Progress in Artificial Intelligence (2018) 7:399–410
https://doi.org/10.1007/s13748-018-0161-9

REGULAR PAPER

Improving the genetic bee colony optimization algorithm for efficient gene selection in microarray data

Adrian Pino Angulo[1] · **Kilho Shin**[1] · **Camilo Velázquez-Rodríguez**[2]

Received: 20 June 2018 / Accepted: 14 August 2018 / Published online: 21 August 2018

Abstract
Feature selection is a very critical component in the workflow of biomedical data mining applications. In particular, there is a need for feature selection methods that can find complex relationships among genes, yet computationally efficient. Within the scope of microarray data analysis, the genetic bee colony (GBC) algorithm is one of the best feature selection algorithms, which leverages the combination between genetic and ant colony optimization algorithms to search for the optimal solution. In this paper, we analyse in depth the fundamentals lying behind the GBC and propose some improvements in both efficiency and accuracy, so that researchers can even take more advantage of this excellent method. By (i) replacing the filtering phase of GBC with a more efficient technique, (ii) improving the population generation in the artificial colony algorithm used in GBC, and (iii) improving the exploitation method in GBC, our experiments in microarray data sets reveal that our new method GBC+ is not only significantly more accurate, but also around ten times faster on average than the original

Keywords Feature selection · Gene selection · Microarray data · Machine learning

1 Introduction

A fundamental issue in microarray data analysis is to learn the functional relationship $\ell()$ between different expression levels of m sequence of genes $X = \{x_1, x_2, \dots, x_m\}$ and a discrete class output $C = \{c_1, c_2, \dots, c_t\}$. x_j is a vector of real numbers, where $x_j^i \in g_i$ is the expression level of the i-th gene in the j-th sequence. C represents a finite set of possible results associated with a given sequence. As an instance, x_j might be a vector of values (or sequence) associated with the cells of a tumour biopsy or the cells of the tissue of a healthy patient, whereas C represents whether the patient has cancer or not. In high-dimensional microarray data, the output C is not necessarily determined by the complete set of genes $\{g_1, g_2, g_n\}$; instead, it may be determined by a relatively small number of genes $\{\bar{g}_1, \bar{g}_2, \dots, \bar{g}_{n'}\}$, where $n' \ll n$. In fact, finding a set of genes that approximately determine C brings three key benefits to the knowledge discovery process:

✉ Adrian Pino Angulo
apinoa85@gmail.com

1 Graduate School of Applied Informatics, University of Hyogo, Kobe 651-2197, Japan

2 Grupo de Procesamiento de Datos Biomédicos, Universidad de Holguín, Holguín, Cuba

– Simplification of models, so that researchers can interpret them more easily. This allows a better understanding of the relationship between the input and the output [1].
– Shorter training time, which leads to a faster and more cost-effective microarray data classification.
– Improvement of the prediction performance of classification due to the removal of noisy genes.

Gene selection methods are able to identify and remove unneeded, irrelevant, and redundant genes from data that do not contribute to the improvement of the accuracy of a predictive model [2]. A gene selection algorithm can be seen as the combination of a search method that generates candidate gene subsets, along with an evaluation function, which assigns a score to the candidate gene subset according to its ability to uniquely determine class labels with high likelihood [1].

The simplest algorithm is to test each of the 2^n possible subset of genes finding the one which minimizes the error prediction rate. However, this is an exhaustive search of the space, and its computational cost is prohibitively high. Therefore, alternative search-based techniques have been constantly proposed by the machine learning community. In a broad sense, gene selection algorithms take either the *filter* approach, the *wrapper* approach, or the *hybrid* approach.

The filter approach takes advantage of statistical properties intrinsic to data sets and aims to extract relevant genes that are effective to arbitrary classifier algorithms. By contrast, the wrapper approach aims to select relevant genes so as to improve performance of particular focused classifier algorithms, by involving the classifier algorithms in the selection process. The filters are fast while the wrappers guarantee good results. The *hybrid* approach takes advantage of both the filter and the wrapper: as a general rule, they are faster than wrappers and more accurate than filters. The *genetic bee colony* (GBC) algorithm is a good example within the hybrid approach [3].

GBC is a twofold algorithm, which leverage genetics and ant colony operators to guarantee not to be trapped by local optima. In the first step, GBC uses the filter-based MRMR algorithm to remove irrelevant and redundant genes. Only a few tens or hundreds out of hundred thousands genes are selected in this step. The main goal of feature selection is to identify features (1) that have high correlation with the target class (relevance) but (2) low mutual relevance among them (redundancy). Peng et al. [4] have proposed the algorithm named the *Max-Relevance and Min-Redundancy* algorithm (MRMR), which finds approximate solutions to the aforementioned problem efficiently. MRMR evaluates each subset of genes by the *Mutual Information Difference* measure $\mathrm{MID}_\alpha(\cdot,\cdot)$ defined as shown below:

$$\mathrm{MID}_\alpha(f,\emptyset) = I(f,C); \tag{1}$$

$$\mathrm{MID}_\alpha(f,S) = I(f,C) - \frac{2\alpha}{k}\sum_{f'\in S} I(f,f'), \tag{2}$$

where $I(f,f')$ represents the mutual information between the two genes f and f'. MRMR takes the forward search approach, and hence, the variable S that holds the features selected at each iteration of the for loop (line 2–5) is initialized to the empty set (line 1). Then, for each iteration of the for loop, a single feature f that maximizes $\mathrm{MID}_\alpha(f,S)$ is added to S.

Algorithm 1 MRMR [4]

```
Require: Data set $D$ described by a feature set $\mathbb{F}$ and a number $q$ of
    features to select.
Ensure: A feature subset $\{\bar{f}_1,\dots,\bar{f}_q\} \subset \mathbb{F}$.
1: $S = \emptyset$
2: for $k = 1,\dots,q$ do
3:     $\bar{f}_k \in \mathrm{argmax}\{\mathrm{MID}_\alpha(f,S) \mid f \in \mathbb{F}\setminus S\}$
4:     Add $\bar{f}_k$ to $S$.
5: end for
6: return S
```

In the second step of the GBC algorithm, the genes selected by MRMR are combined to search for the best subset in a narrower search space. The generation of new candidate subsets is accomplished through some bio-inspired operators in the following order: (1) generation of random solutions of a population, (2) update of the solutions with more promising neighbour solutions, (3) crossover between the best solution found so far and the candidate solutions, and (4) mutation. Every candidate subset is assessed by training and testing the *Support Vector Machine* (SVM) in the reduced data set.

