XLIX International Symposium on Operational Research
SYM-OP-IS 2022
Vrnjačka Banja, 19 - 22.09.2022.
UNIVERSITY OF BELGRADE
Faculty of Economics and Business

BEE COLONY OPTIMIZATION FOR MULTI-LABEL FEATURE SELECTION

LUKA MATIJEVIĆ[1]

[1] Mathematical Institute of the Serbian Academy of Sciences and Arts, luka@mi.sanu.ac.rs

Abstract: *In this paper, we consider the problem of feature selection for multi-label data. Multi-label feature selection is a process of finding the appropriate subset of features that allows multi-label classifiers to find better solutions in a shorter amount of time. For this purpose, we developed the Bee Colony Optimization algorithm based on mutual information and compared it with other metaheuristics from literature, i.e. Ant Colony Optimization and Memetic Algorithm. After testing it on several benchmark instances, we concluded that our approach outperforms the other two methods.*

Keywords: *Combinatorial optimization, Metaheuristics, Mutual information, Classification*

1. INTRODUCTION

In recent years, machine learning (**ML**) and data mining techniques have become invaluable tools in business, medicine, in banking and finance, and many other professional areas. With the evergrowing amount of data and its dimensionality, it is becoming increasingly important to properly select the appropriate subset of features that will allow the aforementioned techniques to provide users with good quality predictions in a reasonable amount of time.

The *Feature Selection* (**FS**) problem aims at reducing the dimensionality of the data by removing less relevant features. *Multi-label FS* is a more general case of FS, in the sense that each object in the data can have multiple labels associated with it. There are two ways of approaching this problem: transforming the multi-label data into single-label data and applying classical FS, or constructing algorithms that can directly deal with multi-label data. In both cases, there are three main types of methods: *filter*, *wrapper*, and *embedded* methods. Filter methods use statistical techniques to reduce the number of features, without evaluating the result with some specific ML model. In contrast, wrapper methods use ML models to evaluate every considered solution. Embedded methods reduce the number of features during the learning process itself. In this paper, we are only interested in wrapper methods, or, more precisely metaheuristics.

The main contribution of this paper is in the development of the *Bee Colony Optimization* (**BCO**) algorithm for the multi-label FS problem. We also presented a stochastic way of adding and removing features from the current solution, that can be utilized in other metaheuristics as well.

This paper is organized as follows: In Section 2 we present the formulation of the problem, followed by the relevant literature in Section 3. In Section 4 we provide a detailed description of our method, and in Section 5 we present the results of experimental evaluation.

2. PROBLEM FORMULATION

Multi-label Feature Selection (**MLFS**) problem can be formulated in the following manner:

► **Definition 1.** Let $\mathcal{D}$ be a dataset where each row has a finite set of features $S = \{s_1, s_2, \cdots, s_n\}$ and $\mathcal{M}$ is a specific machine learning model. The objective is to determine a feature subset S_k of size k ($k < n$) where

$$\max_{S_k \subset S} Accuracy(\mathcal{M}(\mathcal{D}[S_k])) \ , \tag{1}$$

$\mathcal{D}[S_k]$ being the transformation of the original dataset. Only the features from S_k are present in $\mathcal{D}[S_k]$.

The function *accuracy* can be defined as follows.

► **Definition 2.** Let us assume that dataset $\mathcal{D}$ consists of n test instances (x_i, Y_i), $i = 1...n$, where Y_i is a subset of labels associated with instance x_i. We will denote the predicted set of labels as Z_i for each test instance. We define classification accuracy as:

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n} I(Z_i = Y_i) \tag{2}$$

where $I(true) = 1$ and $I(false) = 0$.

We can substitute the function *Accuracy* for any other appropriate multi-label metric, such as *Hamming-Loss* or *Precision*. Some of these metrics are described in a paper by Tsoumakas et al. (2009) [16].

3. RELATED WORK

There have been several attempts during recent years to apply metaheuristic methods to multi-label feature selection.

