Süleyman Demirel Üniversitesi
Fen Bilimleri Enstitüsü Dergisi
Cilt 24, Sayı 1, 120-131, 2020

Süleyman Demirel University
Journal of Natural and Applied Sciences
Volume 24, Issue 1, 120-131, 2020

DOI: 10.19113/sdufenbed.635465

A Locally Searched Binary Artificial Bee Colony Algorithm Based on Hamming Distance for Binary Optimization

Zeynep Banu ÖZGER*[1], Bülent BOLAT[2], Banu Diri[3]

[1]Sütçü İmam Üniversitesi, Mühendislik Mimarlık Fakültesi, Bilgisayar Mühendisliği Bölümü, 46040, Kahramanmaraş, Türkiye
[2]Yıldız Teknik Üniversitesi, Elektrik-Elektronik Fakültesi, Elektronik ve Haberleşme Mühendisliği Bölümü, 34220, İstanbul, Türkiye
[3]Yıldız Teknik Üniversitesi, Elektrik-Elektronik Fakültesi, Bilgisayar Mühendisliği Bölümü, 34220, İstanbul, Türkiye

(Alınış / Received: 21.10.2019, Kabul / Accepted: 03.02.2020, Online Yayınlanma / Published Online: 20.04.2020)

Keywords
Artificial bee colony,
Data mining,
Heuristic algorithms,
Machine learning

Abstract: Artificial Bee Colony is a population based, bio-inspired optimization algorithm that developed for continues problems. The aim of this study is to develop a binary version of the Artificial Bee Colony (ABC) Algorithm to solve feature subset selection problem on bigger data. ABC Algorithm, has good global search capability but there is a lack of local search in the algorithm. To overcome this problem, the neighbor selection mechanism in the employed bee phase is improved by changing the new source generation formula that has hamming distance based local search capacity. With a re-population strategy, the diversity of the population is increased and premature convergence is prevented. To measure the effectiveness of the proposed algorithm, fourteen datasets which have more than 100 features were selected from UCI Machine Learning Repository and processed by the proposed algorithm. The performance of the proposed algorithm was compared to three well-known algorithms in terms of classification error, feature size and computation time. The results proved that the increased local search ability improves the performance of the algorithm for all criteria.

Hamming Mesafesi ile Lokal Arama Tabanlı İkili Yapay Arı Kolonisi Algoritması

Anahtar Kelimeler
Yapay arı kolonisi,
Veri madenciliği,
Sezgisel algoritmalar,
Makine öğrenmesi

Özet: Yapay Arı Kolonisi Algoritması sürekli uzay problemleri için geliştirilen, popülasyon tabanlı, doğadan esinlemeli bir optimizasyon algoritmasıdır. Bu çalışmanın amacı, büyük veride, öznitelik alt küme seçimi problemini efektif bir biçimde çözmek için Yapay Arı Koloni (YAK) Algoritmasının ikili bir versiyonunu geliştirmektir. YAK Algoritması başarılı bir global yakınsama sunmakla birlikte lokal bölgedeki olası çözümleri gözden kaçırabilmektedir. Algoritmanın komşu kaynak seçimi mekanizmasına, Hamming Mesafe ölçümü tabanlı bir yerel arama prosedürü eklenmiştir. Ayrıca, yeniden nüfus stratejisi ile popülasyonun çeşitliliği artırılmış ve erken yakınsama önlenmiştir. UCI Makine Öğrenmesi Havuzu'ndan, öznitelik sayısı 100'den fazla olan 14 veri kümesi seçilmiş ve önerilen yöntem ile öznitelik seçimi yapılmıştır. Algoritmanın performansı, yaygın kullanılan ve başarısı kanıtlanmış üç sezgisel algoritma ile sınıflandırma hatası, seçilen öznitelik sayısı ve hesapsal maliyet bakımından karşılaştırılmıştır. Elde edilen sonuçlar, YAK algoritmasına entegre edilen lokal arama prosedürünün, algoritmanın performansını tüm kriterler bakımından artırdığını göstermektedir.

1. Introduction

The process of selecting the distinctive features (attributes) of a given dataset is called feature subset selection [1], or simply features selection. Unlike the process of feature extraction, a feature selection process finds the most relevant features of the data without using a transformation and removes all unnecessary features. Three common feature selection methods are reported in the literature:

* Corresponding author: zeynepozger@ksu.edu.tr

filters, wrappers and hybrid methods. Filters select the most relevant top n features according to an arbitrary quality score such as t-score, p-stat, etc., but they do not consider the relationships between features. The computational cost of filters is low; however, their performance is not sufficient for most applications [2]. Furthermore, when the datasets differ in size and complexity, it is a problem to determine what the value of n will be. Wrappers consist of an induction algorithm to select a subset of the features and a learner to produce feedback for the induction algorithm. Wrappers yield better performance in terms of computational complexity. For example, if the data consists of n features, 2n possible subsets need to be searched. Especially for very large datasets, it is not possible to search the entire set of possibilities [3]. To overcome this problem, hybrid methods have been developed. A hybrid feature selection method first reduces the number of features to a reasonable value by using a filter. Then, the best subset is determined by a wrapper.

The feature selection process has two main steps: subset generation and subset evaluation. Subset generation is basically a search procedure. Since heuristic search algorithms do not need to search the entire space, they are widely used in wrappers to overcome the curse of dimensionality. Particle swarm optimization [4-6], genetic algorithms [7, 8], genetic programming [9], ant colony optimization [10-12], the Bat algorithm [13], tabu search [14, 15], simulated annealing [16, 17] and differential evolution [18, 19] are some examples of heuristic algorithms that have been applied to feature selection problems.