Despite GBC is fast, very accurate, and difficult to be trapped by local optima; we found through empirical experiments that GBC has three main gaps in its design that are still subject to improvements:

- In very high-dimensional data sets, MRMR is relatively slow, taking around fifty percentage of the running time of the entire algorithm.
- GBC does not use the information provided by MRMR (i.e. the correlation of each gene with the class variable or the correlation between genes) to create an initial population of solutions closer to the global optima. The initial solutions are created at random, instead.
- During the search, no useful information that can give insight on the quality of the features is stored or analysed. Taking advantage of such information may lead to a significantly faster convergence.

We address these issues as follows.

1. We propose changing the MRMR method by the MRMR+ [6], which finds the same solution of MRMR, but is significantly faster.
2. To generate subsets in the initial population of GBC, we propose a new function that leverages the correlation scores given by MRMR to efficiently create diverse solutions while being closer to the optimal solution.
3. During the entire search, we constantly predict the goodness of a gene, by observing the effect of adding such features to the candidate solutions in the population. Later on, the goodness of a gene is used to discover the level of interaction among genes. Genes with high interaction score are then likely to be grouped in the same subset, while the creation of subsets with genes with low interaction score is avoided. This method tends to make GBC converging faster and escaping from local optima.

Section 2 gives a brief explanation of GBC algorithm. In Sect. 3, we formally present our contribution as an improvement of GBC algorithm and propose a new algorithm called GBC+. In Sect. 4, we make an extensive evaluation by comparing the proposed algorithm GBC+ with other four feature selection benchmark algorithms in 16 microarray data sets.

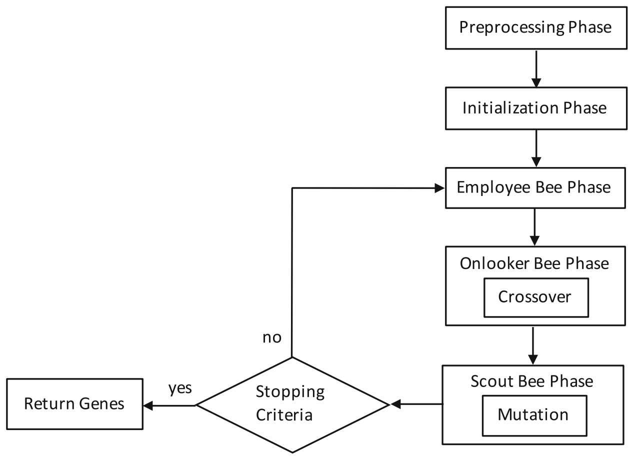

Fig. 1 Main phases of the GBC algorithm

2 Genetic bee colony for gene selection

GBC is a novel hybrid meta-heuristic algorithm that takes advantage of two bio-inspired methods: genetic algorithms (GA) and artificial bee colony (ABC) optimization algorithm. As Fig. 1 shows, GBC is composed of five phases. In the *Preprocessing* phase, the grand majority of features are removed by the filter-based MRMR algorithm. Afterwards, the first SN candidate solutions are randomly generated in the *initialization* phase similarly to the initialization phase in the ABC meta-heuristic algorithms [7].

In the *Employee Bee* phase, the genetic crossover operation is performed between the *Queen Bee*, which is the best solution found so far and solutions randomly chosen from the population, to generate new diverse solutions closer to the optima. Subsequently, the *Scout Bee* phase is accomplished by resetting the solutions trapped by local optima. Also, in this phase, the genetic mutation operation is performed over the *Queen Bee* to intensify the search. The search stops when a number of generations are accomplished. Usually, the number of generations is equal to 100 [3]. The candidate solutions are evaluated by testing the SVM classifier using the leave-one-out cross-validation procedure. In addition to represent a candidate solution, a binary representation is used (0 for a gene not selected and 1 otherwise) as suggested in [3]. In the remainder of this section, we briefly explain the different phases aforementioned.

2.1 Phase 1: the preprocessing phase

In high-dimensional microarray data sets, with hundred of thousands of genes, it is infeasible to apply evolutionary algorithms such as GA and ABC. Therefore, GBC takes advantage of the filter-based algorithm MRMR to remove irrelevant and redundant genes at the very beginning of the search to narrow down the space of solution from 2^n to 2^{qt} subsets, with $qt \ll n$.

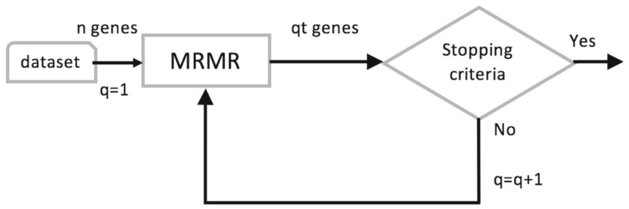

Fig. 2 Preprocessing phase in the GBC algorithm. t is usually fixed to 50 genes

As shown in Fig. 2, MRMR is run several times until the stopping criteria are met. In each run, MRMR returns the subset selected in the previous run plus t additional genes. The preprocessing phase stops when the returned subset can uniquely determine the class variable. That is $\mathrm{SVM}(G) = 1.0$, being $\mathrm{SVM}(G)$ the accuracy reached by the classifier SVM in the reduced data set composed of the current qt genes in G.

2.2 Phase 2: the initialization phase

In the second phase, GBC generates the initial population composed of SN solutions. Each solution is represented as a group of genes indices that are selected from the subset G, returned by the MRMR algorithm. To build a solution, a linear forward selection search is performed. That is, a gene is randomly selected from G and then is tested in the current solution. If the current solution improves by adding the gene, then we continue adding genes while the current solution improves. The solution is built when a gene does not improve it. The i-th gene in a solution is randomly selected according to the following equation:

$$x_j^i = \mathrm{rand}(0, 1) \times qt, \tag{3}$$

where rand(0, 1) represents a random number generator in the range of [0,1) with a normal distribution.

2.3 Phase 3: the employee bee phase

In the *artificial bee colony* (ABC) population-based algorithm, the colony consists of three group of bees: employee bees, onlooker bees, and scout bees [8]. The position of a food source represents a potential solution while the amount of nectar in a food source corresponds to the accuracy of the associated solution. The ABC optimization problem consists in finding the food source with higher amount of nectar through the social cooperation of bees.

