Bee Colony Optimization for Feature Selection

Jana Vuckovic[1] and Tatjana Davidovic[2]

[1] Faculty of Mathematics, University of Belgrade jana.vuck@gmail.com
[2] Mathematical Institute of Serbian Academy of Sciences and Arts
tanjad@mi.sanu.ac.rs

Abstract. Many optimization problems are known to be NP hard problems. One of these problems is a Feature Selection (FS). It can be formulated as follows. Suppose we are given a data set which consists of many instances, each described by many attributes and a particular way of learning how to classify those instances. Our goal is to decrease the number of attributes in order to reduce the dimension of classification problem with minimum degradation (if any) in performances of learning. In this paper we apply metaheuristic named Bee Colony Optimization (BCO) to this problem. We consider four different variations of BCO, test and discuss their performances on 5 databases from UCI Machine Learning Repository.

Keywords: NP-hard Optimization Problems · Clustering · Classification · Metaheuristic.

1 Introduction

Classification is an optimization problem with a goal is to categorize elements into different classes based on their properties. There are many Machine Learning (ML) algorithms for classification. Classification usually refers to supervised learning, while clustering refers to unsupervised learning. In this paper we use clustering algorithm K-Means as a way of learning. More about the algorithm can be found in [6].

Complexity of classification and clustering algorithms usually depend on number of attributes. Therefore, we want to decrease that number as much as possible, but without affecting the performance of the underlying learning procedure. This problem is known as a Feature Selection (FS). The trivial solution to this problem is checking all the subsets of attributes. However, this approach has the exponential complexity. Hence, it can be used only for very small number of attributes, which is rarely the case. In this paper we address this problem by using metaheuristics.

Metaheuristics are high level heuristics applied to find a good solution to the considered optimization problem. There is no guarantee how close the found solution is to the optimal solution. Metaheuristics are usually inspired by natural events and model behaviour of some population. In this paper we observe metaheuristic called Bee Colony Optimization (BCO), which is an example of Swarm Intelligence algorithm. We present four different implementations of BCO. Overview of BCO is given in [2], [7]. Authors of those papers differentiated two versions of BCO called constructive (BCOc) and improvement (BCOi). We applied BCOi version of BCO on FS while in [4], [5] BCOc is applied.

Paper is organized as follows. In the second section we give an overview of the most popular FS methods. In the third section we briefly describe BCO and provide the details about the implementation of our algorithms. Fourth section contains information about databases that we used and results we obtained. Conclusion is given in fifth section.

2 Feature Selection

We consider FS problem, that can be formulated as follows. Let us assume that we are given a set O of m objects, each described with n attributes. In this paper we consider only numerical values of attributes. Let $x \subset R^{n \times m}$ be matrix, such that $x_{i,j}$ represents a value of the i-th attribute of the j-th object. Each object belongs to exactly one of K categories, and that information is stored in vector y. We can apply many machine learning algorithms in order to teach our program how to classify objects into correct categories and we can evaluate each solution we get. So, given the data (x, y), ML algorithm and concrete way to evaluate solution, the goal is to find the most significant subset of attributes.

Feature selection methods can be divided into three main categories [1]:

- **Filter methods** are based on statistical analysis. Their job is to calculate a relevance of an attribute, usually using some statistical test or using some correlation criteria. So this methods do not call ML algorithm and because of that they are very efficient.
- **Wrapper methods** consider FS as an optimization problem, which search space consists of all subsets of $\{1, 2...n\}$ (except the empty subset) and objective function of a subset is evaluation of classification using that attributes. Since there are $2^n - 1$ elements in a search space, solution that consists of checking every combination of attributes, has the exponential complexity. Wrapper methods use some search algorithms with polynomial complexity, but it does not guarantee that the found subset maximizes the objective function. Although wrapper algorithms have polynomial complexity, they call ML algorithm many times. Because of this, wrapper methods are time expensive, and they are usually applied only to smaller data sets. Also, they depend on ML algorithm we use. Some of Wrapper Methods are Forward Selection, Backward Selection etc.
- **Embedded methods** select features during the model training. This implies that embedded methods can be used only with supervised learning algorithms. However, they can be both as fast as filter methods and considering the interaction of features like wrapper methods do.