If an algorithm produces multiple solutions and tries to improve the solutions iteratively, it is called a population-based algorithm. Artificial Bee Colony (ABC) is a population-based swarm intelligence algorithm that simulates the foraging behaviour of honeybees. A bee colony includes three types of bees: employed bees, onlooker bees and scout bees. Communication between the employed and onlooker bees is based on sharing information the quality of the food source. Scout bees are responsible for searching for new sources when a food source is abandoned.

ABC is a stochastic optimization technique that operates on continuous space. In this study the feature selection problem is tackled at binary space. Therefore, it is necessary either to transform the result vector to binary space using a binarization function, or to adapt the equations in the employed bee phase to work with binary vectors. The literature reports several binary versions of ABC algorithms. Ozturk et al. [20] presented an algorithm that uses binary vectors and a cross-over genetic operator to solve the dynamic image clustering and 0-1 knapsack problems. Jia et al. [21] utilized bitwise AND, OR, and XOR operators to generate new sources for solving some benchmark functions. Kiran and Gunduz [22] also employed the bitwise XOR operator for the uncapacitated location problem at binary space. Kashan et al. [23] proposed a distance-based ABC algorithm called DisABC. They preferred the Jaccard Coefficient measure on binary vectors for solving pure binary optimization problems. Ozturk et al. [24] added genetic cross-over, swap and selection operators to DisABC and named their variant as IDisABC. IDisABC is used for dynamic clustering problems. Because of its effectiveness, Hancer et al. [25] hybridized DisABC with the Differential Evolution algorithm (MDisABC) for feature selection problems. Singhal et al. [26] also utilized DisABC to produce new binary sources for solving the unit commitment problem. They modified the Ø vector, which is a component of the production mechanism for new sources. Additionally, in order to increase the convergence rate, the abandoned source is determined as the global best vector. Yurtkuran and Emel [27] determined different cross-over operators for producing new sources both in the employed and scout bee phases. Zhang and Zhang [28] also used binary vectors as food sources to construct spanning trees in vehicular ad hoc networks. In this work, to produce a candidate solution in the employed bee phase, two random bits are selected from the current source, and then they are inverted. Zhang and Gu [29] proposed a combination of the insertion and swap operators to solve the flow shop scheduling problem. Tasgetiren et al. [30] improved the previous work by adding a local search mechanism in the employed and onlooker bee phases. The algorithm proposed by Zhang and Ye [31] produced a new source by performing a local search around the current source, and then selecting the best one in the employed bee phase. Ye and Chen [32] proposed a method that uses some local search strategies based on velocity computation for solving the minimum attribute reduction problem. Ribas et al. [33] also used the local search strategy in their study for the blocking flow shop problem. Han et al. [34] developed a differential evolution-based binary ABC algorithm for the flow shop scheduling problem. Schiezaro and Pedrini [35] utilized a perturbation parameter in the employed bee phase. In this method, a random number from a distribution between 0 and 1 is generated for each position of the current source. If this random value is greater than a predefined perturbation parameter, the bit is set as 1. Ozmen et al. [36] produced new sources in discrete space with substitution and shift operators.

The methods mentioned above use binary vectors and/or impose changes on the component of the ABC algorithm that produces the new source. However, some other methods use continuous vectors, and then transform the results to the binary space. Wei and Hanning [37] used the round function for four

benchmark functions and named it BABC. Tran and Wu [38] modified BABC by changing the transformation function: they preferred a sigmoid function and expanded the threshold values. While Kiran [39] used the mod and the round operators together for solving the uncapacitated facility location problem, Mandala and Gupta [40] utilized the tangent function to transform the continuous results into binary space.

ABC is a strong optimization algorithm. Therefore it applied to many optimization problems. To avoid falling to local optimum, the local search capability of the algorithm is limited. Candidate solutions are generated somewhere between two sources. We examined that, because of this strategy, sometimes causes to lose a good solution. This issue motivated us to adopt a local search procedure in the algorithm. Our aim was to add a local search procedure to the algorithm without losing its heuristic.

The goal of this study is to develop a binary version of the ABC algorithm to solve the feature subset selection problem on large datasets. To achieve this goal, BitABC [21], which is a binary variant based on bitwise operators, is selected as a starting point with reference to our previous studies [41]. To improve the local search ability of this algorithm, the neighbour selection mechanism in the employed bee phase is improved by changing the new source generation formula and by using multiple candidates for new sources. To improve exploration capacity, the initialization phase is updated. Additionally, a repopulation strategy is provided to increase the diversity of the population. To measure the effectiveness of the proposed algorithm, fourteen datasets that each has more than 100 features were selected from the UCI Machine Learning Repository and processed by the proposed algorithm. The performance of the proposed algorithm was compared to that of three well-known heuristic algorithms.

Organization of the paper as follows. Section 2 introduces the principles of ABC, Binary ABC, and BitABC. The proposed method is introduced in the Section 3. Section 4 summarizes the experimental results. Finally, Section 5 devotes to the concluding remarks.

2. Material and Method

In this section the background of study is presented.

