross-validation to estimate a classifier model is a biased protocol. In a correct experiment design, the dataset must be split before gene selection is achieved: each step of cross-validation performs gene selection and classification, as described in figure 1. Let us notice that when model learning also includes data preprocessing and preselection, these steps must also be included into the cross-validation loop.

Ranking coefficients from linear classifiers

Binary classification can be viewed as the task of separating classes in the attribute space and different types of decision boundaries can be explored. For a linear classifier, the decision boundary is an hyperplane determined by a linear combination of the input attributes.

Formally, we consider a training set of n samples belonging to two classes; each sample is noted $\{X_i, Y_i\}$ where $\{X_i\}$ is the vector of dimension m of attribute values describing the sample and Y_i the class label. The hyperplane is thus represented by a vector $w = (w_i)$ and a constant b and the classification of a sample X is decided according to the sign of the function $f(X) = w \cdot X + b$. According to this equation, the inputs that are weighted by the largest (positive or negative) values have a greater influence on the decision. Consequently, the coefficients of

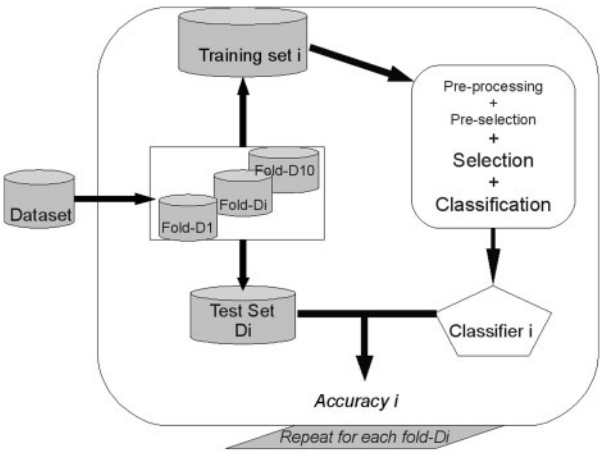

Figure 1: Cross-validation schema for gene selection and classification.

vector w are ranking coefficients about the relevance of each attribute. Among the most famous linear classifiers, we find the Fisher's linear discriminant analysis (LDA) and linear SVMs that we describe now with more details.

Linear SVM

SVMs are powerful classification algorithms [20, 21]. In computational biology, they have been applied with success to problems like protein remote homology detection, functional classification of promoter regions, prediction of protein–protein interactions, etc., see [22] for a detailed review.

SVM rely on two key ideas. The first is that SVM algorithms compute stable classifiers by searching for a decision boundary that has the maximum margin with the examples. The second is that complex decision boundaries can be computed by using linear machines in a high-dimensional feature space, implicitly represented by a kernel function. Our presentation will focus on linear SVMs because they are known to be well suited to datasets of high dimensionality [12, 23, 24] and they offer a clear biological interpretation of the results.

For a given training set of labeled samples, a linear SVM determines an optimal hyperplane that divides the positively and negatively labeled samples with the maximum margin of separation. A noteworthy property of SVM is that the hyperplane depends only on a small number of training examples called the support vectors, they are the closest training examples to the decision boundary and they determine the margin. When the samples cannot be linearly separated, slack variables ξ_i are added to allow misclassification of difficult or noisy examples, leading to a soft margin SVM.

A soft-margin linear SVM classifier aims at solving the following optimization problem:

$$\min_{w,b,\xi_i} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \qquad (1)$$

subject to $Y_i(w \cdot X_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$, $i = 1, \ldots, n$.

In this formulation, w is the vector that determines the separating hyperplane; C is a given penalty term that controls the cost of misclassification errors. To solve this optimization problem, it is convenient to consider the dual formulation [20]:

$$\min_{\alpha_i} \frac{1}{2}\sum_{i=1}^{n}\sum_{l=1}^{n}\alpha_i\alpha_l Y_i Y_l X_i \cdot X_l - \sum_{i=1}^{n}\alpha_i \qquad (2)$$

subject to $\sum_{i=1}^{n} Y_i\alpha_i = 0$ *and* $0 \leq \alpha_i \leq C$.

The decision function of the linear SVM classifier for an input vector X is given by $f(X) = w \cdot X + b$ with $w = \sum_{i=1}^{n}\alpha_i Y_i X_i$ and $b = Y_i - w \cdot X_i$. The vector w is a linear combination of the training samples. Most coefficients α_i are zero and the training samples with nonzero coefficients are the support vectors. Moreover, the maximum margin M is given by

$$M = \frac{2}{\|w\|} \qquad (3)$$

As stated before, the vector w, and consequently the orientation of the hyperplane, gives information about the relevance of each attribute. If the plane is orthogonal to a particular dimension, then that attribute is informative, and vice versa. Therefore, given the vector w of a linear SVM, we define a ranking coefficient vector c by:

$$c_j = \left(w_j\right)^2 \; j = 1, \ldots, m \qquad (4)$$

For each of the m attributes, c_j measures the relevance of this attribute for the classification problem, with the smallest values for the least relevant ones.

GENE SELECTION AND METAHEURISTICS

In this section, we show that gene selection can be casted as a combinatorial search problem, and consequently be handled by a class of modern optimization methods called metaheuristics.

Gene selection and combinatorial search

A combinatorial search problem (more precisely, a problem instance) is typically characterized by a pair (S, f), where S is the search space composed of the candidate solutions (or configurations) and f a cost (or objective) function to be maximized (or minimized). The problem is then to find a best solution $I^* \in S$ such that $f(I^*) \geq f(I)$ for any element $I \in S$ [$f(I^*) \leq f(I)$ for a minimization problem]. Such a solution I^* is called an optimal solution or a global optimum.

Gene selection can be considered from such a combinatorial search perspective. Indeed, given a set Ge of k candidate genes, the primary goal is to seek, among the possible subsets of Ge, a particular (small) subset G^* such that G^* leads to the highest prediction accuracy. One easily notes that the search space is the set of all the possible subsets. Clearly,

this space is highly combinatorial since there are 2^k possible subsets. So it will be illusory to try to carry out an exhaustive enumeration of all the possible candidate solutions for any reasonable value of k, say, >20. Consequently, instead of seeking the optimal subset by going through the whole search space, one can seek the best possible subset by exploring, in a selective way, a limited number of good candidate subsets. To fulfill this task, *metaheuristics* constitute a class of prominent methods [25], which offers different strategies to explore a large combinatorial search space.