One of the first papers published on this topic was by Zhang et al. (2009) [18], who used the *Principal Component Analysis* (**PCA**) to reduce the number of irrelevant features before applying a *Genetic algorithm* (**GA**). Shao et al. (2013) [14] used mutation-based *simulated annealing* (**SA**), combined with GA and *greedy hill-climbing* algorithm. Similarly to [18], Yu et al. (2014) [17] discarded irrelevant features by using a forward search strategy, before applying the GA. In research by Lee and Kim (2015) [8], a *Memetic algorithm* (**MA**) is proposed, using a local search based on *mutual information*. Jungjit and Freitas (2015) [7] used a basic GA but introduced a different fitness function based on the correlation between features and labels and between pairs of features. *Particle Swarm Optimization* (**PSO**) is utilized by Zhang et al. (2017) [19] for MLFS, combined with a local learning strategy. In a study by Dowlatshahi et al. (2017) [5], the authors proposed a novel approach called *Epsilon-Greedy Swarm Optimizer*. Paniri et al. (2019) [13] utilized the *Ant Colony Optimization* (**ACO**) algorithm for MLFS.

4. PROPOSED METHOD

In this paper, we propose the *Bee Colony Optimization* based on the improvement concept (**BCOi**) for finding a satisfactory solution to MLFS. BCO is a stochastic, nature-inspired, population based metaheuristic, first introduced by Lučić and Teodorović [9, 10, 11], and since then it has been successfully applied to many optimization problems [1, 3, 4]. A more detailed description of the algorithm and its application can be found in papers [2, 15].

Firstly, we adopt the function $Q(f)$ from paper [8] for evaluating the influence of features on the current solution (Equation 3). In this function, f stands for a specific feature we want to evaluate, Y is a set of all labels, and S_k is the set of already selected features. The function $I(f,h)$ is approximated mutual information, presented in Equation 4. $H(f)$ represents the entropy of variable f, while $H(f,h)$ is the joint entropy of variables f and h. With this in mind, we can construct stochastic methods for adding and removing features from the solution.

$$Q(f) = \sum_{y_j \in Y} I(f, y_j) - \sum_{f_j \in S_k} I(f, f_j) \tag{3}$$

$$I(f,h) = H(f) + H(h) - H(f,h) \tag{4}$$

When adding a new feature, we first evaluate every feature f that is not present in the current solution, by calculating the $Q(f)$. Instead of adding the feature with the best (highest) value of $Q(f)$, we opted for a stochastic approach, inspired by the GRASP metaheuristic [6]. Therefore, we detect α best features and choose one of them at random to add it to the current solution. This method is presented in Algorithm 1. Likewise, we remove a feature from the current solution in a similar manner, by calculating $Q(f)$ for all of the features present in the current solution, detecting α of those with the lowest value, and choosing one at random (Algorithm 2). The value of α is provided as a hyperparameter.

Algorithm 1 Procedure that adds a feature into the set of selected features

1: **procedure** ADD_FEATURE($features$, α)
2: $A \leftarrow F \setminus features$ ▷ F is the set of all the possible features
3: **for** $\forall f \in A$ **do**
4: Evaluate $Q(f)$
5: **end for**
6: $f \leftarrow$ randomly choose one of α best evaluated features ▷ Higher $Q(f)$ indicates a better feature
7: $features \leftarrow features \cup \{f\}$
8: **return** $features$
9: **end procedure**

Algorithm 2 Procedure that removes a feature from the set of selected features

1: **procedure** DEL_FEATURE($features, alpha$)
2: **for** $\forall f \in features$ **do**
3: Evaluate $Q(f)$
4: **end for**
5: $f \leftarrow$ randomly choose one of α worst evaluated features ▷ Lower Q(f) indicates a worse feature
6: $features \leftarrow features \setminus \{f\}$
7: **return** $features$
8: **end procedure**