2.1. Artificial bee colony algorithm

ABC is a bio-inspired swarm intelligence algorithm proposed by Karaboga in 2005 [42]. The algorithm was developed to solve multi-dimensional optimization problems by modelling the foraging, learning, and knowledge sharing behaviors of honey bees. Typically, during the food searching process three distinct bee types are present. Employed bees exploit their sources and provide information to onlooker bees about the food sources. The onlooker bees select a food source and try to optimize it. Scout bees search new sources randomly. The food sources represent possible solutions, and the quality of a solution is indicated by the amount of nectar. Each food source is exploited by one employed bee and optimized by one onlooker bee; therefore, the number of the food source is equal to the number of the employed and onlooker bees. The original ABC algorithm includes 4 phases:

1. Initialization phase: Suppose that there are N bees in the hive and each bee represents a solution that is expressed by D parameters. The initial food sources are generated randomly by (1).

$$x_{ij} = x_j^{min} + U(0,1)(x_j^{max} - x_j^{min}) \quad (1)$$

Where $x_i=\{x_{i1}, x_{i2}...x_{ij}...x_{id}\}$is the i^{th} food source, U(0, 1) is a uniformly distributed random variable between 0 and 1, and x_j^{min} and x_j^{max} are the minimum and maximum values of parameter j.

The current source number is represented by i, and j is the dimension of the vector.

2. Employed bee phase: Each employed bee directs to a food source and then finds a new source in the vicinity of the current source by using (2).

$$v_{ij} = x_{ij} + \emptyset_{ij}(x_{ij} - x_{kj}) \quad (2)$$

Where i is the index of the current source, k is the index of a randomly selected neighbor source, and $\emptyset_{ij}$ is a uniformly distributed random number between -1 and 1. If the fitness value of the new source is better than the previous one, the employed bee memorizes it, sets the number of trials to zero, and abandons the previous source. Otherwise, the number of trials of the current source is increased by 1 until a predefined maximum value is reached.

3. Onlooker bee phase: Once the employed bees provide information to the onlooker bees about the location and the quality of the food sources, each onlooker bee selects a source by using the roulette-wheel method (3). According to this equation, if the nectar volume of a source is high, it has a high probability of being selected. In other words, onlooker bees make a greedy selection according to information provided by employed bees. The selection probability of the i^{th} source is calculated as follows:

$$p_i = \frac{fitness_i}{\sum_{j=1}^{N} fitness_j} \quad (3)$$

where 'fitness$_i$' is the fitness value of 'x_i' and 'N' is the total number of the food sources. After calculating the probabilities of all sources, a uniformly distributed random value between 0 and 1 is generated. If p_i is greater than the random value, a new source is generated using equation (2) and its fitness value is calculated. An onlooker bee selects the new source if it has better fitness value than the current source, and the onlooker bee becomes an employed bee for the next iteration.

4. Scout bee phase: There is a pre-defined threshold value for the number of trials of the sources. When the number of trials of a source exceeds this threshold value, it is assumed that this source is a local optimum, and the source is abandoned. If a food source is abandoned, the bee that had been assigned to it becomes a scout bee, and a new source is generated by using (1) for the scout bee.

2.2. Binary artificial bee colony algorithm

Initially, the ABC algorithm was designed to solve optimization problems effectively in continuous space. Hence, the original ABC algorithm is not suitable for binary optimization problems such as feature selection. There have been a few attempts to overcome this issue reported in the literature, as seen in Section 1. Their common properties are summarized as follows. The sources are represented by binary vectors, and the size of the vector is equal to the number of features. Logic 1 in this vector indicates that the corresponding feature is being selected, and logic 0 indicates the opposite. The fitness value of any source is calculated by using the classification accuracy of the feature subset determined by the source.

2.3. Bitwise operators ABC algorithm

Ji et al. [21] introduced a binary ABC algorithm (BitABC) based on the bitwise operators between binary vectors. In this work, the authors used binary vectors, and to generate a new source they used bitwise operators. The authors revised the initialization method and the new source generation strategy of the employed and onlooker bee phases. In the initialization phase, ABC uses the maximum and the minimum values of each parameter. However, if the problem comes from a binary space, any parameter can only be 0 or 1. Sources are initialized for BitABC with equation (4). According to this, a random value between 0 and 1 is produced, and if this value greater than a threshold value, corresponding dimension of vector is set to 1 otherwise it is set to 0.

$$x_{ij} = \begin{cases} 1, if\ rand(0,1) \geq threshold \\ 0, if\ rand\ (0,1) < threshold \end{cases} \quad (4)$$

ABC uses arithmetic operators for generating new sources. In binary space, arithmetic operators are not meaningful. To overcome this issue BitABC changed the new food source production method in the employed and onlooker bee phases by using binary operators instead of arithmetic operators.

2.4. The proposed method

In our previous paper [41], we applied binary ABC algorithms on the feature selection problem using 10 datasets. We have seen that one of these binary algorithms (BitABC) yielded better results in the feature selection process. Its better results and simplicity lead us to select it a starting point. The weakness of BitABC is its lack of local search ability.

The main goal of this study is to develop a binary ABC algorithm that can solve the feature selection problem effectively. As mentioned above, in our previous experiments we found that BitABC achieved better accuracy than the other binary ABC variants. However, because of its new source generation strategy, its local search capacity is limited. Hence, this deduction led us to improve the local search capacity of BitABC. In the ABC algorithm, local search is performed in the onlooker and employed bee phases. Then, we have modified the new source generation method of the employed bee step of BitABC.

The ABC algorithm's strategy for new source generation is as follows. In the i^{tch} iteration, a new source is generated by using the current source x_i, a randomly selected neighbor x_k and a random value Ø. Therefore, the new source should fall within the range between x_i and x_k. The $Ø_i$ term determines whether the new source is close to x_i or x_k. If the new source is far from the current source, the bee moves away from its own local area, and the current source is potentially abandoned, even it contains a promising amount of nectar. Abandoning a potentially good region causes the lack of local search ability of the algorithm. To improve the local search ability of ABC, we limited the distance between the current source and the new source. By using such restrictions the risk of abandoning a promising source is reduced. Since the position of a given source is represented by a binary string, Hamming distance is used as the distance metric. Additionally, since using arithmetic operators is not meaningful in binary space, we used bitwise operators like BitABC. The Ø vector is also a binary vector.