Metaheuristics can be roughly classified into two large categories: neighborhood-based local search methods [26] and population-based evolutionary algorithms [27]. Representative local search methods include Simulated Annealing and Tabu Search (TS), while Genetic Algorithms are a well-known example of evolutionary methods.

Common characteristics of metaheuristics

Despite the apparent differences among different metaheuristics, they share a number of common characteristics concerning in particular the following two elements:

- *Encoding*: the way of representing the candidate solutions of the search space.
- *Fitness evaluation*: the way of measuring the quality of the candidate solutions.

We discuss here these issues in the context of gene selection. In later sections, we show how they are integrated in different metaheuristic-based algorithms for gene selection and classification.

Gene subset representation—encoding

To apply a metaheuristic search algorithm, one needs first to find out a way to represent or encode the candidate solutions of the search space. For gene selection, each gene can be represented by a binary variable such that the variable takes value 1 or 0 according to whether the gene is selected or not. Using this simple encoding, any candidate gene subset can be conveniently represented by a binary vector.

More precisely, given an initial set of k genes $Ge = \{g_1, \ldots, g_k\}$, a subset $G \subset Ge$ can be identified by the binary k-vector $I^G = [I_1^G, \ldots, I_k^G]$ such that $I_i^G = 1$ $(i = 1, \ldots, k)$, if gene $g_i \in G$ (g_i is selected), $I_i^G = 0$ otherwise. For instance, $I^G = [1,0,0,1]$ designates a subset of two genes among a set of four genes: the first and last genes are selected while the second and third ones are not selected. This representation of gene subset is widely employed in the literature in many search algorithms, in particular in genetic algorithms (see Section 7).

This basic encoding can be extended by including additional and problem-specific information. One possibility concerns gene ranking information when a linear classifier is used for prediction estimation. Such an augmented subset representation has the advantage of enabling the design of specialized and effective search operators and algorithms.

In Section 2.2, we have explained that a linear SVM provides, for each gene, a positive coefficient that evaluates the importance of this gene in the decision function. This ranking information can be used to complement the binary k-vector.

Formally, given I^G a binary vector representing a candidate gene subset G (call it *gene subset vector*) and I^C a real-valued vector representing the gene ranking coefficients defined by Equation (4) (call it *ranking coefficient vector*), we define $I = < I^G, I^C >$ to be our augmented representation of gene subset G. Let us notice that for each nonselected gene ($I_i^G = 0$) the corresponding coefficient in I^C is null ($I_i^C = 0$). Throughout this article, we will employ this augmented representation.

Gene subset evaluation—fitness function

To explore the given search space, a metaheuristic search algorithm needs a fitness function to evaluate the quality of each candidate solution. This evaluation function introduces an order among the solutions of the search space, allowing thus the comparison of a pair of solutions. For many cases, the fitness function takes simply the form of the initial objective function.

For gene selection, the primary goal is to select a relevant gene subset leading to a high classification accuracy. Consequently, a first fitness function for a candidate subset can be defined by the prediction accuracy achieved by a given classifier built on this subset representation. Gene selection also aims to select small gene subsets. The previous fitness function can then be reinforced by a second objective minimizing the number of selected genes. The following fitness function f_1 takes into account

these objectives and combines them into a unique value

$$f_1(I) = \frac{\text{Acc}(I^G) + \left(1 - \frac{|I^G|}{k}\right)}{2} \quad (5)$$

In this formula, $\text{Acc}(I^G)$ $(0 \leq \text{Acc}(I^G) \leq 1)$ is the prediction accuracy of the classifier (see Section 2.1). Let us recall that k is the initial number of genes in the method. Consequently, k is the size of the encoding of an individual I whereas $|I^G|$ is the number of selected genes (the number of 1 in I^G).

We can consider other fitness functions according to the available information characterizing the quality of a gene subset. We have seen in Section 2.2 that a SVM computes a decision function with the greatest separation between the classes. This separation is measured by the margin M_{SVM} defined in Equation (3). A greater margin means a better separation between the classes and this element can be used to compare different selected subsets. More precisely, if two gene subsets enable class discrimination with the same accuracy, we consider the margin of each classifier to break this tie. This idea leads to the following fitness function f_2:

$$f_2(I) = < \text{Acc}(I^G), M_{\text{SVM}}(I^G) > \quad (6)$$

Given two candidate solutions I and J, it is possible to compare them: $f_2(I)$ is better than $f_2(J)$ if the following condition is satisfied: $(\text{Acc}(I^G) > \text{Acc}(J^G))$ or $(\text{Acc}(I^G) = \text{Acc}(J^G)$ and $M_{\text{SVM}}(I^G) > M_{\text{SVM}}(J^G))$.

This fitness function is particularly useful when a space of candidates with the same number of genes should be explored. In this case, the number of genes to be selected is given to the algorithm as a parameter, the goal of the search algorithm is then to identify the most pertinent gene subset of the desired size for classification.

Finally, since gene selection has two different objectives, methods such as evolutionary multiobjective optimization (EMO) algorithms constitute a natural and interesting solution approach [28], even if it is not exposed in this article.

Knowledge-based search operators

From a fundamental point of view, metaheuristics offer strategies and mechanisms for exploiting and exploring the given search space. The dual concept of 'exploitation and exploration' (also known under the term 'intensification and diversification') covers two fundamental and complementary aspects of any effective search. Exploitation emphasizes the ability of a method to examine in depth specific search areas, while exploration is the ability of a method to find promising search areas. A search method based solely on exploitation will confine the search in a limited area and fail to visit other areas. Similarly, a method relying heavily on exploration will lack capacity to examine in depth a given area and miss out the solutions of good quality. To be effective, a search method needs to conciliate exploitation and exploration. Different metaheuristics offer a variety of strategies and mechanisms to achieve this objective.

From a practical perspective, metaheuristics provide just a general optimization framework that can potentially be applied to various search problems. However, it should also be clear that a blind application of metaheuristics to a problem will not be able to lead to satisfactory solutions. To be effective, a metaheuristic must be carefully adapted to the given problem and integrate problem-specific knowledge within its search operators and strategies to ensure a good balance between exploration and exploitation. In what follows, we show how such an adaptation can be realized within TS and Evolutionary algorithms.