In Algorithm 3, we provided an overall structure of our BCOi algorithm. First, we initialize all the bees with a starting solution (Line 2). The starting solution is generated by randomly choosing one feature and then calling *ADD_FEATURE* procedure $(n-1)$ times, where n is the preset number of features in the solution. After each bee is initialized with a solution and each solution is evaluated, the best solution among them is memorized (Line 4). The evaluation is performed by transforming the dataset so that it consists only of features present in the solution, invoking the *Multi-label K-nearest neighbors* algorithm on the testing part of the dataset, and applying the aforementioned metric to the results. The main part of the algorithm consists of two steps, repeated iteratively: forward pass (Lines 6-12) and backward pass (Lines 13-16). During the forward pass, each bee transforms its solution, evaluates it, and if it proves better than the current best solution, the newfound solution becomes the new current best solution. The transformation of a solution is done by applying *ADD_FEATURE* function k times, followed by invoking *DEL_FEATURE* function k times. The argument k is determined dynamically, based on the number of consecutive iterations in which none of the bees found a new best solution. The idea for this approach comes from the *Variable Neighborhood Search* metaheuristic (**VNS**) [12], where the size of the neighborhood is increased after each non-improving iteration. In the backward pass of the algorithm, each bee decides whether it is going to stay loyal to its solution and explore it further, or to adopt a solution from some other bee. The probability of a bee staying loyal to its solution is calculated as:

$$p_i = \frac{v_i - v_{min}}{v_{max} - v_{min}} \tag{5}$$

where v_i is the quality of the solution of the $i-th$ bee, v_{max} and v_{min} are qualities of the best and the worst solutions among all the bees, respectively. This way, we can guarantee that at least one bee is going to stay loyal to its solution. After each bee decided on its loyalty, the bees that abandoned their solution have to choose some loyal bee and adopt its solution (Line 16). The probability of each loyal bee being selected is calculated as:

$$p_i = \frac{v_i}{\sum_{v_j \in B_l} v_j} \tag{6}$$

where v_i is the quality of the solution of the $i-th$ bee and B_l is the set of loyal bees. Therefore, a loyal bee with a better solution has a higher chance of recruiting disloyal bees. Finally, in order to prevent the algorithm from being stuck in the local optimum, we introduce the concept of reinitialization after a certain number of iterations without any improvement (Line 23).

Table 1: The selected parameter values

ACO	MA	BCOi
number_of_ants = 25	population_size = 15	number_of_bees = 30
$\beta = 0.8$	v = 500	$\alpha = 5$
$\rho = 0.1$	h = 15	$k_{max} = 5$
	crossover_probability = 0.5	
	mutation_probability = 0.1	

Algorithm 3 Bee Colony Optimization

procedure BCO($number\_of\_bees$, α, k_{max})
  $bees \leftarrow initialize(number\_of\_bees, \alpha)$
  $k \leftarrow 1$
  $best\_solution \leftarrow find\_best(bees)$
  while stopping criterion is not met **do**
    for $\forall b \in bees$ **do** ▷ Forward pass
      $transform(b, \alpha, k)$
      $value \leftarrow evaluate(b)$
      if $value > evaluate(best\_solution)$ **then**
        $best\_solution \leftarrow b$
      end if
    end for
    for $\forall b \in bees$ **do** ▷ Backward pass
      $decide\_loyalty(b)$
    end for
    $recruitment(bees)$
    if there was an improvement **then**
      $k \leftarrow 1$
    else
      $k \leftarrow k + 1$
    end if
    if $k > k_{max}$ **then**
      $bees \leftarrow initialize(number\_of\_bees, \alpha)$
    end if
  end while
  return $best\_solution$
end procedure

5. EXPERIMENTAL EVALUATION

To evaluate our approach, we implemented three algorithms in total: ACO presented in [13], MA presented in [8], and our own method (BCOi). All three algorithms were written in Python programming language and executed on a personal laptop with an Intel i7-10750H CPU and 32GB of RAM, under the Ubuntu 20.04 operating system. The algorithms were tested on four benchmark datasets for multi-label classification, available at `https://www.uco.es/kdis/mllresources/`. Since some of the datasets contained features with continuous values, those features were first discretized by dividing the interval into 10 bins. Each test was repeated 30 times with a different value for the random number generator seed.

The values of parameters were determined by using the *iRace*[1] package for R programming language with a budget of 200 tests. The meaning behind the parameters for ACO and MA can be found in the papers in which these methods were originally proposed for MLFS. In Table 1 we present the obtained values for each parameter.

As a stopping criterion, we used a limit of 500 calls to the fitness function, for all three algorithms, as the evaluation procedure is highly expensive in terms of computational power, especially for larger datasets.