More about FS methods can be found in [1]. According to previous explanation of different FS methods, our approach is mostly similar to wrapper methods.

It considers FS as an optimization problem, uses metaheuristic as a search algorithm, and K-means as an unsupervised learning procedure.

3 Bee Colony Optimization Method

3.1 Overview

Bee Colony Optimization (BCO) is a metaheuristic inspired by foraging habits of Honeybees [2]. It is an iterative method. At the beginning, initialization is done, meaning each bee is given a solution. One iteration consists of NC repetitions of forward pass and backward pass, and selecting solution for the next iteration for each bee. During forward pass each bee transforms its solution, the best global solution is updated, if needed, and the stopping condition is checked. During backward pass each bee evaluates its solution, decides whether it stays loyal or becomes uncommitted follower, and at the end recruiting is done. Bees which stay loyal to its current solution will later continue on transforming that particular solution. Bees which are not loyal to their solutions are called uncommitted followers and they are assigned a loyal bee during recruiting. We call that loyal bee a recruiter of the uncommitted follower. After being recruited, uncommitted follower changes its solution into recruiter's solution. We are constantly checking whether the global best solution has been improved. The very end of iteration is selecting a solution for the next iteration for each bee. All the above described steps are done until a stopping condition is satisfied. Stopping condition can be time limit, maximal number of iterations, number of function evaluations, etc. Whether the stopping condition is satisfied is checked during every forward pass, because they require most of the time.

According to the categorization in [2], [7] BCO implementations for FS can be divided into two categories. If transformation in forward pass consists only of adding the attributes, that is known as BCOc, whereas BCOi enables both adding and removing the attributes. We consider BCOi.

In this paper we will discuss two different ways of transformations, which we call *tournament* and *roulette*, and two different ways of making a loyalty decision, which we call *roulette* and *mean*. That gives us four different variations of algorithm in total.

Pseudo-code is provided in Algorithm 1, while the remainder of this section contains the concrete details for initialization, transformation, evaluation, loyalty decision, recruiting and solution selection we used.

3.2 Implementation

First of all, we introduce the notation. Let n be a number of attributes, m a number of objects, K a number of categories, $x \in R^{n \times m}$ data which describes objects to be categorized and $y \in \{1, 2...K\}^m$ vector which stores correct categories of each object. A solution is represented as a vector $v \in \{0, 1\}^n$, such that if $v[i] = 1$ then attribute i is included, otherwise it is not. Basic parameters

Algorithm 1 Pseudo-code of BCO algorithm

```
1: procedure BCO($B, NC, STOP, unsuccess\_max$)
2:     Initialization()
3:     $unsuccess = 0$
4:     while True do:
5:         $Improvement$ = False
6:         for $step$ in $1 : NC$ do:
7:             for $b$ in $1 : B$ do:
8:                 Transform($sol[b]$)
9:                 Evaluate($sol[b]$)
10:                imp = Update(($sol_{best}, f_{best}$))
11:                $Improvement = Improvement$ or $imp$
12:                if $STOP$ then:
13:                    return $sol_{best}$
14:            DecideLoyalty()
15:            Recruiting()
16:        if $Improvement$ then:
17:            $unsuccess = 0$
18:        else:
19:            $unsuccess + = 1$
20:        if $unsuccess == unsuccess\_max$ then:
21:            $unsuccess = 0$
22:            Initialization()
23:        else:
24:            for $b$ in $1 : B$ do :
25:                $sol[b] = sol_{best}$
```

of BCO are B which represents a number of bees, NC a number of alternations between forward and backward passes per iterations, a stopping condition $STOP$ and $unsuccess\_max$ which will be described later in the paper. Vectors sol and f are such that $sol[j]$ and $f[j]$ represent current solution of j-th bee and its fitness, for $j = 1..B$. sol_{best} and f_{best} represent respectively the best global solution and its fitness.