TS FOR GENE SELECTION

To illustrate how local search metaheuristics can be applied to the gene selection problem, we take the example of TS [29]. The described TS algorithm uses the solution encoding and a fitness evaluation function defined in Section 3.2. We explain below the general TS procedure as well as some other key components.

A brief review of TS

TS is an advanced metaheuristic designed for tackling hard combinatorial optimization problems. TS relies on a neighborhood relation as well as some forms of memory and learning strategy to explore effectively a search space.

Let (S, f) be our search problem where S and f are, the search space and optimization objective, respectively. A neighborhood N over S is any function that associates to each individual $I \in S$ some solutions $N(I) \subset S$. Any solution $I' \in N(I)$ is called a neighboring solution or simply a neighbor of I. For a given neighborhood N, a solution I is

a *local optimum* with respect to N if I is the best among the solutions in $N(I)$.

The notion of neighborhood can be explained in terms of *move* operator. Typically, applying a move mv to a solution I changes slightly I and leads to a neighboring solution I'. This transition from a solution to a neighbor is denoted by $I' = I \oplus mv$. Let $\Gamma(I)$ be the set of all possible moves which can be applied to I, then the neighborhood $N(I)$ of I can be defined by: $N(I) = \{I \oplus mv | mv \in \Gamma(I)\}$.

A typical TS algorithm begins with an initial configuration I in S and proceeds iteratively to visit a series of locally best configurations following the neighborhood. At each iteration, a *best* neighbor $I' \in N(I)$ is sought to replace the current configuration even if I' does not improve the current configuration in terms of the cost function. To avoid the problem of possible cycling and to allow the search to go beyond local optima, TS introduces the notion of *Tabu list*, one of the most important components of the method.

A tabu list is a special short-term memory that maintains a selective history H, composed of previously encountered solutions or more generally pertinent attributes (or moves) of such solutions. A simple TS strategy based on this short-term memory H consists in preventing solutions of H from being reconsidered for next tt iterations (tt, called tabu tenure, is problem dependent). Now, at each iteration, TS searches for a best neighbor from this dynamically modified neighborhood $N(H,s)$, instead of $N(s)$ itself. Such a strategy prevents Tabu from being trapped in short-term cycling and allows the search process to go beyond local optima.

When moves instead of solutions are recorded in tabu list, some nonvisited, yet interesting solutions may be prevented from being considered. *Aspiration criteria* may be used to overcome this problem. One widely used aspiration criterion consists of removing a tabu classification from a move when the move leads to a solution better than the best obtained so far. In Section 4.3, an example is provided.

In what follows, we show a TS algorithm adapted to the gene selection problem [30]. The TS algorithm uses the solution encoding and fitness evaluation function described in Section 3.2.

Move and neighborhood

Gene selection is a binary problem since each gene is represented by a binary variable to indicate whether it is selected or not. For such a problem, there are three commonly used move operators.

- *drop*: change the value of a single variable from 1 to 0,
- *add*: change the value of a single variable from 0 to 1,
- *swap*: swap the values of two variables that have different values.

For gene selection, we adopt the 'swap' operator and reinforce it with additional ranking information. The basic idea of this operator is to drop a mediocre gene g_i from a candidate solution and add another gene g_j. More precisely, let $I =< I^G, I^C >$ be a solution with $I^G = [I_1^G, I_2^G, \ldots, I_k^G]$ and $I^C = [I_1^C, I_2^C, \ldots, I_k^C]$, define:

- $i = \text{ArgMin}_j\{I_j^C | I_j^G = 1\}$, i.e. i identifies the gene g_i as the least relevant gene among the current selected genes.
- $O = \{j | I_j^G = 0\}$, i.e. O is the set of nonselected genes in the current solution I.

Then our move operator exchanges the gene g_i and any gene $g_j, j \in O$. In other words, one drops, from the current solution I, the least informative gene g_i (identified by the above index i) and adds a nonselected gene g_j ($j \in O$). This can be formally written as: $mv(i,j) = (g_i : 1 \to 0; g_j : 0 \to 1)$.

This neighborhood is very useful for exploring gene combinations of fixed size. In order to explore gene groups of variable sizes, one can simply change the number of fixed genes by adding or removing one or more genes in the group of selected genes. Once again, the decision on the gene(s) to be added or removed can be influenced by the ranking information.

Tabu list management

One critical issue of TS is the tabu list management: what to store in the tabu list T and how to define the tabu tenure tt. For the above 'swap' move operator that drops a gene g_i and adds a gene g_j, there are several ways to define the tabu list. First, each time a move $mv(i,j)$ is carried out, one forbids the strict reverse move, i.e. adding g_i and dropping g_j. In this case, the pair of indexes (i,j) is stored in the tabu list for the period of the tabu tenure. Second, one can record only the dropped gene g_i in T and forbid thus the reselection of g_i for the next tt iterations. Notice

that such a tabu list does not forbid the removal of the newly selected gene g_j during the same period. Third, it is also possible to store only the newly selected gene g_j, making it impossible to remove this gene for the next tt iterations.

The tabu tenure tt is a problem-dependent parameter. Two main techniques are used in practice. The tabu tenure can be static or dynamic. Once fixed, a static tabu tenure remains unchanged during the search. Typically, such a tabu tenure is determined using some preliminary experiments or defined as a function of problem-related information (e.g. the size of the considered problem instance). A dynamic tabu tenure is defined as a function of some changing information that can be collected during the search (e.g. the objective value, frequency of moves, etc.).

One notices that recording moves, instead of solutions (i.e. selected gene subsets), in the tabu list may occasionally prevent the search from reaching a good solution. Take the swap move as an example. Suppose one makes a move $mv(i,j)$ (i.e. dropping g_i and adding g_j) at iteration K and stores (i,j) in the tabu list for the next $tt = 10$ iterations. Consequently, the reverse move $mv(j,i)$ (i.e. reinserting g_i and dropping g_j) is not possible until iteration $K + 10$ due to its tabu status. However, a favorable situation may appear during this period after, say, five iterations such that making this move $mv(j,i)$ leads to the *best* gene subset even found. In this case, this move should be accepted to obtain the best solution, even if the move is classified as tabu. This is an application of the so-called aspiration criterion; the tabu status of a move is aspired or revoked.