GBC algorithm uses this analogy to improve the current population of solutions. GBC sends the employee bees to search in the neighbour of the current SN solutions (food

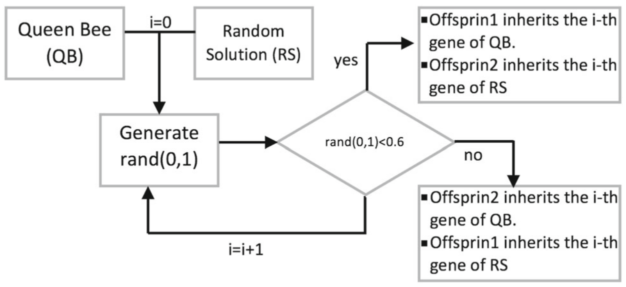
Queen Bee (QB)
i=0
Random Solution (RS)
yes
▪Offspring1 inherits the i-th gene of QB.
▪Offspring2 inherits the i-th gene of RS
Generate rand(0,1)
rand(0,1)<0.6
no
▪Offspring2 inherits the i-th gene of QB.
▪Offspring1 inherits the i-th gene of RS
i=i+1

Fig. 3 Crossover operation between the *Queen Bee* and a solution randomly selected from the population. The crossover parameter was fixed to 0.6 as recommended in [3]

sources) to find solutions that may be closer to the global optima. A neighbour solution is determined by changing the index genes of a current solution by the following equation:

$$v_j^i = \begin{cases} x_j^i + K & \text{if } |S| - K > x_j^i \\ x_j^i - K & \text{otherwise,} \end{cases} \tag{4}$$

with $K = \text{rand}(-1,1) \times (x_j^i - x_j^k)$, where rand$(-1,1)$ denotes a random real number in the range of $[-1,1]$ and k is a random integer number in $[0, SN-1]$.

2.4 Phase 4: the onlooker bee phase

In GBC, the crossover operation is used to share information between employee and onlooker bees in the optimization search space (hive). The employee bees indicate their location of the food sources to the onlooker bees, by a waggling movement.

As Fig. 3 depicts, the crossover operation is accomplished by the *Queen Bee*, which is the best solution found so far, and a solution randomly selected from the current population of bees. The probability a solution has to be selected depends only on its accuracy and can be computed as follows:

$$P(x_j) = \frac{\text{SVM}(x_j)}{\sum_{k=1}^{SN} \text{SVM}(x_k)}. \tag{5}$$

Uniform crossover works by treating each gene independently and making a random choice as to which parent it should inherit, as shown in Fig. 3.

2.5 Phase 5: the scout bee phase

In the GBC algorithm, the scout bee phase is twofold. First, we check all the employee bees in the population and reset all those that have been trapped by local optima. This is achieved by counting the number of times c that we perform an operation in a solution with no improvements. At this point, if $c > \delta$, then we replace the solution with a new subset ran-

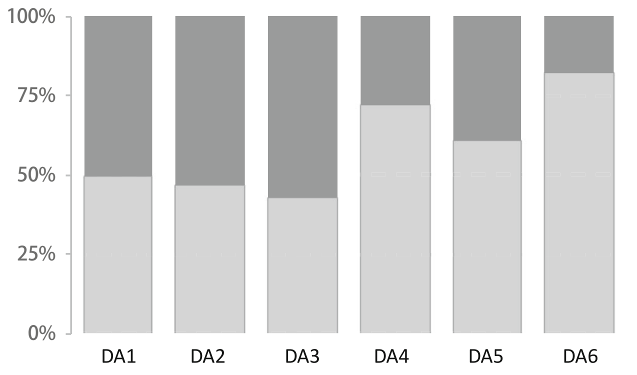

Fig. 4 Percentage of time required by MRMR (the lighter area) in the GBC algorithm

domly generated. δ represents the consecutive number of times we allow a solution to fail on its attempt to be improved. Usually, $\delta = 5$ is used.

Second, a mutation operator is applied to the *Queen Bee* to intensify the search in the neighbourhood of the best solution found so far. Consequently, for each gene in the *Queen Bee*, we generate a random number r in $[0, 1]$ and if $r < \alpha$ then the i-th gene mutates according to the following equation:

$$v_j^i = \begin{cases} \text{Queen}B^i + K & \text{if } |S| - K > \text{Queen}B^i \\ x_j^i - K & \text{otherwise,} \end{cases} \tag{6}$$

with $K = \text{rand}(-1,1) \times (\text{Queen}B^i - x_j^k)$, where rand$(-1,1)$ denotes a random number generator in the range of $[-1,1]$, k is an integer random number in $[0, SN-1]$, and Queen$B$ is the best solution found so far. Usually, $\alpha = 0.01$ is used.

3 Our proposal

Since we are especially interested in high-dimensional data, we made some modifications to the GBC algorithm, so that GBC could be effectively used in high-dimensional domains. In this section, we present some improvements made to the GBC algorithm in terms of efficiency and accuracy.

3.1 First phase of GBC: the MRMR algorithm

MRMR is used in GBC as a filter to remove redundant and irrelevant genes prior to the population-based search. Although MRMR drastically reduces the search space, we have found that the time taken by MRMR is extremely large in relation to the total running time of GBC. Figure 4 depicts the percentage of the running time of MRMR in GBC (lighter area), and the percentage of the running time of the rest of the algorithm (darker area). The data sets used are six microarray data sets from the Rough Sets and Current Trends in Computing conference (RSCTC'2010) discovery challenge [9].

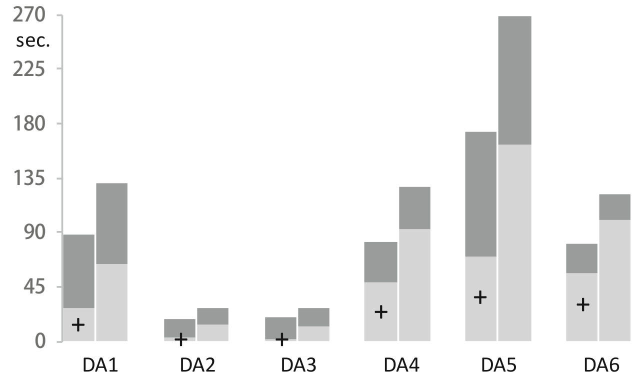

Fig. 5 Comparison of running time (in seconds) in the GBC algorithm when using the original MRMR and the faster MRMR+ in the filter phase. The area with the + symbol represents the MRMR+ while the darker area is the running time of the rest of the GBC algorithm. Results of the original GBC are on the right of the MRMR+

The data sets in Fig. 4 are sorted according to their number of genes. The number of genes and number of instances of these data sets can be found in Table 2. As can be seen, for the first three data sets, which have fewer genes, the running time of MRMR is nearly to the fifty percentage of the entire running time of the GBC algorithm, while for the remaining of the data sets, the running time of MRMR represents more than the sixty percentage of the total running time.