Initialization provides each bee with a same solution s randomly generated in the following way. Each attribute is evaluated by calculating the accuracy of clustering depending only on that attribute. Assume that we store that information in vector h. After that, we define vector h_{norm} such that:

$$h_{norm}[i] = \frac{h[i] - h_{min}}{h_{max} - h_{min}}.$$

$h_{norm}[i]$ takes value from the interval [0,1] and it represents probability that i-th attribute is chosen. Now we use roulette selection, to decide for each attribute whether it will be included. So s is constructed as follows: for each attribute $attr = 1..n$, we generate a random number r from uniform $U(0, 1)$ distribution. If $r \leq h_{norm}[attr]$, then $s[attr] = 1$, otherwise $s[attr] = 0$.

Transformation is implemented in two different ways, named *tournament* and *roulette*, inspired by tournament and roulette selection. Transforming a solution means adding or removing some attributes. In our case, half of bees are always removing attributes and half of bees are always adding attributes. This implies that we always set parameter B to be even.

Tournament transformation is implemented using tournament selection. In case of adding attributes each bee is randomly given a set of five not selected attributes and the bee is supposed to select one which best suits with its current solution. So for each of those attributes bee evaluates solution obtained by adding that attribute, and picks the one with the highest evaluation. This is repeated r times where r is randomly chosen from $\{1, 2, 3\}$. If in some point five not selected attributes do not exist, we randomly chose five attributes that are already included in the solution, remove them and continue with the procedure. Similar is in the case of removing attributes. Tournament approach implies many solution evaluations, which could be time expensive.

Roulette transformation involves less solution evaluations. In case of adding attributes, bee considers all the attributes that are not contained in its solution. Let $Unsel$ be the set of all attributes with that property. Assume that $h[i]$ represent accuracy of clustering depending only on attribute i (same as before). Let us define two values h_{min}^{Unsel} and h_{max}^{Unsel} that represent minimum and maximum value of $\{h[i] | i \in Unsel\}$. We also define $h_{norm}^{Unsel}[i]$, for $i \in Unsel$ such that:

$$h_{norm}^{Unsel}[i] = \frac{h[i] - h_{min}^{Unsel}}{h_{max}^{Unsel} - h_{min}^{Unsel}}$$

For each attribute $i \in Unsel$ we generate random number r from uniform $U(0, 1)$ distribution. If $r \leq h_{norm}^{Unsel}[i]$ we add attribute i. Removing attributes is similar.

Evaluation, or as we also say, calculating fitness of a solution, is defined as accuracy of clustering. We always know number K of categories we have, so we can use K-Means algorithm for clustering. We do clustering only using attributes contained in a particular solution. Let us define vector y_labels, such that if $y\_labels[i] = c$, for some $1 \leq i \leq m$ and $1 \leq c \leq K$, then instance i belongs to cluster c. The idea is that the objects from same cluster should be in same category. So, we have to decide for each cluster a category which represents it. The easiest way is to calculate how many instances from each category is in a particular cluster, and to pick category which has the most representatives in that cluster. After we find mapping from clusters to categories, we define accuracy as a quotient of correct guesses and number of objects m.

Notice that evaluation of a solution demands lots of time, so we do not want to evaluate same solutions twice. In order to avoid it, we use hash table to store already evaluated solutions and their fitness.

Making a loyalty decision is done in two different ways - mean and roulette. Both approaches guarantee existence of at least one loyal bee. One approach is deterministic, while other is stochastic. Before we introduce those approaches, let us recall that $f[b]$ represent fitness of solution corresponding to bee b.

Mean principle suggests that bee b is loyal if and only if $f[b] \geq f_{mean}$, where

$$f_{mean} = \frac{\sum_{b=1}^{B} f[b]}{B}.$$

Roulette principle assigns each bee b a probability $P[b]$ to stay loyal to its solution. It is calculated as follows:

$$P[b] = \frac{f[b] - f_{min}}{f_{max} - f_{min}},$$

where $f_{min} = \min_{b=1..B} f[b]$ and $f_{max} = \max_{b=1..B} f[b]$. After vector P is calculated, for each bee we generate a random number r from uniform distribution U(0,1). If $r \leq P[b]$, bee b stays loyal, otherwise it becomes an uncommitted follower.