Diversification

To be effective, TS needs specific strategies to diversify the search and avoid being trapped in local optima. We describe here a simple diversification technique based on the perturbation of a best solution.

More specifically, let I^* be the best local optimum recorded when TS begins to stagnate. The perturbation procedure takes I^* as its input and modifies I^* in some way to generate a new solution [31]. There are several possibilities to perturb a solution. The simplest one applies several times a move operator to the local optimum, leading to a randomly diversified new solution. For instance, one can simply remove x (x being a small integer) genes chosen randomly and add y other genes. A more effective strategy would use problem-specific knowledge to do controlled perturbations [32]. For instance, the removal and addition of genes can be conditioned with a probability proportional to the ranking coefficient of each gene.

Whatever the perturbation is applied, we obtain a new (diversified) solution, whereupon a new round of TS search procedure can be launched. Notice that the perturbation strength should be carefully controlled. If the perturbation is too strong, the search would resemble to a random restart. If the perturbation is too weak, the search would go easily back to the staring local optimum.

General TS procedure for gene selection

Algorithm 4.5 shows the general TS procedure for gene selection. It starts with an initial gene subset I_0 that can be generated randomly or provided by any other method. It then alternates an exploitation (intensification) phase (inner 'while' loop) and an exploration (diversification) phase (outer 'while' loop). The outer 'while' loop condition fixes the number of diversification phases wanted, while the inner 'while' loop condition indicates when the TS phase should be stopped to launch a diversification. Each time a new solution I^G (gene subset) is generated, the learning SVM model is called to assess the predictive accuracy of the gene subset and to obtain the ranking coefficient vector I^C. This vector is used by the move operator to favor the removal of the least informative gene.

Algorithm 1: TS for Gene Selection

```
1: Input: N and f - neighborhood and fitness
2: Output: I* - the best gene subset found
3: I0 = InitGenerate() (generate an initial solution - gene subset G)
4: I = I0 (I is the current solution)
5: Set tabu list to empty;
6: f(I) = fitnessEval(I);
7: I* = I (I* records the best gene subset found)
8: f* = f(I)
9: while Not Stop-Condition-1 do
10:    while Not Stop-Condition-2 do
11:       Choose a move mv(i, j) not forbidden by the tabu list;
12:       I = I ⊕ mv(i, j);
13:       Update tabu list;
14:       f(I) = fitnessEval(I);
15:       if f(I) is better than f* then
16:          I* = I (update the best solution ever found); f* = f(I);
17:       end if
18:    end while
19:    I = Perturbation(I*);
20:    Empty tabu list;
21: end while
```

GENETIC ALGORITHM FOR GENE SELECTION

In this section, we present a dedicated Genetic Algorithm for gene selection and classification as an example of evolutionary metaheuristics. Apart from the encoding and fitness functions already defined, we focus on two other key components: specialized crossover and mutation operators.

A brief review of genetic algorithm

Genetic algorithms mimic, in a very rudimentary way, some principles of Darwin's theory of evolution [33–35]. For instance, by sexual reproduction, a child inherits the genes from the parents (crossover). A child may have genes different from those of his parents because of alterations in genes (mutations). Many descendants can be produced, but only those individuals that are best adapted to the environment survive and transmit their genes to their offspring (natural selection). Chance plays a leading role to produce new individuals different from their parents.

As initially introduced by Holland [33], GAs code the solution space using a unique encoding in the form of binary string. Based on this universal encoding, a set of standard genetic operators are defined, including particularly crossover and mutation. For instance, one-point crossover takes as its input two existing solutions A and B (parents), cut A and B at a position j chosen randomly and exchanges the corresponding segments of A and B after position j. This leads naturally to two new solutions (offspring). Two-point and uniform crossovers are two other examples. Similarly, mutation operates on a given solution and changes probabilistically some values of the given solution.

In addition to crossover and mutation, selection decides which solutions will survive through generations. Typically, selection is stochastic and designed such that good solutions (as measured by a fitness function) have more chance to be conserved than bad solutions. Well-known selection methods include roulette wheel selection and tournament selection. Selection is also useful to determine parents for reproduction.

Despite the name, GAs are in fact more a general computational framework than a particular algorithm. Consequently, GAs should be carefully adapted to the given problem at hand. Even if some components of GAs such as selection and population management are problem-independent, most of other elements are to be tailored with problem-specific knowledge.

In what follows, we show how the GA can be adapted to the gene selection problem [36, 37]. The resulting GA algorithm uses the solution encoding and fitness evaluation function described in Section 3.2.

Dedicated crossover operator

Crossover is one of the key evolution operators for a GA and constitutes one leading force for exploring the search space. The basic idea of crossover is very appealing since this gives a way of generating new solutions from existing ones. In practice, instead of applying blind crossover, it is preferable to consider dedicated crossover operators such that they are able to ensure the heritage of good properties from parents to offspring. To illustrate the point, we describe here such a specialized crossover for gene selection.

Given the goal of selecting small subsets of predictive genes, the dedicated crossover operator will follow two fundamental principles: (i) to conserve the genes shared by the parents and (ii) to preserve 'high quality' genes from each parent even if they are not shared by both parents.

The notion of 'quality' of a gene here is defined by the corresponding ranking coefficient given by a linear classifier (e.g. SVM). Notice that applying the first principle will have as main effect of getting smaller and smaller gene subsets, while applying the second principle allows us to keep up good genes along the search process.