In fact, at each iteration, MRMR computes $\text{MID}_\alpha(f, S)$ for $(n - k + 1)$ features, and $\text{MID}_\alpha(f, S)$ includes k values of mutual information. Hence, the algorithm computes $(n - k + 1)k$ mutual information values at each iteration, and the total number of computing mutual information is

$$\sum_{k=1}^{q}(n-k+1)k = \frac{(3n-2q+2)q(q+1)}{6}, \tag{7}$$

where $n = |\mathbb{F}|$. This number is not small enough to perform gene selection on large microarray data sets [5].

In [6], we present a solution to this problem by proposing the algorithm MRMR+, which finds the same set as MRMR, but in a significantly shorter time. Our experiments in very high-dimensional data reveal that MRMR+ is 13 times faster than MRMR on average [6]. Therefore, we may expect a significant improvement in terms of efficiency when we replace MRMR with MRMR+ in the GBC algorithm. To test the effect of replacing MRMR with MRMR+ over GBC, we run experiments in the RSCTC'2010 challenge data sets and collect the running time of both versions as shown in Fig. 5.

In all cases, MRMR+ is more than two times faster than the original MRMR, which significantly improves the running time of GBC.

3.2 Initialization phase

Most population-based techniques for solving optimization problems in artificial intelligence generate the first solutions of the population randomly. While this is essential to ensure diversity at the early stage of the search, it also may affect the convergence speed of the algorithm. GBC, in the *initialization phase*, does not make use of neither the relevance nor the redundancy scores of each gene, computed by MRMR, to build the SN initial solutions. Instead, GBC selects features at random. Since we work with high-dimensional microarray data, we are very interested in making GBC to converge faster towards the most promising solutions. Therefore, in the *initialization* phase, we propose to make use of the relevance and redundancy score of each gene, which are computed by MRMR in the previous phase, to efficiently create an initial population composed of diverse, but accurate solutions.

We define our proposal as follows. Given a set of all genes sorted according to the order they were selected by MRMR, we randomly select a feature and test it in the current solution (initially the empty solution), if the accuracy of the current solution is increased, then we add the gene; otherwise, we stop the search and start creating a new solution by the same procedure. This is, in fact, the same procedure GBC uses to create the initial population. However, additionally we assign to each gene g_i a probability $P_{\gamma_j}(i)$ to be selected when creating the j-th solution as follows:

$$P_{\gamma_j}(i) = \frac{1-\gamma_j}{1-\gamma_j^n} \times \gamma_j^{i-1}, \tag{8}$$

where n is the number of features in the data and γ_j is a decreasing function in the range of (0, 1) as follows:

$$\gamma_j = 1 - \frac{1}{j+1} \tag{9}$$

Figure 6 represents the shape of the probability function $P_{\gamma_j}(i)$. As can be inferred from Fig. 6, $P_{\gamma_j}(i)$ has several properties:

– First, as a probability function, $P_{\gamma_j}(i)$ is exhaustive, that is: $\sum_{i=1}^{n} P_{\gamma_j}(i) = 1$.
– Second, $P_{\gamma_j}(i)$ is a decreasing function. Therefore,

$$P_{\gamma_j}(1) > P_{\gamma_j}(2) > \cdots > P_{\gamma_j}(n), \tag{10}$$

always holds. This means that first genes in the ranking are likely to be selected. Genes will be ranked in the same order they were selected by the MRMR algorithm. Therefore, the first genes in the ranking are highly correlated with the class variable and are not highly correlated with other features in the ranking.

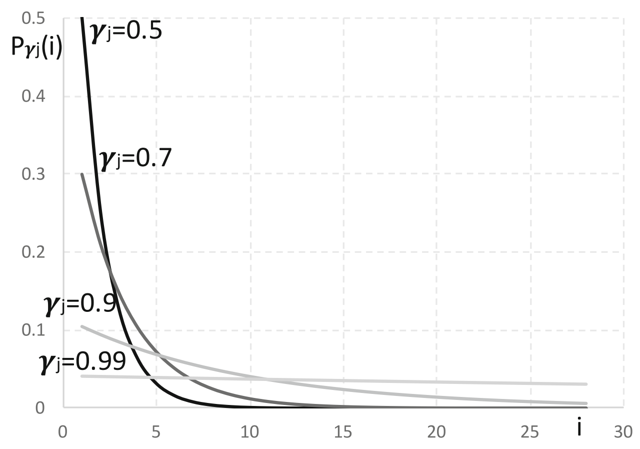

Fig. 6 Chart representing the probability of choosing feature f_i to be tested in the j-th solution

– Third, the larger the number of solutions already built, the more equal are the probabilities to be chosen for all genes. That is,

$$P_{\gamma_t}(i) > P_{\gamma_{t+1}}(i) > \cdots > P_{\gamma_{SN}}(i). \tag{11}$$

However, for a sufficiently large value of j, for example, with $SN/2 \leq k \leq SN$,

$$P_{\gamma_k}(1) \approx P_{\gamma_k}(2) \approx \cdots \approx P_{\gamma_k}(n), \tag{12}$$

holds.

$P_{\gamma_j}(i)$ guarantees that the first solutions are likely to contain genes with high correlation with the class and low correlation with the other genes. Therefore, these solutions may have high accuracy. For the rest of the solutions, the probability of selecting any gene tends to $1/SN$. Consequently, we can expect that the initial population will be composed of two types of solutions: solutions with high accuracy and random solutions. This creates a good synergy in the search since now we have a diverse population with some accurate solutions that may make the algorithm converge faster. To evaluate our proposed method, we run experiments in several data sets and measure the accuracy of the solutions in the population of the original method (grey curve) and the proposed (black curve). Figure 7 depicts the results. To build the chart, we sorted the solutions according to their accuracy for both methods.