[1] https://cran.r-project.org/web/packages/irace/index.html

Table 2: Average Accuracy over 30 independent runs

Dataset	Num. instances	Num. features	Num. labels	ACO	MA	BCOi
Emotions	593	72	6	0.2574 (0.0107)	0.3054 (0.0066)	**0.3115 (0.0095)**
Birds	645	260	19	0.5108 (0.0039)	0.5012 (0.0045)	**0.5158 (0.0068)**
Yeast	2417	130	14	0.1586 (0.0037)	0.1747 (0.0073)	**0.1759 (0.0079)**
Scene	2407	294	6	0.4153 (0.0151)	0.5099 (0.0052)	**0.5279 (0.0110)**

Furthermore, we wanted to emphasize the differences in performance between metaheuristics while keeping the fitness function budget relatively low. We suggest that in practice this limit should be set higher if the user has the necessary resources, in order for algorithms to obtain better quality solutions. Larger limits have been tested on the smallest dataset *Emotions* (1000, 1500, and 2000 iterations), and we exhibited the same trend as the one presented in Figure 1.

In Table 2 we presented the obtained results for each dataset under consideration. The first column represents the name of the dataset, the second, third, and fourth show the number of instances, features, and labels respectively, whereas the three last columns present the average accuracy over 30 runs, with standard deviation presented in parenthesis. We can clearly see that BCOi outperformed the other two algorithms in terms of average accuracy for all datasets.

In Figure 1 we present a boxplot showing the performance of each algorithm over 30 independent runs for each dataset. Furthermore, we performed *Wilcoxon pair-wise statistical test* comparing BCOi with the other two algorithms, for each dataset separately. The results are presented in Table 3. Additionally, we calculated the 95% confidence interval of the location parameter for each of the tests performed and presented it below p-values in Table 3. With the significance level of $\alpha = 0.05$, we concluded that BCOi was statistically different from ACO in 3 out of 4 cases, and in 2 out of 4 cases compared to MA. These results support conclusions based on Figure 1.

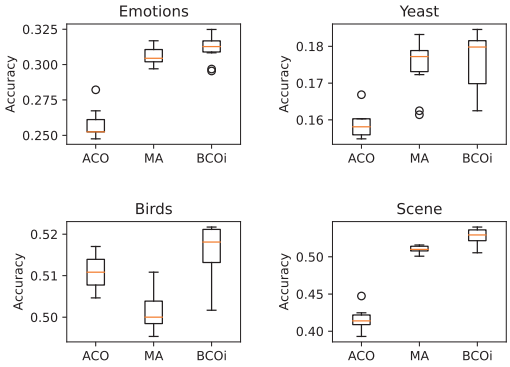

Figure 1 Boxplots showcasing the difference in the accuracy of tested algorithms on four datasets

Table 3: p-values of Wilcoxon pair-wise statistical tests and 95% confidence interval for location parameter comparing BCOi to other proposed methods

Dataset	ACO	MA
Emotions	**0.0001717 [0.04438175, 0.06411505]**	0.1609 [-0.00155461, 0.01450039]
Birds	0.06312 [-0.0004934487, 0.0107584934]	**0.0005501 [0.008805312, 0.020857824]**
Yeast	**0.0003197 [0.009839083, 0.024617684]**	0.4268 [-0.006605605, 0.007226751]
Scene	**0.0001242 [0.1034662, 0.1247954]**	**0.001345 [0.01040258, 0.02699941]**

6. CONCLUSION

In this paper, we examined metaheuristic approaches to multi-label feature selection problems. We presented a version of Bee Colony Optimization based on mutual information and compared it to methods already present in the literature, i.e. Ant Colony Optimization algorithm and Memetic algorithm. Experimental evaluation was performed on four benchmark datasets.

We demonstrated that BCO was statistically better than ACO in 3/4 cases and better than MA in 2/4 cases, while performing relatively the same in relation to the rest of the cases. Future research should include testing these algorithms on larger datasets and potentially finding an alternative to calculating the mutual information, given that it is a computationally expensive task.