More formally, let $I =< I^G, I^C >$ and $J =< J^G, J^C >$ be two selected parents. The crossover operator combines I and J to obtain a single child $K =< K^G, K^C >$ by the following steps [36]:

(1) Extract the subset of genes shared by both parents by Boolean logic AND operator ($\otimes$) and arrange them in an intermediary gene subset vector F.

$$F = I^G \otimes J^G$$

(2) For the subset of genes obtained from the first step, extract the maximum coefficients $\max_I$ and $\max_J$ accordingly from their original ranking vectors I^C and J^C.

$$\max_I = \max\left\{I_i^C | i \textit{ such that } F_i = 1\right\}$$

and

$$\max_J = \max\left\{J_i^C | i \text{ such that } F_i = 1\right\}$$

(3) This step aims to transmit high-quality genes from each parents I and J which are not retained by the logic AND operator in the first step. These are genes with a ranking coefficient greater than $\max_I$ and $\max_J$. The genes selected from I and J are stored in two intermediary vectors AI and AJ

$$AI_i = \begin{cases} 1 & \text{if } I_i^G = 1 \text{ and } F_i = 0 \text{ and } I_i^C > \max_I \\ 0 & \text{otherwise} \end{cases}$$

and

$$AJ_i = \begin{cases} 1 & \text{if } J_i^G = 1 \text{ and } F_i = 0 \text{ and } J_i^C > \max_J \\ 0 & \text{otherwise} \end{cases}$$

(4) The gene subset vector K^G of the offspring K is then obtained by grouping all the genes of F, AI and AJ using the logical 'OR' operator ($\oplus$).

$$K^G = F \oplus AI \oplus AJ$$

The ranking coefficient vector K^C will be filled up when the individual K is evaluated by the SVM-based fitness function.

Note that another dedicated crossover operator for gene selection was presented in [37]. In this case, ranking information from Fisher's LDA is favorably used during the crossover operation to generate new solutions. The influence of these informed crossover operators is studied in [38] in comparison with random crossovers (one-point, uniform), showing a clear performance gain.

Dedicated mutation operator

Traditionally, mutation plays a role of diversification in the search. Conventional mutation flips randomly some variables. In the context of gene selection, a simple mutation operator will randomly add or drop some genes. An alternative mutation operator can be designed such that it eliminates some 'mediocre' genes and at the same time randomly adds other genes to keep some degree of diversity in the GA population.

To illustrate this point, we propose the following operator. Given an individual $I = < I^G, I^C >$, applying the mutation operator to I consists in carrying out the following steps.

(1) The first step calculates the average ranking coefficient of a selected gene in the individual I.

$$\bar{c} = \frac{\sum_{i=1}^{k} I_i^C}{|I^G|}$$

(2) The second step eliminates (with a probability) 'mediocre' genes (i.e. those whith a ranking coefficient inferior to the average) and for each deleted gene randomly introduces a new gene. $\forall I_i^G = 1$ and $I_i^C < \bar{c}$ $(i = 1, \ldots, k)$, mutate I_i^G with probability p^m. If a mutation does occur, take randomly a I_j^G such that $I_j^G = 0$ and set I_j^G to 1.

One notices that this mutation operator shares similarities with the 'swap' move operator used by the TS algorithm. Other dedicated mutation operators may be possible. For instance, one can design a mutation operator which removes 'mediocre' genes, but add no additional genes. Such a mutation, thus, plays the role of reducing the number of selected genes.

General GA procedure

The general GA procedure for gene selection is given in Algorithm 2. Let us recall that the initial set of genes Ge contains k genes. An initial population P is randomly generated such that the number of genes in each individual varies between k and $k/2$ genes. From this population, the fitness of each individual I is evaluated using the function defined in 'Common characteristics of metaheunistics' section.

To obtain a new population, a temporary population P' is used. To fill up P', the top *NbElit* individuals of P are first copied to P'. This process named *elitism* aims to keep the best individuals from one generation to another. The rest of P' is completed with solutions created by applying the dedicated crossover and mutation operators presented above. A selection method is applied to P to generate a pool of candidate solutions. From this pool, single or pairs of solutions are randomly chosen for mutation or crossover applications, new resulting solutions being inserted in P'. Once P' is filled up,

it replaces P to become the current population. The GA stops when a fixed number of generations is reached.

Algorithm 2: Genetic Algorithm for gene selection

```
1: Input: |P| - size of population
2:        NbElit - number of elite solutions to conserve
3:        maxGen - number of authorized generations
4: Output: G* - the best gene subset found
5: P = PopGenerate(|P|)
6: for each I ∈ P do f(I) = fitnessEval(P)
7: I* = best(P) (I* records the best gene subset found)
8: f* = f(I*)
9: Iter = 0
10: while Iter < maxGen do
11:   Generate the temporary population P'
12:   Copy the NbElit best individuals of P into P'
13:   Produce |P| − NbElit offspring by crossover and mutation
14:   Add the offspring to P'
15:   P = P'
16:   Iter = Iter + 1
17: end while
```

Additional comments: MA

From the above Genetic Algorithm and TS algorithm of 'TS for gene selection' section, it is possible to create a combined method by integrating the TS procedure algorithm within the GA, leading to a so-called MA [39]. From a general perspective, MA combines the genetic framework and local search framework [40]. MA proves to be quite successful in solving many hard combinatorial search problems. The purpose of a MA is to take advantage of the complementary nature of the GA and TS search methods. Indeed, it is generally believed that the GA framework offers more facilities for exploration, while neighborhood search has more capability for exploitation. Combining them may offer a better balance between exploitation and exploration which is highly desirable for an effective search.

More precisely, our MA is obtained by rewriting line 13 of Algorithm 2 such that each new solution I created by crossover is immediately improved by the TS procedure. In some sense, mutation is generally replaced by the TS procedure.

PERFORMANCE OF METAHEURISTICS FOR GENE SELECTION

As the main purpose of this article is to give a review of metaheuristic methods for gene selection, we do not present detailed computational results. Nevertheless, it is useful to give some indications about the performance of these approaches. For this purpose, we provide a summary of a set of publicly available datasets that have been the object of many studies. We then illustrate the range of computational performance that can be achieved with the metaheuristics on these datasets.

Datasets and computational performance of metaheuristic algorithms

Since the first publications about molecular classification of cancer [41, 42], several datasets have been studied and are publicly available, for example, on the Kent Ridge Biomedical repository (http://datam.i2r.a-star.edu.sg/datasets/krbd/). Table 1 gives a brief description of the datasets used in our experiments. For each dataset, we recall the number of genes and the number of samples, to illustrate the great challenge of such dimensionality. We also indicate the first paper that has proposed a detailed presentation and an analysis of this dataset. With the constant development of microarray technology, more datasets are now available and they provide some kind of benchmarks for the numerous methods of analysis that are proposed in the bioinformatics community. It is important to test a method on several datasets because special characteristics can be observed in some of them and class discrimination may be of different difficulty. For example, it is now well recognized that the two kinds of Leukemia studied in [41] can be easily discriminated even with a very small number of genes, while the Colon cancer dataset is more difficult, perhaps because it contains some mis-classified samples [43].