As mentioned above, the location of the new source is determined by the random Ø vector and a randomly selected neighbor. It is not possible to restrict the distance between the selected neighbor and the current source. For example, if the data have 100 features, the food sources must have 100 dimensions. Therefore, the Hamming distance between the

current source and the random neighbor should be any integer between 1 and 100, and using existing methods, there is no way to limit the distance to be lower than a certain number. However, using the BitABC algorithm, this can easily be controlled by modifying the new source generation formula that is used in BitABC. New source generation formula is updated with equation (5).

$$v_i = \emptyset_2 AND\ (\emptyset_1 AND\ (x_i\ OR\ x_k) \quad (5)$$

Where $\emptyset_1$ and $\emptyset_2$ are random vectors, x_i is the current source, x_k is a randomly selected neighbor and v_i is the new source. As an example, let x_i = {1 0 1 1 1 1 0 0 0 0 0 0} be the current source, x_k = {1 0 1 0 1 1 0 0 0 1 1 0} be the neighbor source, and Ø be a random binary vector, dH(x_i, x_k) = d where dH(x, y) is the Hamming distance between the binary vectors x and y, and d is an integer. To increase the diversity of the selected features, i.e. the position of 1s in the source vector, the bitwise 'OR' operator is applied to the current sources and the neighbor sources using equation (6).

$$y = x_i OR\ x_k = \{1\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 0\} \quad (6)$$

This operation increases the diversity of the selected features; however, the resulting vector y has greater than or equal to the number of 1s in x_i. Since the main goal of the feature selection process is to find the minimal set of relevant features, this result is inappropriate. Another problem is that, after such operation, the distance between x_i and x_k increases, and the ability to search locally is reduced. To resolve these problems, a random $\emptyset_1$ vector is used. If dH(x_i, x_k)) = r, then $\emptyset_1$ is produced by altering random d bits of x_i. Let d = 5 and $\emptyset_1$ ={0 0 1 1 1 1 0 1 1 0 1 1}; then y_2 is found using equation (7):

$$y_2 = \emptyset_1 AND\ (x_i\ OR\ x_k) = \{0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\} \quad (7)$$

Where dH(x_i, y_2) = 1 < d. However, after simulation experiments we saw that if another random term, e.g. $\emptyset_2$, is used, it is possible to find better new sources. This new $\emptyset_1$ vector is derived by altering r bits of $\emptyset_1$. Again, let d = 5 and $\emptyset_2$ ={0 1 1 0 0 0 0 1 1 0 0 1}. By using equation (5), the final solution of the new source v_i becomes v_i= {0 0 1 0 0 0 0 0 0 0 0 0}. The distance between v_i and x_i is a Gaussian-like distributed variable between 1 and d. In a feature selection problem, identifying distinctive feature subgroups is important. By using (5) at the employed bee phase, we provided that there are small changes between the previous and current sources. Thus, the algorithm is able to identify features that are highly discriminatory when combined together.

As a final improvement, to increase the local search ability of the ABC algorithm, five new candidate sources were generated for each employed bee within a maximum distance of r from the current source, and the best one was selected. This step concludes the first version of the proposed method. Following preliminary experiments, a new version of this method has been developed. If the fitness of the current source is better than all of the new candidate sources, it means that the current source is a local optimum. If d is chosen as a small value, then the new candidates originate from narrow vicinity, and it is possible to miss a better solution due to the highly localized nature of the search algorithm. To fix this problem, a small change has been made to the proposed algorithm. First, the number of the candidate sources is reduced to three. If these three candidates cannot produce a better solution than the current source, a fourth candidate is produced by using a new source generation formula (8) of BitABC because this equation always produces a new source that has a larger distance from the current one.

$$v_i = x_i XOR\ (\emptyset_i\ AND\ (x_i\ OR\ x_k)) \quad (8)$$

If the fitness of the fourth candidate is better than the current, the new solution of this candidate becomes the current source for the next step. Otherwise, the number of trials parameter of the i^{th} source is increased by four (four new sources are generated). The flowchart of the proposed method is shown in Figure 1. We named this method exBitABC.

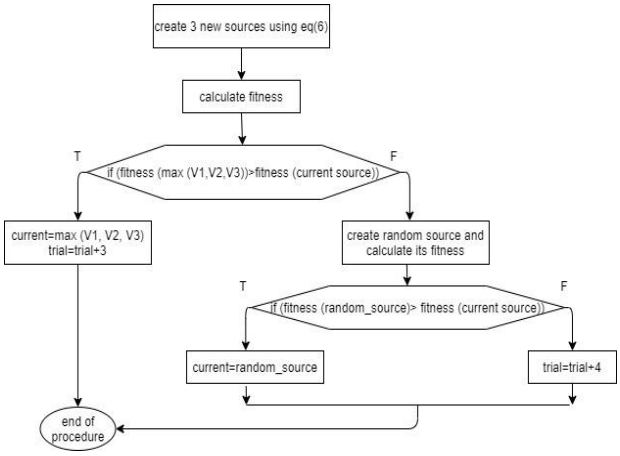


Figure 1. Flow chart of the new source generation procedure of exBitABC

The initial food sources are determined randomly by the ABC algorithm. In this way, any two sources are close to each other. Namely, the Hamming Distance between any two sources can be a small value. Our purpose is to increase the search capacity in the local regions around the sources. If two or more sources are close to each other, the regions being searched will also overlap. To overcome this problem, the initialization strategy is updated: The search space is divided into regions, and the number of regions is equal to the number of food sources. In this way, the distance between sources is equal. For the feature selection problem, the size of the search space is equal to the feature size of the dataset. Consequently, the distance between any two sources is determined

by using equation (9). Each employed bee is responsible for a single specified region when iterations are initiated.