3.3 Intensification

Meta-heuristic optimization algorithms often perform well approximating solutions because they first diversify the search looking for candidate solutions without making any assumption about the underlying fitness landscape. Second, they intensify the search looking for more promising solutions once they explore diverse regions in the solution space. In GBC, the intensification process is accomplished by means of the genetic crossover and mutation operations in the *onlooker bee* and *scout bee* phases, respectively. However, through experiments, we realized that in the *scout bee* phase the mutation operation, described in Sect. 2.5, does not make any effect in the intensification process due to the extremely low mutation probability of genes. To carry out a richer intensification process, we adopt the following method.

– First, we determine the goodness of a gene according to its occurrence in the solutions of the population. A gene $f_i \in S$ that is in solutions S, with $\text{SVM}(S) > \mu$, and is not in solutions R, with $\text{SVM}(R) \leq \mu$, must have high goodness. We define the goodness $\ell(f_i)$ of feature f_i as follows.

$$\ell(f_i) = \frac{\sum_{S \in P} \delta^{+}_{f_i,\mu} \times \text{SVM}(S)}{\theta^{+}} - \frac{\sum_{S \in P} \delta^{-}_{f_i,\mu} \times \text{SVM}(S)}{\theta^{-}}, \tag{13}$$

where $\delta^{+} = 1$ if $\delta^{+} > \mu$ and $\delta^{+} = 0$ if $\delta^{+} \leq \mu$, being μ the average of the fitness of all solutions in the population, δ^{-} has opposite value to δ^{+}, and θ^{+} and θ^{-} are the sum of the fitness of all solutions S in the population such that $\text{SVM}(S) > \mu$ and $\text{SVM}(S) \leq \mu$,respectively. $\ell(f_i)$ is a normalized coefficient in the range of $[-1, 1]$. A value of 1 means gene f_i is in all solutions where $\delta^{+}_{f_i,\mu} = 1$ and is not in any solution where $\delta^{-}_{f_i,\mu} = 1$. A value of -1 means f_i is present in all solutions where $\delta^{-}_{f_i,\mu} = 1$ and not in any solutions with $\delta^{+}_{f_i,\mu} = 1$.

– Second, we sort the genes according to their goodness ℓ and store them in two different sets. Genes with $\ell(f_i) > 0$ are sorted in increasing order and stored in S^{+} while genes with $\ell(f_i) \leq 0$ are sorted in decreasing order and stored in S^{-}. Afterwards, we run a greedy forward selection search starting with the *Queen Bee* solution and using genes in S^{+}. That is, we test adding to *Queen Bee*, all genes in $S+$, one by one, and the gene that maximize $\text{SVM}(\text{Queen}B \cup \{f_i\})$ is added to $QueenB$. We stop searching when neither of the genes improves the current $QueenB$. Subsequently, a greedy backward search is performed using the genes in S^{-}. That is, we start with the current *Queen bee* and test eliminating genes in S^{-} from $QueenB$, if present. The gene that maximizes $\text{SVM}(\text{Queen}B \setminus \{f_i\})$ is removed from $QueenB$. The search stops when no feature improves $QueenB$.

To test our method, we run GBC twice for each data set: first, we run the original GBC and second we replace the mutation operation with our proposed intensification method.

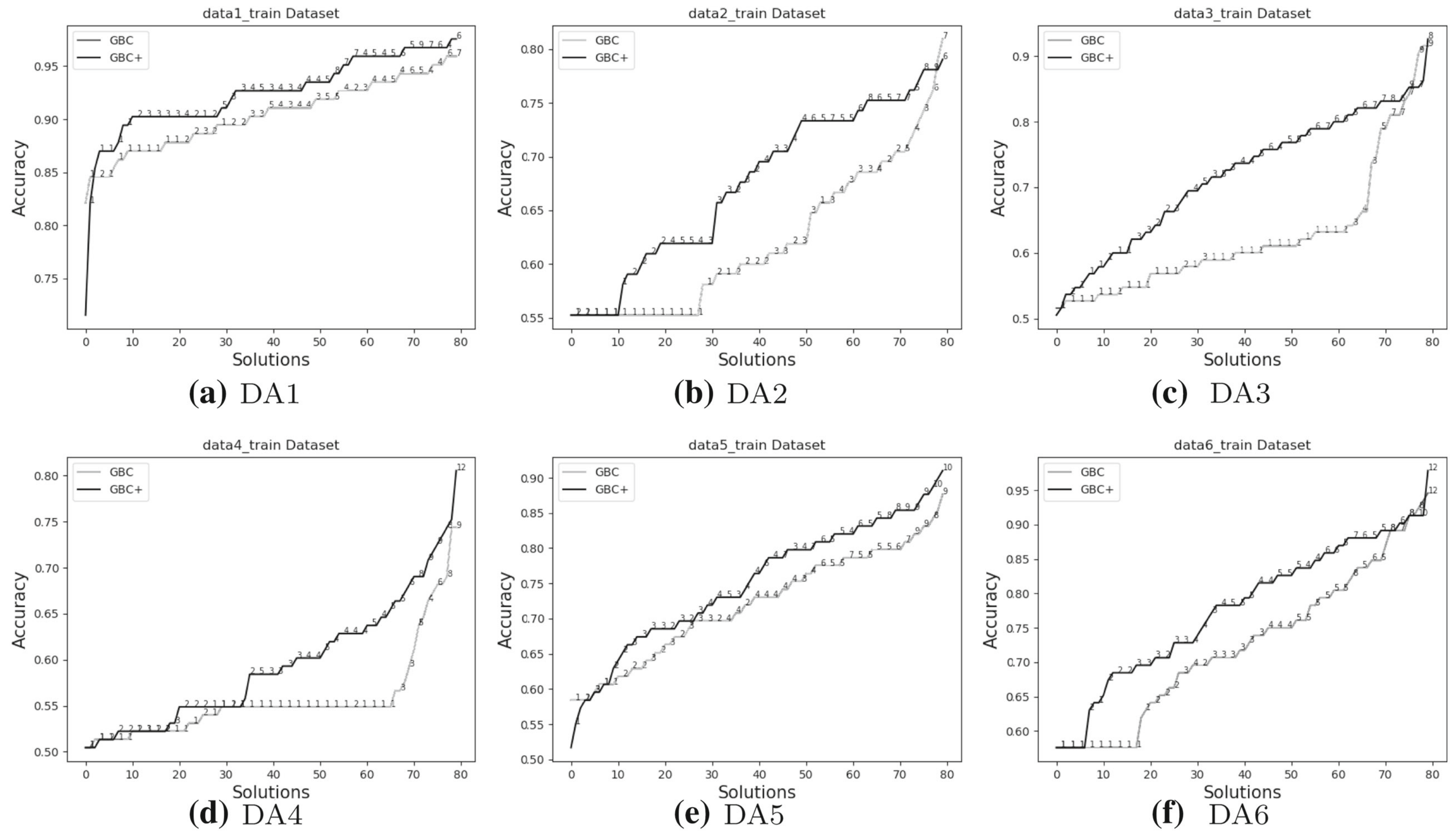

Fig. 7 Accuracy of the solutions in the population of the original method (grey) and the proposed method (black curve). A number of selected genes are located over each solution. **a** DA1, **b** DA2, **c** DA3, **d** DA4, **e** DA5, **f** DA6