The computational performance of a metaheuristic algorithm depends first on the adaptation of the chosen metaheuristic to the target problem. As exposed in the previous sections, the adaptation concerns a careful design of the encoding, the fitness function, the neighborhood relations, the search operators like crossover and mutation. The different metaheuristic algorithms presented in this article have been assessed through extensive experiments on the above datasets [30, 36, 37, 39].

In these experiments, the 10-fold cross-validation protocol exposed in 'Classification elements for embedded methods' section is rigorously followed

Table I: Summary of datasets used for experimentation

Dataset	#Genes	#Samples	References
Colon Cancer	2000	62	[42]
Leukemia	7129	72	[41]
Breast Cancer	24481	97	[43]
Lung Cancer	12533	181	[44]
Prostate Cancer	12600	109	[45]
Ovarian Cancer	15154	253	[46]
CNS Cancer	7129	60	[47]
Lymphoma[47]	4026	47	[48]
Lymphoma[96]	4026	96	[48]

for the estimation of prediction accuracy of a metaheuristic algorithm. Computational results obtained from these experiments (with linear SVM and LDA classifiers) have been compared with those from some 14 most recent gene selection methods. The reference methods include, for example, ensemble machine learning methods (bootstrap, bagging) combined with neural networks, neuro-fuzzy methods, genetic algorithms, generalized discriminant analysis.

These comparisons have shown that the metaheuristic approach performs very well and competes quite favorably with the reference algorithms. Indeed, for all the nine datasets, the metaheuristic algorithms are able to reach the same or a better prediction accuracy (between 96% and 100%) with a small number of selected genes (often less than 20). For instance, one remarkable result from the metaheuristic approach concerns the Lymphoma[47] dataset. With the MA and the SVM classifier, the 10-fold cross-validation accuracy is 100% for an average number of genes of 5.7 with a SD of 2.16 (the average accuracy is 100% because each fold of cross-validation achieves this rate) [39]. Previously reported results are <98% with at least 20 genes [50, 51].

Exploration of gene combinations

Filter methods and greedy methods like RFE use a deterministic criterion to select genes without considering the relations among them. Consequently, if a gene is wrongly discarded, it can never be recovered to become a member of a relevant gene group. Contrary to these methods, the metaheuristic approach naturally explores a large number of gene combinations, increasing considerably the possibility of discovering interesting multiple gene groups.

From a practical point of view, multiple solutions may constitute valuable candidates for further biological investigations. In fact, it is well known that there may exist several or many relevant subsets that can lead to accurate classification of a microarray dataset [52]. Along the exploration by a heuristic search, one can obtain an archive of gene groups of good quality. Information contained in these solutions can be cross-validated with biological knowledge and profitably explored by the biologist.

Moreover, multiple gene subsets provided by the metaheuristic approach can be contrasted and combined with gene groups provided by other independent methods. This permits to identify frequently selected genes that would be good candidates for focused biological studies. At the same time, such an archive also enables the identification of the least frequently selected genes to understand their contribution to the class discrimination.

Instead of this postinterpretation of list of selected genes, it would be possible and interesting to integrate a priori knowledge of gene networks into a selection and classification method. For instance, the authors of [53] project the expression profiles onto a graph representing biological information between genes (for example, a metabolic gene network). The distance between expression profiles is modified according to the proximity of nodes in the graph. Unsupervised and supervised classification are then applied to these transformed data and the obtained results can easily be interpreted in terms of the network information.

Such an idea of using a priori biological knowledge could be applied in gene selection and classification algorithms based on metaheuristics. For instance, information about interaction network could be used to define specific rules to transform the neighborhood relations used in this article. More generally, such biological information could favorably be explored to refine the search operators and devise more focused strategies of the metaheuristic algorithm. This would help to achieve better solutions, which additionally could be more apt for biological interpretations.

RELATED RESEARCH

Since the first public datasets have appeared, a lot of publications have been devoted to the problem of gene selection for classification of microarray datasets. A great variety of approaches are now available. Even when considering only the metaheuristics, it is difficult to give an exhaustive list. So the goal of this

section is limited to show common properties shared by most heuristic methods.

Several studies have used genetic algorithms to deal with gene selection (see for instance [54–56]). All these methods share a set of common characteristics: they represent the selected gene subsets by a binary vector, employ standard (blind) crossover and mutation operators to evolve a population and use a specific classifier (kNN, SVM for fitness evaluation). Consequently, most of these methods belong to the wrapper family whereas the approach presented in this article is an embedded method with specific genetic operators.

Additionally, a great number of studies follow the same strategy that can be summarized by the following procedure:

(1) Preselect a reduced number of genes (this number may vary from 100 to 1000)
(2) Apply an evolutionary algorithm to explore the search space and record gene subsets of good quality (that achieve a high classification rate)
(3) Compute the frequency of each gene appearing in these gene subsets
(4) Pick up the most frequently chosen genes to form the final gene subset

The work reported in [57] is an illustration of such an approach where a SVM classifier and different kernels are experimented in order to find the best set of parameters for the three datasets considered. A standard genetic algorithm, with random crossover and mutation operators, is used to select different gene subsets from different training sets. The same ideas were used in [58] for a kNN classifier, with an additional feature. At the fourth step of the above procedure, a Z-score analysis of the most frequently selected genes identifies the most interesting genes for discrimination and helps to find the appropriate number of genes to consider.

Besides this class of methods, we find several works that try to incorporate knowledge in the metaheuristic exploration. For instance, [59] presents a MA where a genetic algorithm is hybridized with local search. The GA is based on standard crossover and mutation operators and an SVM classifier for the fitness evaluation. At each generation, the elite individual (the one with the best fitness value) undergoes a local search improvement that relies on two binary operators. The *Add* operator ranks the unselected genes according to a correlation measure and adds the most relevant one to the subset. The correlation measure also identifies the most relevant gene already selected in the elite individual and the *Del* operator removes genes that are redundant with this top ranked gene. Redundancy between genes is evaluated by the notion of Markov blanket. Local search is used, therefore, as a kind of filter and the GA as a wrapper that calls the SVM as a black box.