$$minDistance = \frac{featureSize}{\#sources} \quad (9)$$

The roulette-wheel method used in the onlooker bee phase directs more bees to better sources. However, this strategy may cause a decrease of diversity over time. Since the proposed method performs a comprehensive local search around each source, the proximity of sources leads to searching in the same regions. To overcome this problem and to increase diversity, a re-population strategy is applied. At the end of each iteration, the Hamming distances between the sources are calculated. If the distance between two sources is less than d (the predefined maximum Hamming distance value that mentioned above), the one with better fitness is kept, and the other is re-initialized. In this way, premature convergence is prevented and diversity is increased. Also, this algorithm supports our more comprehensive local search strategy without decreasing global search.

3. Results

To show the effectiveness of our method, 14 benchmark datasets with different sizes were selected from the UCI Machine Learning Repository, and a binary particle swarm optimization method (BinPSO), genetic algorithm (GA), a binary ABC algorithm (BitABC), and the proposed algorithm were applied on these benchmarks. We did not need to reapply the other binary ABC variants in the literature since we have previously examined these [41]. To prove the effectiveness of feature selection, the benchmark data were classified by using 5-NN without feature selection. Table 1 summarizes the number of features, the number of classes, and the sizes of the datasets selected for the benchmark testing. During the experiments, all datasets were randomly divided into three parts. The first part includes approximately 60% of each dataset as the training set, 20% of each dataset as the test set and the remaining data are used as the validation set. The validation data is never used in the feature selection process. During the feature selection, a 5-NN is trained with the training set by using the selected subset of features, and then its effectiveness is measured by using the test set. Once the best subset is determined, its performance is measured by using validation set.

The initial population was generated by using a threshold value of 0.85, i.e. any bit of xi set to 0 with a probability of 0.85. Since these algorithms include non-deterministic operations, all of the experiments were repeated 30 times, and the results were averaged to confirm the results statistically. The distance parameter of exBitABC (d) was set to 15. The number of the trials value for the scout bee was set to 50. All of the parameters were chosen by trial and error. For the BinPSO, learning factors c1 and c2 were selected as 2, the initial weight (w) was set to 0.9, and the maximum speed (Vmax) for particles was selected as 0.6, as stated in [43]. For the GA, the crossover and mutation rates were set as 0.8 and 0.2, respectively, as reported in [44]. BinPSO and GA were implemented according to the suggestions of the authors of [43, 44].

Table 1. The datasets consider in this work

Dataset Name	#Features	#Classes	#Samples
Hill-Valley	100	2	606
UrbanLandCover	148	9	168
Musk-I	166	2	476
Arrhythmia	279	16	452
LSVTVoice Rehabilitation	309	2	126
Madelon	500	2	2600
Secom	591	2	1567
Malacious	513	2	373
Isolet5	617	26	1559
Multiple Features	649	6	2000
CNAE	857	9	1080
Micromass	1300	9	931
Gisette	5000	2	6000
Arcene	10000	2	200

In a feature selection problem, the quality of the selection method is determined by the accuracy and the number of the selected features. Since the accuracy is more important than the number of the selected features, the fitness and accuracy values are determined as in equation (10) and (11) respectively, where c=0.9995.

$$fitness = (c * acc) + ((1 - c)\frac{\#selectedFeatures}{\#totalFeatures}) \quad (10)$$

$$acc. = \frac{\#correctlyClassifiedInstances}{\#totalInstances} \quad (11)$$

Population size is one of the important parameters for swarm intelligence algorithms, and generally it is problem dependent. To determine the size of a population, the colony sizes 30, 40, and 50 were compared according to consumed fitness number as shown in Figure 2. There is a general opinion that the error decreases as the size of the population increases. However, our results did not support this idea. For all other datasets, after the algorithm consumed 1000 fitnesses, the system became stable. Although the increase of the population size for some datasets reduced the error, we can see that 40 bees are sufficient for the system for the majority of datasets. A much larger population is needed for hard optimization, problems but it is clear that after reaching a sufficient population size, increasing the population size only increases the execution time. As shown in the convergence graph, generally train set