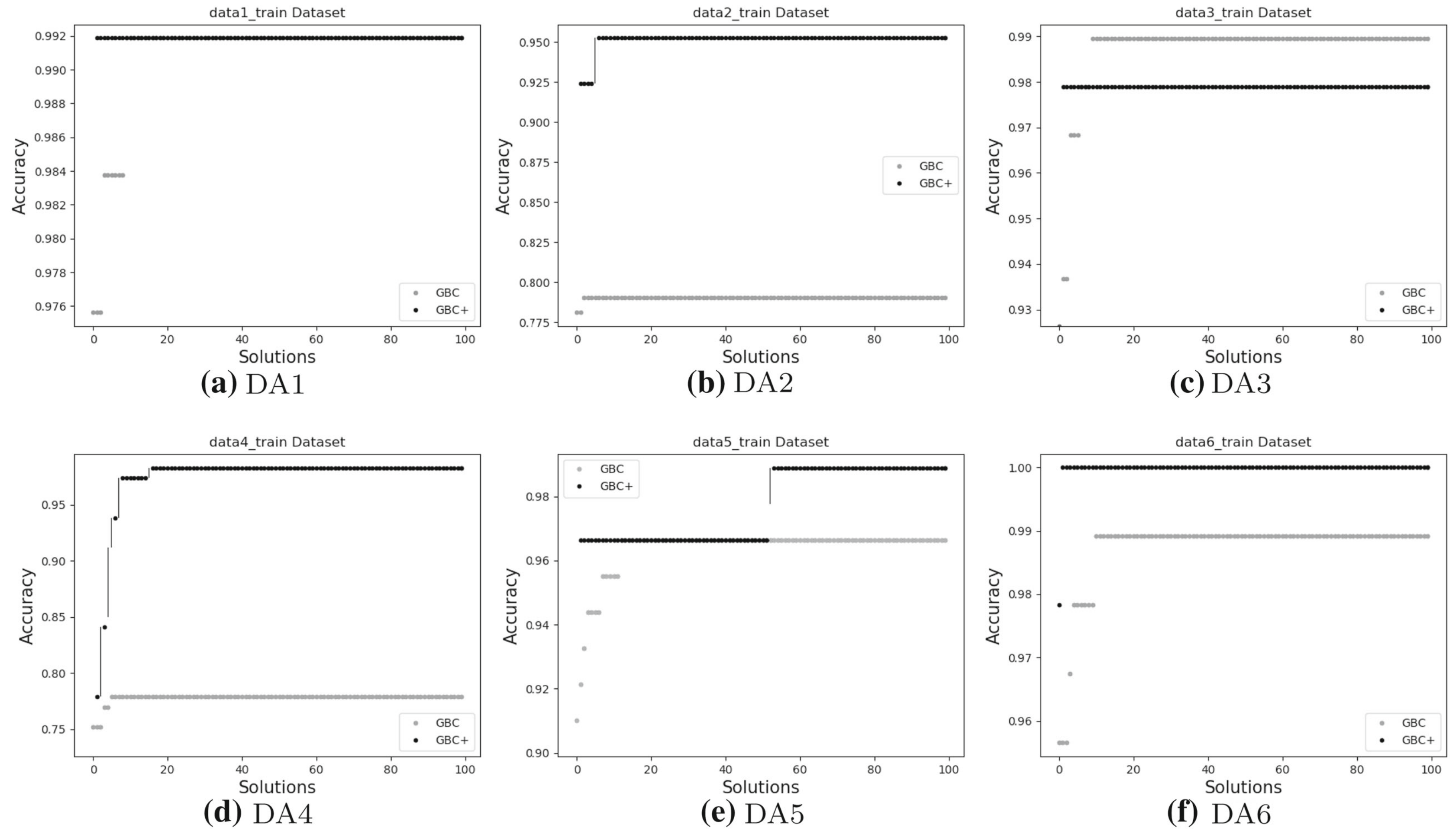

Fig. 8 Comparison between the original GBC (grey points) and GBC with the proposed method of intensification (black points). The line between two black points quantifies an improvement in the *Queen Bee* by the proposed method. **a** DA1, **b** DA2, **c** DA3, **d** DA4, **e** DA5, **f** DA6

Table 1 Percentage of time saved by the minor improvements respect to the original GBC algorithm

	Stopping criteria		Lookup table	All
	(i)	(ii)		
DA1	79/41	–	94/55	81/44
DA2	75/73	–	41/40	85/85
DA3	64/62	–	51 /50	84/84
DA4	64/52	–	72/63	88/73
DA5	60/27	–	76/52	73/52
DA6	63/26	92/68	93/61	86/53
AVE.	67/47	92/68	71/53	83/65

Values are expressed as % of number of evaluations saved/percentage of running time saved
AVE stands for average

Figure 8 depicts the results. The line between two black points quantifies an improvement in the *Queen Bee* by the proposed method.

3.4 Minor improvements

In addition to the improvements we have proposed in the previous sections, we have detected some small gaps on the design of the original GBC that could lead to undesirable results. Our final proposals are as follows.

- First, we consider two more stopping criteria in the GBC algorithm: (i) stopping the search when SVM(QueenB) $\geq \lambda$ and (ii) stopping the search when the current *QueenB* remains the same after t consecutive cycles.
- Second, we implement a *lookup table* that stores the solutions already evaluated and their respective accuracy. Every time a candidate solution is going to be evaluated, we inspect the lookup table to avoid duplicated evaluations.

These modifications may look naive. However, we run experiments to determine the evaluations saved by implementing these modifications in the original GBC algorithm and results were very impressive. Table 1 shows the results. We report that in neither of the data sets the accuracy varies with respect to the original GBC.