CONCLUSION

Gene selection is basically a combinatorial search problem within very high dimensions. From this perspective, metaheuristic optimization methods constitute an appropriate solution approach for this difficult task. Three examples are provided showing how to apply a neighborhood metaheuristic (TS) and two evolutionary metaheuristics (Genetic Algorithm and MA) to gene selection. Through these examples, it is argued that metaheuristics should be carefully adapted to the given problem by integrating problem-specific knowledge within their search components such as move operator, fitness function, crossover and mutation. Otherwise, the goal of finding high-quality solutions in a high-dimension search space will be compromised.

Computational assessments showed that the metaheuristic approach is effective to find predictive gene groups of small sizes (often less than 20 genes). Moreover, this approach naturally provides multiple solutions that cannot be achieved by filter methods. These groups of genes can be further analyzed by *a posteriori* biological interpretation in terms of functions and pathways. For this purpose, public tools such as Gene Ontology, Biocarta, GenMAPP and KEGG can be used to cross-validate the gene subsets with respect to known biological functions and genetic networks.

As shown in [53], it is a very promising idea to integrate a priori gene network knowledge or other relevant biological information (to be discovered) into the analysis of gene expression data. Using such information in a metaheuristic algorithm would further improve the search performance and perhaps more importantly would lead to solutions that could be interpreted more easily from a biological perspective.

After a few years that have produced an abundant literature on the subject, Allison *et al.* [19] underlines that some good principles are now recognized for the analysis of microarray datasets, while some questions

remain opened. We can hope that these questions will be solved in the near future, so that microarray analysis can provide reliable biomarkers and valuable diagnostic tools. In this perspective, it is important that the community be aware of all the effective tools available for gene selection and analysis of microarray.

Key Points

- Review of recent advances in metaheuristics-based search methods applied to gene selection and classification of microarray data.
- Emphasis on the pertinence of integrating problem-specific knowledge within the search operators and strategies to ensure the search efficiency.
- Recall some elements about performance of metaheuristics-based search methods and discuss possibilities for further extensions.

Acknowledgements
We are grateful to the two reviewers of the article for their detailed and insightful comments which helped to improve the presentation of the work.

FUNDING

French Biogenouest®; Bioinformatics Program of the Region Pays de La Loire.

References

1. Stoughton RB. Applications of DNA microarrays in biology. *Ann Rev Biochem* 2005;**74**:53–82.
2. Madeira SC, Oliveira AL. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 2004;**1**:24–45.
3. Dimaggio P, Mcallister S, Floudas C, *et al*. Biclustering via optimal re-ordering of data matrices in systems biology: rigorous methods and comparative studies. *BMC Bioinformatics* 2008;**9**:458.
4. Mitchell T. *Machine Learning*. New York: McGraw-Hill, 1997.
5. Duda RO, Hart PE, Stork DG. *Pattern Classification*, 2nd ed John Wiley & Sons, New York, 2001.
6. Kohavi R, John GH. Wrappers for feature subset selection. *Artif Intell* 1997;**97**:273–324.
7. Guyon I, Elisseeff A. An introduction to variable and feature selection. *J Mach Learn Res* 2003;**3**:1157–82.
8. Saeys Y, Inza I, Larraga P. A review of feature selection techniques in bioinformatics. *Bioinformatics* 2007;**23**: 2507–17.
9. Dudoit S, Fridlyand J, Speed T. Comparison of discrimination methods for the classification of tumors using gene expression data. *J Am Stat Assoc* 2002;**97**:77–87.
10. Su Y, Murali TM, Pavlovic V, *et al*. Rankgene: identification of diagnostic genes based on expression data. *Bioinformatics* 2003;**19**:1578–9.
11. Amaldi E, Kann V. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theor. Comput. Sci.* 1998;**209**:237–60.
12. Guyon I, Weston J, Barnhill S, *et al*. Gene selection for cancer classification using support vector machines. *Mach Learn* 2002;**46**:389–422.
13. Ma S, Huang J. Penalized feature selection and classification in bioinformatics. *Brief. Bioinform.* 2008;**9**:392–403.
14. Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. CA: Morgan Kaufmann San Mateo, 1995, 1137–43.
15. Braga-Neto U, Dougherty ER. Is cross-validation valid for small-sample microarray classification? *Bioinformatics* 2004; **20**:374–80.
16. Ambroise C, McLachlan G. Selection bias in gene extraction on the basis of microarray gene-expression data. *Proc Natl Acad Sci USA* 2002;**99**:6562–6.
17. Simon R, Radmacher MD, Dobbin K, *et al*. Pitfalls in the use of dna microarray data for diagnostic and prognostic classification. *J Natl Cancer Inst* 2003;**95**:14–18.
18. Lee S. Mistakes in validating the accuracy of a prediction classifier in high-dimensional but small-sample microarray data. *Stat Methods Med Res* 2008;**17**:635–42.
19. Allison DB, Cui X, Page GP, *et al*. Microarray data analysis: from disarray to consolidation and consensus. *Nat Rev Genet* 2006;**7**:55–65.
20. Boser BE, Guyon I, Vapnik V. A training algorithm for optimal margin classifiers. In: *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*. Pittsburgh, Pennsylvania, United States: ACM Press, 1992, 144-52.
21. Vapnik V. *Statistical Learning Theory*. New York: John Wiley, 1998.
22. Noble WS. Support vector machine applications in computational biology. In: Schölkopf B, Tsuda K, Vert J-P, (eds). *Kernel Methods in Computational Biology*. MIT Press, 2004, 71–92.
23. Rakotomamonjy A. Variable selection using SVM-based criteria. *Mach Learn Res* 2003;**3**:1357–70.
24. Marchiori E, Sebag M. Bayesian learning with local support vector machines for cancer classification with gene expression data. In: *EvoWorkshops-05, Lausanne, Switzerland, Vol. 3449 of Lecture Notes in Computer Science*. Berlin/Heidelberg: Springer, 2005;74–83.
25. Glover F, Kochenberger G. *Handbook of Metaheuristics (International Series in Operations Research & Management Science)*. Berlin/Heidelberg: Springer, 2003.
26. Hoos H, Stutzle T. *Stochastic Local Search: Foundations and Applications*. San Francisco, CA: Morgan Kaufmann Publishers Inc., 2004.
27. Bäck T, Fogel DB, Michalewicz Z (eds). *Handbook of Evolutionary Computation*. Berlin/Heidelberg: Springer, 1997.
28. Liu J, Iba H. Selecting informative genes using a multiobjective evolutionary algorithm. *Congress on Evolutionary Computation*, Hawaii, USA, vol. 1. IEEE Press, 2002, 297–302.