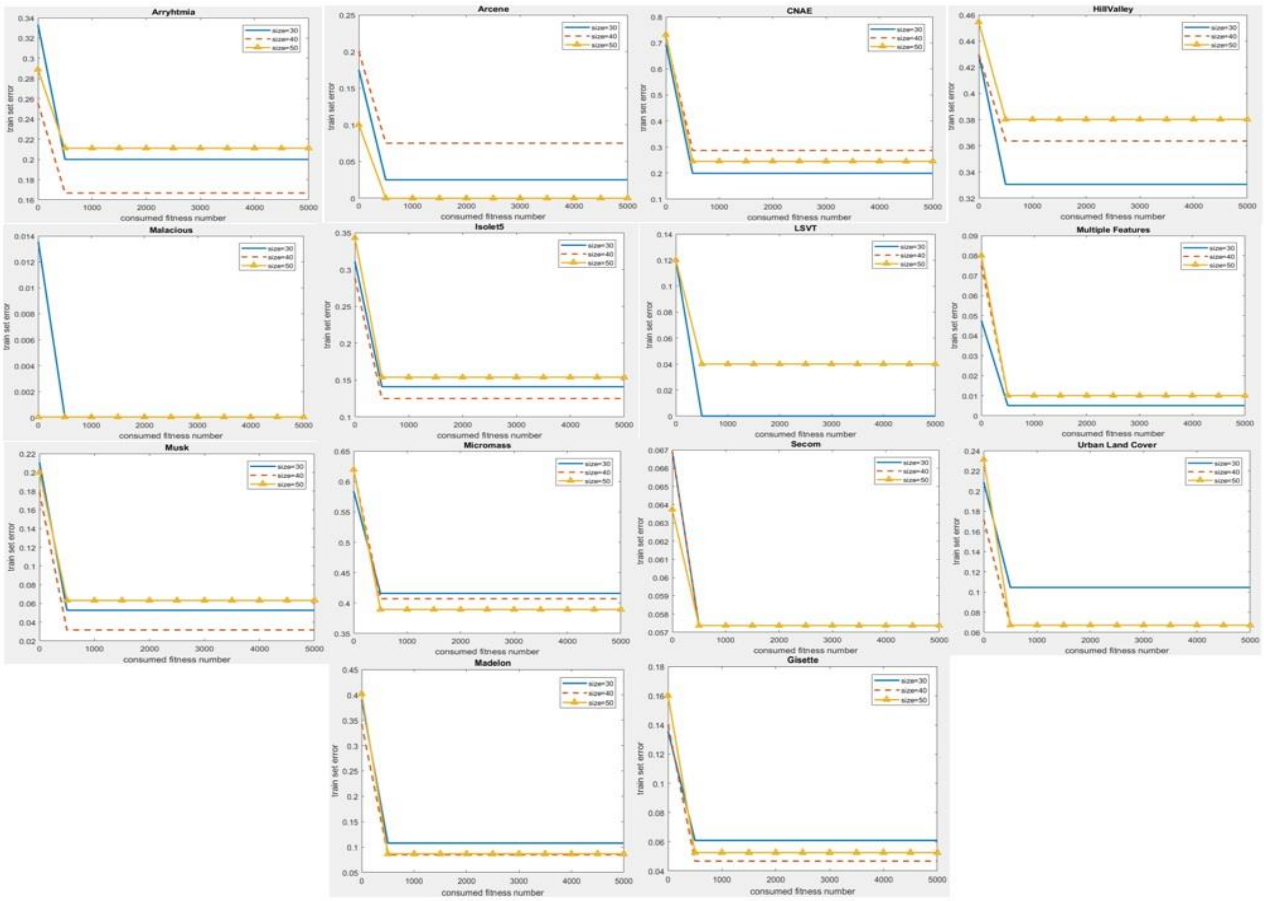


Figure 2. Convergence graph according to the consumed fitness number

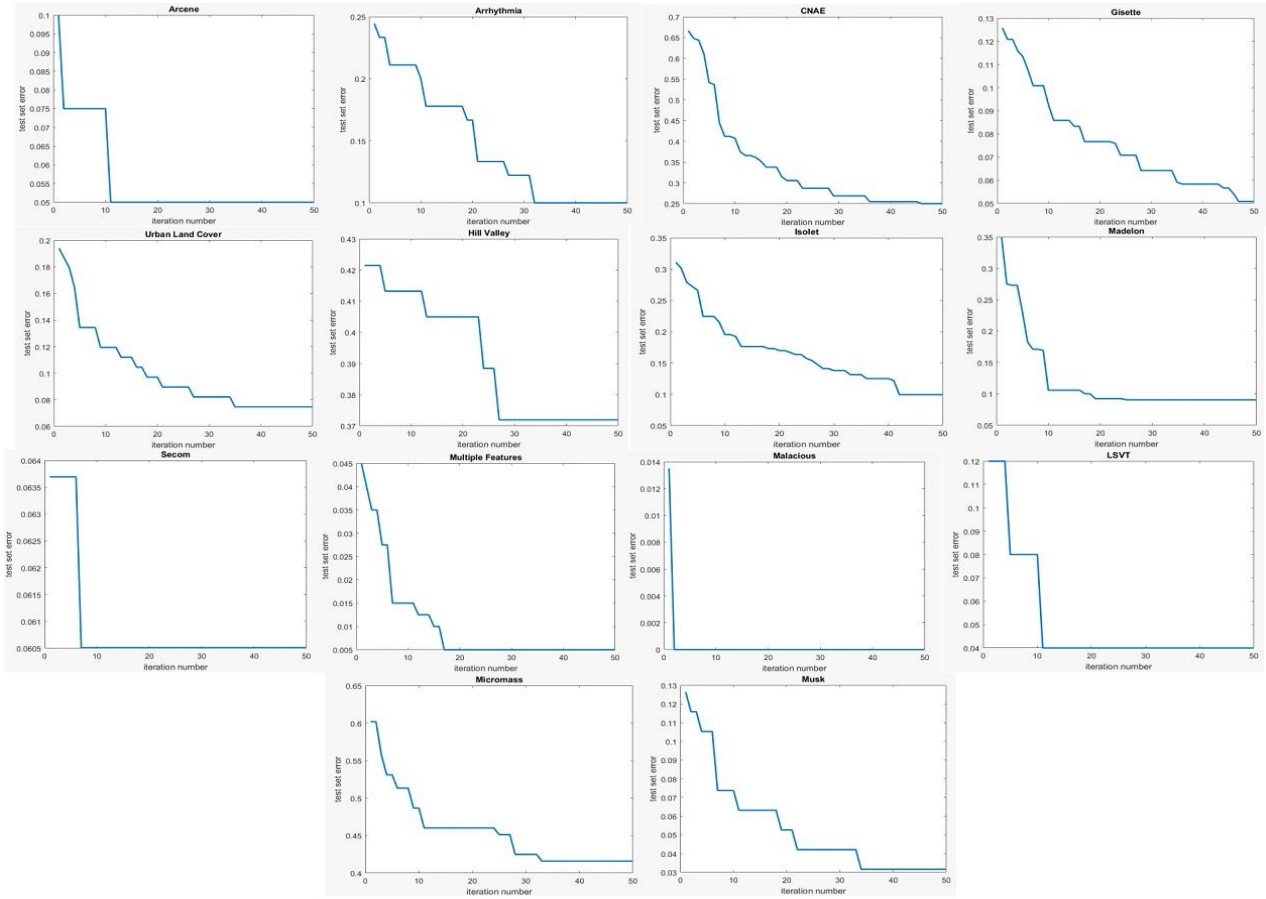


Figure 3. Convergence graph according to iterations

error differences are very small for different population sizes, but we saw that the difference between the execution times is quite high. For LSVT and Secom datasets, the curves for sizes 30 and 40 overlap. Also for Malacious, Multiple Features, and Madelon datasets the curves for sizes 40 and 50 overlap. Therefore, for our datasets population size was set as 40. In ABC algorithms, the sources are being improved in both the employed bee and onlooker bee phases. Namely, the total population size is the sum of the numbers of employed and onlooker bees. However, in GA and PSO, each source can be improved in only one phase in a cycle. Therefore there are 20 employed and onlooker bees for exBitABC and BitABC, 40 individuals for GA and 40 particles for PSO.

To compare different algorithms with each other, the maximum number of iterations is typically used. exBitABC consumes more than one fitness for each resource. According to [45-47], to make a fair comparison in such situations, it is more suitable to use execution time or number of consumed fitness. Background applications, updates, etc. affect the running time of the PC; therefore, the consumed fitness number is selected as the stopping criterion and this is based on the dataset. The consumed fitness number is calculated by counting the number of new sources that produced by the algorithm and setting this as the stopping criterion for BitABC, GA, and Bin PSO.

ABC is an iterative algorithm. At the end of any iteration, if the algorithm converges to an optimum, the algorithm stops to improve, and the fitness value remains constant after this point. Subsequently, continuing the iterations usually does not improve the result but increases the execution time. It is important to find the optimum iteration number to reduce the time complexity. The convergence graph related to iteration number and test set error is shown in Figure 3. The number of max iteration is chosen by trial and-error and is set as 50 for exBitABC. It is clear that for all datasets, 50 iterations are sufficient, and all of them became stable after reaching an optimum point.

At our previous work [41], we implemented, different binary ABC variants to feature selection problem. We saw that, BitABC got better results according to other binary ABC variants. Therefore in this study, we did not apply again other binary ABC algorithms. Since the proposed methods are derived from BitABC, we first compared the performance of the proposed method with respect to BitABC. However, we also compared our results with other well-known meta-heuristic algorithms such as Genetic Algorithm (GA) and Binary Particle Swarm Optimization Algorithm (BinPSO). In order to test the usefulness of the feature selection process, the 5-NN classifier was applied to datasets without any feature reduction.