Acknowledgement

This work was supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia, Agreement No. 451-03-9/2021-14/200029 and by the Science Fund of the Republic of Serbia, Grant *"AI4TrustBC: Advanced Artificial Intelligence Techniques for Analysis and Design of System Components Based on Trustworthy BlockChain Technology"*.

REFERENCES

[**1**] Davidović, T., Ramljak, D., Šelmić, M., & Teodorović, D. (2011). *Bee colony optimization for the p-center problem*. Computers & Operations Research, 38(10), 1367-1376.

[**2**] Davidović, T., Teodorović, D., & Šelmić, M. (2015). *Bee colony optimization-part I: the algorithm overview*. Yugoslav Journal of Operations Research, 25(1), 33-56.

[**3**] Davidović, T., Šelmić, M., Teodorović, D., & Ramljak, D. (2012). *Bee colony optimization for scheduling independent tasks to identical processors*. Journal of heuristics, 18(4), 549-569.

[**4**] Dimitrijević, B., Teodorović, D., Simić, V., & Šelmić, M. (2012). *Bee colony optimization approach to solving the anticovering location problem*. Journal of Computing in Civil Engineering, 26(6), 759-768.

[**5**] Dowlatshahi, M. B., Derhami, V., & Nezamabadi-pour, H. (2017). *Ensemble of filter-based rankers to guide an epsilon-greedy swarm optimizer for high-dimensional feature subset selection*. Information, 8(4), 152.

[**6**] Feo, T. A., & Resende, M. G. (1995). *Greedy randomized adaptive search procedures*. Journal of global optimization, 6(2), 109-133.

[**7**] Jungjit, S., & Freitas, A. A. (2015, April). *A new genetic algorithm for multi-label correlation-based feature selection*. In 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (pp. 285-290).

[**8**] Lee, J., & Kim, D. W. (2015). *Memetic feature selection algorithm for multi-label classification*. Information Sciences, 293, 80-96.

[**9**] Lučić, P., & Teodorović, D. (2001, June). *Bee system: modeling combinatorial optimization transportation engineering problems by swarm intelligence*. In Preprints of the TRISTAN IV triennial symposium on transportation analysis (pp. 441-445).

[**10**] Lučić, P., & Teodorović, D. (2002, November). *Transportation modeling: an artificial life approach*. In 14th IEEE International Conference on Tools with Artificial Intelligence, 2002.(ICTAI 2002). Proceedings. (pp. 216-223). IEEE.

[**11**] Lučić, P., & Teodorović, D. (2003). *Computing with bees: attacking complex transportation engineering problems*. International Journal on Artificial Intelligence Tools, 12(03), 375-394.

[**12**] Mladenović, N., & Hansen, P. (1997). *Variable neighborhood search*. Computers & operations research, 24(11), 1097-1100.

[**13**] Paniri, M., Dowlatshahi, M. B., & Nezamabadi-Pour, H. (2020). *MLACO: A multi-label feature selection algorithm based on ant colony optimization*. Knowledge-Based Systems, 192, 105285.

[**14**] Shao, H., Li, G., Liu, G., & Wang, Y. (2013). *Symptom selection for multi-label data of inquiry diagnosis in traditional Chinese medicine*. Science China Information Sciences, 56(5), 1-13.

[**15**] Teodorović, D., Šelmić, M., & Davidović, T. (2015). *Bee Colony Optimization-part II: The application survey*. Yugoslav Journal of Operations Research, 25(2), 185-219.

[**16**] Tsoumakas, G., Katakis, I., & Vlahavas, I. (2009). *Mining multi-label data*. Data mining and knowledge discovery handbook, 667-685.

[**17**] Yu, Y., & Wang, Y. (2014, October). *Feature selection for multi-label learning using mutual information and GA*. In International Conference on Rough Sets and Knowledge Technology (pp. 454-463). Springer, Cham.

[**18**] Zhang, M. L., Peña, J. M., & Robles, V. (2009). *Feature selection for multi-label naive Bayes classification*. Information Sciences, 179(19), 3218-3229.

[**19**] Zhang, Y., Gong, D. W., Sun, X. Y., & Guo, Y. N. (2017). *A PSO-based multi-objective multi-label feature selection method in classification*. Scientific reports, 7(1), 1-12.