29. Glover F, Laguna M. *Tabu Search*. Boston: Kluwer Academic Publishers, 1997.
30. Hernandez-Hernandez JC, Duval B, Hao J-K. SVM-based local search for gene selection and classification of microarray data. In: Elloumi M, Küng J, Linial M, Murphy RF, Schneider K, Toma C (eds.) *BIRD-08-Communications in Computer and Information Science*, Vol. 13. Berlin/Heidelberg: Springer, 2008, 499–508.
31. Lourenco HR, Martin O, Stützle T. Iterated local search. In: Glover F, Kochenberger G (eds). *Handbook of Metaheuristics*. Berlin/Heidelberg: Springer, 2003.
32. Lü Z, Hao J-K. A critical element-guided perturbation strategy for iterated local search. In: Cotta C, Cowling P, (eds). *EvoCOP-09, Tübingen, Germany, Vol. 5482 of Lecture Notes in Computer Science*. Berlin/Heidelberg: Springer, 2009;1–12.
33. Holland JH. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI: University of Michigan Press, 1975.
34. Goldberg DE. *Genetic Algorithms in Search, Optimization, and Machine Learning*, Vol. 3. Reading, MA: Addison-Wesley, 1989.
35. Mitchell M. *An Introduction to Genetic Algorithms*. USA: MIT press, 1998.
36. Hernandez Hernandez JC, Duval B, Hao J-K. A genetic embedded approach for gene selection and classification of microarray data. In: Marchiori E, Moore JH, Rajapakse JC, (eds). *EvoBIO-07, Vol. 4447 of Lecture Notes in Computer Science*. Berlin/Heidelberg: Springer, 2007, 90–101.
37. Bonilla Huerta E, Duval B, Hao J-K. Gene selection for microarray data by a LDA-based genetic algorithm. In: Chetty M, Ngom A, Ahmad S (eds). *PRIB-08, Melbourne, Australia, Vol. 5265 of Lecture Notes in Bioinformatics*. Berlin/Heidelberg: Springer, 2008, 250–61.
38. Hernandez Hernandez JC, Duval B, Hao J-K. A study of crossover operators for gene selection of microarray data. In: Monmarché N, Talbi E-G, Collet P, Schoenauer M, Lutton E (eds). *EA-07, Vol. 4926 of Lecture Notes in Computer Science*. Berlin/Heidelberg: Springer, 2007, 243–54.
39. Duval B, Hao J-K, Hernandez Hernandez JC. A memetic algorithm for gene selection and molecular classification of cancer. In: *Proceedings of the GECCO-09, Montréal*, Canada. New York: ACM Press, 2009.
40. Moscato P. Memetic algorithms: a short introduction, Ch. 14. In: Corne D, Dorigo M, Glover F (eds). *New Ideas in Optimization.* McGraw-Hill, 1999, 219–34.
41. Golub T, Slonim D, Tamayo P, *et al*. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* 1999;**286**:531–7.
42. Alon U, Barkai N, Notterman D, *et al*. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc Natl Acad Sci USA* 1999;**96**:6745–50.
43. Van't Veer LJ, Dai H, Van de Vijver MJ, *et al*. Gene expression profiling predicts clinical outcome of breast cancer. *Nature* 2002;**415**:530–6.
44. Gordon GJ, Jensen RV, Hsiao LL, *et al*. Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *Cancer Res* 2002;**17**:4963–7.
45. Singh D, Febbo P, Ross K, *et al*. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell* 2002;**1**: 203–9.
46. Petricoin EF, Ardekani AM, Hitt BA, *et al*. Use of proteomic patterns in serum to identify ovarian cancer. *Lancet* 2002;**359**:572–7.
47. Pomeroy SL, Tamayo P, Gaasenbeek M, *et al*. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature* 2002;**415**:436–42.
48. Alizadeh A, Eisen MB, Davis RE, *et al*. Distinct types of diffuse large (B)–cell lymphoma identified by gene expression profiling. *Nature* 2000;**403**:503–11.
49. Furey T, Cristianini N, Duffy N, *et al*. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics* 2000;**16**: 906–14.
50. Liu B, Cui Q, Jiang T, Ma S. A combinational feature selection and ensemble neural network method for classification of gene expression data. *BMC Bioinformatics* 2004;**5**:1–12.
51. Cho S-B, Won H-H. Cancer classification using ensemble of neural networks with multiple significant gene subsets. *Appl Intell* 2007;**26**:243–50.
52. Ein-Dor L, Kela I, Getz G, *et al*. Outcome signature genes in breast cancer: is there a unique set? *Bioinformatics* 2004;**21**:171–8.
53. Rapaport F, Zinovyev A, Dutreix M, *et al*. Classification of microarray data using gene networks. *BMC Bioinformatics* 2007;**8**:35.
54. Li L, Weinberg C, Darden T, *et al*. Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the GA/KNN method. *Bioinformatics* 2001;**17**:1131–42.
55. Ooi CH, Tan P. Genetic algorithms applied to multi-class prediction for the analysis of gene expression data. *Bioinformatics* 2003;**19**:37–44.
56. Peng S, Xu Q, Ling XB, *et al*. Molecular classification of cancer types from microarray data using the combination of genetic algorithms and support vector machines. *FEBS Lett.* 2003;**555**:358–62.
57. Li S, Wu X, Hu X. Gene selection using genetic algorithm and support vectors machines. *Soft Comput* 2008;**12**:693–8.
58. Jirapech-Umpai T, Stuart Aitken J. Feature selection and classification for microarray data analysis: evolutionary methods for identifying predictive genes. *BMC Bioinformatics* 2005;**6**:148.
59. Zhu Z, Ong YS, Dash M. Markov blanket-embedded genetic algorithm for selection. *Pattern Recognition* 2007;**40**: 3236–48.