Table 1 represents how efficient the minor improvements are. The results are expressed as % of number of evaluation saved/percentage of running time saved. For example, for the DA1 data set, the 79 and 41% were saved with respect to the number of evaluations and running time of the original GBC algorithm, respectively.

4 Experimental evaluation

4.1 Experimental set-up

To validate our proposed algorithm GBC+, we run experiments and collect the running time, the accuracy of each algorithm, and the number of genes selected. We compare our proposed algorithm with four feature selection algorithms, including GBC.

For each data set, we generate m pairs of training and test data subset, where m is the number of instances in the data. Each test data subset represents an instance in the data set, and the training data subset is composed of the rest of the instances. First, we run the feature selection algorithms in the training data and then we reduce the test data according to the features eliminated by the algorithms. Second, we train the C-SVM-with-RBF-Kernel classifier with the reduced training data and then test them on the reduced test data to determine the Area Under the Receiver Operating Characteristic curve (AUC-ROC) score. In addition, the average of the running time taken by each algorithms and the number of genes selected, when is applied to the training data, is collected. In the comparison, we use four feature selection algorithms: GBC, MRMR- GA [10], MRMR- BA [11], and PSO [12], which are population-based algorithms. MRMR- GA is very similar to GBC, but after running MRMR, uses a genetic algorithm to search for the optimal solution. MRMR- BA is also a meta-heuristic algorithm that also uses MRMR as a filter in the first phase to narrow down the search space. In the second phase, MRMR- BA uses the BAT algorithm, which is a bio-inspired technique that emulates the echolocation behaviour of bats. The PSO algorithm is the *Particle Swarm Optimization* algorithm implemented in weka [13]. However, we have added a first phase by running MRMR to reduce the search space as in the GBC algorithm. All the algorithms use as evaluation function the accuracy of the SVM when tested in the given subset to evaluate.

Table 2 shows the characteristics of the data sets used in the experiments, which were collected from two machine learning data repositories: Open Machine Learning [14] and Machine Learning Data; and the feature selection challenge RSCTC'2010 [9].

4.2 Accuracy

Table 3 shows the AUC values of the support vector machine classifier over the reduced sets output by every algorithm. Surprisingly, GBC+ wins in 11 data sets while in the other data sets gets the best results with ties. The other algorithms got good results in a general sense. However, the results of GBC+ are outstanding .

To detect significant differences among algorithms, we compute averaged ranking and the critical distance as stated

Table 2 Characteristics of the data sets used in the experiments

Data	Acronym	#Features	#Instances	#Classes	Source
St. Jude leukaemia	STJ	12,559	327	7	[14]
Breast cancer	BRE	24,482	97	2	[14]
ECML	ECM	27,680	90	43	[15]
Hepatitis C	HEP	22,278	123	4	[15]
Burkitt lymphoma	BUR	22,284	220	3	[14]
Data1	DA1	54,676	123	2	[9]
Data2	DA2	22,284	105	3	[9]
Data3	DA3	22,278	95	5	[9]
Data4	DA4	54,676	113	5	[9]
Data5	DA5	54,614	89	4	[9]
Data6	DA6	59,005	92	5	[9]
Anthracycline	ANT	61,360	159	2	[14]
Mouse type	MOU	45,102	214	7	[14]
Ovarian tumour	OVA	54,622	283	3	[14]
Various cancer	VAR	54,676	383	10	[14]
Pems	PEM	138,673	267	7	[15]

Table 3 Accuracy of SVM in the reduced sets outputted by different feature selection algorithms in several data sets

Alg\data	STJ	BRE	ECM	HEP	BUR	DA1	DA2	DA3
Gbc	0.76	0.94	*1*	*1*	0.96	*0.99*	0.93	0.97
Gbc+	*0.94*	*0.96*	*1*	*1*	*0.98*	*0.99*	*0.97*	*1*
PSO	0.91	0.93	0.97	0.94	0.97	*0.99*	0.94	0.89
MRMR-GA	0.83	0.92	0.99	0.99	0.86	0.94	0.94	0.93
MRMR-BA	0.78	0.93	*1*	0.99	0.96	*0.99*	0.94	0.95
	DA4	DA5	DA6	ANT	MOU	OVA	VAR	PEM
Gbc	0.85	*0.98*	*1*	0.87	0.68	0.96	0.89	0.85
Gbc+	*0.98*	*0.98*	*1*	*0.99*	*0.99*	*0.98*	*0.99*	*0.87*
PSO	0.89	0.96	*1*	0.87	0.83	0.94	*0.99*	0.83
MRMR-GA	0.96	0.93	0.97	*0.99*	0.78	*0.98*	0.87	0.85
MRMR-BA	*0.98*	0.96	*1*	0.93	0.68	0.94	0.99	0.86

in the Nemenyi test. Figure 9 shows the critical distance chart, where the group of algorithms that has no significant differences are connected by a thick line. From Fig. 9, we state that GBC+ is significantly more accurate than the rest of the algorithms, in these data sets with a confidence of 90%. However, results of the rest of the algorithms are not significantly different.

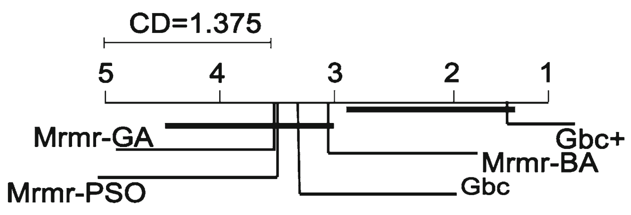

Fig. 9 Nemenyi post hoc chart. Group of algorithms with no significance difference are connected by the thick lines with a confidence of 90%

4.3 Running time and number of features selected

Speaking about the running time, GBC+ is the fastest in all data sets. In fact, GBC+ is ten times faster than GBC on average. Moreover, through a deep analysis of GBC+ behaviour, we discover that although the intensification procedure proposed in Sect. 3.3 may increase the accuracy of GBC+, it also may have a negative impact on its efficiency. We discovered that the intensification process tends to increase the number of features in the *QueenBee*. Therefore, in the crossover operation, all the solutions in the population grows in number of features, which makes the rest of the procedures in the algorithm very heavy and high costly. Although we do not have a solution to this problem at the moment, GBC+ is still fast.