Table 2 summarizes the classification errors; these are shown in terms of mean and standard deviation in percent. If two methods have the same classification error, the one that selects fewer features is considered as better. In Table 2, the results of exBitABC and the results of other meta-heuristic methods are compared and the best results are shown in bold text. The results of 5-NN is given to show the usefulness of the feature selection process. It is not included in comparison. The 'Sig' row shows statistically significant differences obtained by Wilcoxon Rank Sum Test between the algorithms. "+" indicates that exBitABC is significantly better than the compared algorithm. "-" shows that exBitABC is significantly worse than the compared algorithm. If any other algorithm produces a significantly similar result relative to exBitABC, the compared algorithm is labelled with the "∼" symbol.

As seen in Table 2, the proposed method outperforms the other three methods both in terms of validation error and of the number of selected features. With the exception of Isolet, the improvement of test set errors is meaningful. In addition, it is clear that the size of the feature subsets selected by exBitABC is much smaller than the size of the feature subsets selected by the other methods. The Secom, Malacious, Multiple Features, and Gisette datasets are classified with very small error. Consequently, there is no significant difference between the methods in terms of validation set error. For these datasets, success is determined by the number of selected features. It is clear that exBitABC achieves the same error, but using fewer features than other algorithms. Hill Valley and Micromass are difficult datasets: Even if they are classified with 5-NN by using all features, the error rate is very high. Unlike the other methods, exBitABC decreased the error rate for Micromass using fewer features, and the error difference with all features-5NN is small for Hill Valley. When the results of exBitABC are compared with 5-NN, by using feature selection, it is possible to achieve lower or similar test errors by using fewer features. If exBitABC was excluded, BitABC generally produced better test error results than GA and BinPSO. Among these, the smallest subsets were always selected by using BitABC. According to standard deviation values, GA and BinPSO cannot produce stable results for feature subset size. The significance test is applied to show statistically significant differences in terms of only validation set errors. The significance test performance of exBitABC is statistically worse than BinPSO, GA, and BitABC for the Isolet and Musk datasets. For Secom, Malacious, and Multiple Features datasets, no statistically meaningful difference is observed between methods, but it is clear that for these datasets there is no significant difference between validation error results.