Recruiting is the final stage of backward pass where each uncommitted follower chooses which loyal bee to follow. Note that if bee b_1 follows b_2, b_1 overtakes the solution advertised by b_2. This is done using roulette wheel principle, where probability $RP[b]$ that loyal bee b is followed by some uncommitted follower is calculated as follows:

$$RP[b] = \frac{f[b]}{\sum_{i \in Loyal} f[i]}$$

Selecting solution for the next iteration is the final part of each iteration. It is implemented as follows: for each iteration we check whether the global best solution has been improved in that iteration. If the global best solution has not been improved in past *unsuccess_max* iterations, we initialize new solution and assign it to each bee. Otherwise, each bee is assigned the global best solution. Note that *unsuccess_max* is an input parameter for BCO.

4 Experimental Evaluation of BCOi for Feature Selection

The proposed algorithms are implemented in Python using Jupyter Notebook. Libraries which are mostly used are sklearn, numpy and pandas. They are run on OS Ubunutu 20.04 on Intel(R) i7-1065G7 x86_64-based processor with 8GB RAM.

4.1 Databases

In order to compare these four approaches we tested each of them on the examples from UCI Repository of Machine Learning Databases [3]. These databases are chosen because they are suitable for classification, values of attributes are real numbers with no missing values and the number of attributes is larger than 10. The characteristics of used databases are summarized in Table 1.

Table 1. Description of the databases: number of objects, attributes and classes

Name	#objects	#attributes	attribute types	#classes
Breast cancer	569	21	real	2
Cardiotography	2126	28	real	3
Image segmentation	2310	18	real	7
Connectionist bench	208	60	real	2
Parkinson	197	23	real	2

4.2 Results

Testing was done as follows. We set the parameters to be: $B = 8$, $NC = 5$, $unsuccess\_max = 5$ and stopping condition was maximal execution time of 20 seconds. Due to the stochastic nature of the BCO algorithm, for each combination of transformation type and loyalty decision type, we performed 100 repetitions and measure several characteristics:

- average fitness in 100 repetitions
- maximal fitness in 100 repetitions
- standard deviation of fitness in 100 repetitions
- average T-best, where T-best represents time in seconds when the best solution is obtained
- average frequency of repeating solutions

Note that combinations are named $X - Y$, where $X \in \{T, R\}$, stands for type of transformation and $Y \in \{R, M\}$ stands for making a loyalty decision. For example, R - M means roulette transformation combined with making a loyalty decision according to the mean.

In Table 2 we present average fitness in 100 repetitions for each database and we mark in bold the best results. We can see that the best results are obtained by both R-R and R-M in case of Breast cancer, Cardiography, Image Segmentation and Parkinson; and by R-R in case of Connectionist bench. Thus, we conclude that roulette transformation performs better than tournament transformation. Since there is a significant difference between these types of transformations in Image segmentation and Connectionist bench, there is a respectable difference between these types in average. However, if we fix the type of transformation, there is no a big difference whether we use roulette or mean as a type of making loyalty decision.

Table 2. Average fitness

Name	R - R	R - M	T - R	T - M
Breast cancer	**0.959**	**0.959**	0.955	0.955
Cardiotography	**0.837**	**0.837**	0.824	0.824
Image segmentation	**0.812**	**0.812**	0.787	0.786
Connectionist bench	**0.777**	0.775	0.693	0.696
Parkinson	**0.816**	**0.816**	0.815	0.814
Average	**0.840**	0.840	0.814	0.814

In Table 3, we present the maximal fitness obtained in 100 repetitions for each database. According to Table 3, the best solutions are obtained mostly by R-M, except in the case of Parkinson, in which the best solution was obtained by T-R. However, best fitness obtained by R-M is the highest in average.

Table 3. Best fitness

Name	R - R	R - M	T - R	T - M
Breast cancer	0.968	**0.970**	0.967	0.968
Cardiotography	0.848	**0.850**	0.845	0.845
Image segmentation	**0.815**	**0.815**	**0.815**	**0.815**
Connectionist bench	**0.822**	**0.822**	0.750	0.755
Parkinson	0.821	0.821	**0.836**	0.831
Average	0.854	**0.855**	0.843	0.843

We measured standard deviation of fitness in order to determine which algorithm is the most stable. According to Table 4, in each of these databases, difference in stability between R-R and R-M is not huge, as well as the difference between T-R and T-M. In Connectionist bench T-M is the most stable algorithm, while in other databases algorithms that use roulette transformation are the most stable. On databases Image segmentation and Parkinson, difference in stability depending on type of transformation is significant. As a result, the most stable algorithms in average is R-R.