Table 4 Running time of the feature selection algorithms in several data sets

Alg\data	STJ	BRE	ECM	HEP	BUR	DA1	DA2	DA3
Gbc	548	50.3	907	87.6	187	196	83.6	88.3
Gbc+	*8.52*	*16.7*	*31.1*	*7.06*	*51.8*	*120*	*35.5*	*15.7*
PSO	721	78.5	403	102	122	216	101	49
MRMR-GA	378	81.9	524	141	266	278	92.1	76.4
MRMR-BA	182	29.3	108	23.1	103	178	72.7	26.7

	DA4	DA5	DA6	ANT	MOU	OVA	VAR	PEM
Gbc	337	155	157	244	346	331	1704	6397
Gbc+	*101*	*67.9*	*106*	*187*	*42.6*	*123*	*387*	*421*
PSO	324	194	109	434	281	229	2005	5184
MRMR-GA	219	84.8	143	313	329	486	1647	8562
MRMR-BA	328	140	127	207	47.9	166	842	1902

Table 5 Number of genes selected by the feature selection algorithms in several data sets

Alg\data	STJ	BRE	ECM	HEP	BUR	DA1	DA2	DA3
Gbc	*1*	7	*1*	9	10	*6*	*9*	*7*
Gbc+	22	12	2	11	14	7	13	10
PSO	31	14	6	12	14	8	12	27
MRMR-GA	18	14	5	*7*	12	9	21	19
MRMR-BA	5	*6*	4	12	*9*	14	21	14

	DA4	DA5	DA6	ANT	MOU	OVA	VAR	PEM
Gbc	*9*	*10*	11	7	*1*	*6*	13	14
Gbc+	20	11	12	19	27	10	39	27
PSO	12	11	17	11	32	12	*9*	29
MRMR-GA	13	12	*7*	21	38	9	16	*10*
MRMR-BA	*9*	11	21	*6*	29	12	13	18

Speaking about the number of features selected, we realize that all the algorithms get good results. However, algorithm GBC gets the best results in general (Tables 4, 5).

5 Conclusions

In this paper, we deeply analyse the GBC algorithm and point out some drawbacks on its design. We improve the GBC algorithm by improving the filter phase, where we speed up the MRMR algorithm. In addition, we create a new mechanism to generate initial solutions that are closer to the local and global optima. In this sense, we expect the new algorithm to converge faster. Finally, we propose the implementation of a new intensification procedure that leads to a substantial improvement in the accuracy of GBC. Experiments reveal that the new version of GBC is significantly faster and more accurate than its original version and other three state-of-the-art algorithms.

Acknowledgements This work was partially supported by the Grant-in-Aid for Scientific Research (JSPS KAKENHI Grant Number 17H00762) from the Japan Society for the Promotion of Science.

References

1. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3**, 1157 (2003)
2. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artif. Intell. **97**(1), 273 (1997)
3. Alshamlan, H.M., Badr, G.H., Alohali, Y.A.: Genetic bee colony (GBC) algorithm: A new gene selection method for microarray cancer classification. Comput. Biol. Chem. **56**, 49 (2015)
4. Peng, H., Long, F., Ding, C.: Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans. Pattern Anal. Mach. Intell. **27**(8), 1226 (2005)
5. Ramrez-Gallego, S., Lastra, I., Martnez-Rego, D., Boln-Canedo, V., Bentez, J.M., Herrera, F., AlonsoBetanzos, A.: FastmRMR: fast minimum redundancy maximum relevance algorithm for high-dimensional big data. Int. J. Intell. Syst. **32**, 134–152 (2017). https://doi.org/10.1002/int.21833

6. Pino, A., Shin, K.: Mrmr+ and Cfs+ feature selection algorithms for high-dimensional data. submitted to Applied Intelligence (2018) (**under review**)
7. Xiang, W.L., An, M.Q.: An efficient and robust artificial bee colony algorithm for numerical optimization. Comput. Oper. Res. **40**(5), 1256 (2013)
8. Karaboga, D., Akay, B.: A comparative study of artificial bee colony algorithm. Appl. Math. Comput. **214**(1), 108 (2009)
9. Wojnarski, M., Janusz, A., Nguyen, H.S., Bazan, J., Luo, C., Chen, Z., Hu, F., Wang, G., Guan, L., Luo, H., Gao, J., Shen, Y., Nikulin, V., Huang, T.H., McLachlan, G.J., Bošnjak, M., Gamberger, D.: RSCTC'2010 discovery challenge: mining DNA microarray data for medical diagnosis and treatment. In: Szczuka, M., Kryszkiewicz, M., Ramanna, S., Jensen, R., Hu, Q. (eds.) Rough Sets and Current Trends in Computing. 7th International Conference, RSCTC 2010, Warsaw, Poland, June 28–30, 2010 Proceedings, pp. 4–19. Springer, Berlin (2010)
10. El Akadi, A., Amine, A., El Ouardighi, A., Aboutajdine, D.: A two-stage gene selection scheme utilizing MRMR filter and GA wrapper. Knowl. Inf. Syst. **26**(3), 487 (2011)
11. Osama, A., Tajudin, A., Azmi, M., Mohammad, L.: Gene selection for cancer classification by combining minimum redundancy maximum relevancy and bat-inspired algorithm. Int. J. Data Min. Bioinf. **19**(1), 32 (2017)
12. Moraglio, A., Di Chio, C., Poli, R.: Geometric particle swarm optimization. In: Ebner, M., O'Neill, M., Ekárt, A., Vanneschi, L., Esparcia-Alcázar, A.I. (eds.) Genetic Programming, 10th European Conference, EuroGP 2007, Valencia, Spain, April 11–13, 2007, Proceedings. Lecture Notes in Computer Science, vol. 5481, pp. 125–136. Springer, Berlin (2007)
13. Witten, I., Frank, E., Hall, M., Pal, C.: Data Mining: Practical Machine Learning Tools and Techniques. The Morgan Kaufmann Series in Data Management Systems. Elsevier, New York (2016)
14. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: OpenML: networked science in machine learning. SIGKDD Explor. **15**(2), 49 (2013)
15. Wojnarski, M., Stawicki, S., Wojnarowski, P.: TunedIT.org: system for automated evaluation of algorithms in repeatable experiments. In: Szczuka, M., Kryszkiewicz, M., Jensen, R., Hu, Q. (eds.) Rough Sets and Current Trends in Computing, 7th International Conference, RSCTC 2010, Warsaw, Poland, June 28-30, 2010 Proceedings, pp. 12 (2011)