Table 4. Standard deviation of fitness

Name	R - R	R - M	T - R	T - M
Breast cancer	**0.00405**	0.00480	0.00534	0.00602
Cardiotography	0.00542	**0.00514**	0.00728	0.00758
Image segmentation	0.00303	**0.00295**	0.02630	0.02955
Connectionist bench	0.02164	0.02344	0.02319	**0.02016**
Parkinson	**0.00179**	0.00189	0.00883	0.00986
Average	**0.00718**	0.00764	0.01419	0.01463

We measured time when the best solution is obtained, which we call T-best, and in the Table 5, we present average of T-best for each of the combinations. Again, we see that there is no significant difference when it comes to type of making loyalty decision. However roulette transformation is significantly faster than tournament transformation in all the data sets, except Cardiotography. Also in average, roulette transformation is faster.

Table 5. Average T-best (in seconds)

Name	R - R	R -M	T - R	T - M
Breast cancer	**10.825**	11.323	15.059	13.874
Cardiotography	9.729	10.329	**9.033**	9.174
Image segmentation	11.320	**10.557**	13.569	13.595
Connectionist bench	12.401	**12.071**	14.768	13.722
Parkinson	2.909	**2.636**	11.457	11.308
Average	9.439	**9.383**	12.777	12.335

We measured average frequency of repeating considered solutions. Since each of the combinations are running for the same time, combinations with less frequency of repeating the solutions considers a larger amount of solutions. From Table 6, we see that T-R and T-M consider more solutions than R-R and R-M in all data sets (except Breast Cancer) and in average. However, although they consider more solutions, R-R and R-M consider better solutions.

Table 6. Average frequency of repeating considered solutions

Naziv baze	R - R	R - M	T - R	T - M
Breast cancer	1.267	**1.246**	1.293	1.284
Cardiotography	1.389	1.333	1.199	**1.188**
Image segmentation	1.514	1.554	**1.374**	1.381
Connectionist bench	1.393	1.410	**1.203**	1.211
Parkinson	1.775	1.799	1.393	**1.375**
Average	1.468	1.468	1.292	**1.288**

5 Conclusion

In this paper an application of Bee Colony Optimization on Feature Selection problem is considered. In our interest were particularly FS problems in combination with clustering. As the clustering algorithm we used K-Means.

As BCOc has already been explored in literature, we used BCOi. This implies that we do not only add attributes to a solution during Forward Pass, we can also remove them. Therefore, instead of constructing the solution, we transform it, aiming to improve the global best solution. Two different types of transformation and making decisions about loyalty are compared on five different databases from UCI Machine Learning Repository.

From presented results, we can conclude that there is no big difference whether we make loyalty decision according to mean, or using roulette principle. However, we see that roulette transformation better performed than tournament transformation. It provides us with better solutions, it is faster and more stable. However, tournament transformation repeats the solution less frequently, which means that it explores more solutions, but still does not manage to find better solutions than roullete transformation.

References

1. G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.
2. T. Davidović, D. Teodorović, and M. Šelmić. Bee colony optimization Part I: The algorithm overview. *Yugoslav Journal of Operational Research*, 25(1):33–56, 2015.
3. Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.
4. R. Forsati, A. Moayedikia, and A. Keikha. A novel approach for feature selection based on the bee colony optimization. *International Journal of Computer Applications*, 43(8):13–16, 2012.
5. A. Moayedikia, R. Jensen, U. K. Wiil, and R. Forsati. Weighted bee colony algorithm for discrete optimization problems with application to feature selection. *Engineering Applications of Artificial Intelligence*, 44:153–167, 2015.
6. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
7. D. Teodorović, M. Šelmić, and T. Davidović. Bee colony optimization Part II: The application survey. *Yugoslav Journal of Operational Research*, 25(2):185–219, 2015.