Computation time of the algorithms is shown in Table 2. The algorithms were executed on a server with 32

Table 2. The performance comparison of the algorithms

		BitABC	GA	BinPSO	exBitABC	5-NN
Hill-Valley	Valid Error	0.481±0.044	0.482±0.035	0.495±0.004	**0.462±0.034**	0.44±0.034
	#Feature	19.83±3.18	20.7±4.09	14.37±13.45	**8.9±3.45**	
	Sig	+	+	+		
Urban Land Cover	Valid Error	0.188±0.026	0.202±0.042	0.191±0.031	**0.147±0.03**	0.192±0.025
	#Feature	43.23±29.21	30.07±56.62	48.93±13.8	**10.2±2.22**	
	Sig	+	+	+		
Musk-I	Valid Error	0.191±0.042	0.179±0.047	0.173±0.042	**0.155±0.047**	0.147±0.021
	#Feature	21.37±4.05	38.5±7.29	57.2±12.93	**12.67±2.22**	
	Sig	-	-	-		
Arrhythmia	Valid Error	0.337±0.047	0.33±0.04	0.351±0.045	**0.296±0.04**	0.375±0.035
	#Feature	27.67±6.9	51.97±7.66	66.37±29.87	**13.33±2.78**	
	Sig	+	+	+		
LSVT	Valid Error	0.247±0.09	0.26±0.09	0.24±0.08	**0.201±0.102**	0.33±0.062
	#Feature	30.7±6.92	49.1±8.1	83.73±44.14	**19.73±6.57**	
	Sig	+	+	-		
Madelon	Valid Error	0.2±0.021	0.218±0.027	0.26±0.029	**0.1±0.011**	0.271±0.014
	#Feature	51.07±14.16	86.57±9.05	155.8±37.3	**11.3±2.13**	
	Sig	+	+	+		
Secom	Valid Error	0.07±0.004	0.071±0.004	0.069±0.003	**0.069±0.004**	0.071±0.01
	#Feature	72±11.42	90.93±9.92	170.9±81.74	**41.77±21.57**	
	Sig	~	~	-		
Malacious	Valid Error	0.025±0.017	**0.022±0.015**	0.03±0.023	0.023±0.017	0.007±0.007
	#Feature	70.6±9.77	75.83±7.46	125.3±77.9	**24.63±1.35**	
	Sig	~	~	+		
Isolet5	Valid Error	**0.196±0.023**	0.221±0.025	0.242±0.024	0.211±0.022	0.277±0.018
	#Feature	50.6±5.9	109.23±11.59	192.73±50.95	**44.9±9.87**	
	Sig	-	-	+		
Multiple Features	Valid Error	0.032±0.009	0.035±0.01	0.046±0.01	**0.03±0.01**	0.023±0.004
	#Feature	60.67±7.95	109.17±11.73	236.5±35.88	**44.87±14.89**	
	Sig	-	~	+		
Cnae	Valid Error	0.31±0.041	0.309±0.045	0.285±0.047	**0.212±0.034**	0.129±0.015
	#Feature	67.53±10.39	157.47±8.5	350.93±25.98	**44.83±7.99**	
	Sig	+	+	+		
Micromass	Valid Error	0.617±0.067	0.719±0.047	0.71±0.041	**0.587±0.054**	0.688±0.033
	#Feature	58.03±11.79	207.03±14.18	164.57±169.95	**42.9±10.02**	
	Sig	+	+	+		
Gisette	Valid Error	0.073±0.011	0.102±0.011	**0.069±0.007**	0.071±0.009	0.066±0.004
	#Feature	232.17±44.053	761.03±20.28	2104.2±166.21	**146.53±20.48**	
	Sig	~	+	~		
Arcene	Valid Error	0.247±0.06	0.243±0.072	0.262±0.063	**0.182±0.076**	0.229±0.053
	#Feature	924.5±139.33	1504.4±36.13	3233.4±904.37	**776.4±37.68**	
	Sig	+	+	+		

GB RAM and an Intel Xeon E5 2.6 GHz CPU by using MATLAB 2016b. In the case of computational cost, the size of the dataset is evaluated according to the number of cells (\#features x \#samples). Generally, the execution time of exBitABC is shorter than that of other methods. Regarding Table 3, for small datasets no substantial difference is observed between methods. However, when the larger datasets like Gisette are considered, exBitABC is faster than the others.

4. Discussion and Conclusion

The aim of this study is to develop a binary ABC algorithm. Within this scope, existing binary ABC algorithms are examined for strengths and weaknesses. When the performances of these variants are compared, it is clear that, using bitwise operators are successful in binarization process [41]. Therefore, at the proposed method, binarization process was carried out using bitwise operators.

It is also investigated that is it possible to make a limited local search around good sources that have already been discovered without falling local optima. After some research, adding a local search procedure was performed using Hamming distance measurement. So, promising results are obtained when the proposed method applied to feature selection problem. Because of local search strategy, it was consumed multiple fitness at each iteration. For all that, the computational costs did not exceed other algorithms.

Table 3. Time consumptions of algorithms in seconds

Dataset	BitABC	BinPSO	GA	exBitABC
Hill-Val.	45.065 ±0.71	**43.57 ±0.17**	54.1 ±0.71	48.92 ±0.55
UrbL.C	**46.64 ±0.76**	48.49 ±0.36	57.17 ±0.75	49.91 ±0.43
Musk	**42.47 ±0.31**	45.047 ±0.3	54.21 ±0.85	47.32 ±0.24
Arryht.	**43 ±0.62**	48.002 ±0.28	54.68 ±0.32	46.67 ±0.36
Lsvt	40.2 ±0.22	**40.12 ±0.22**	54.54 ±4.67	45.23 ±0.17
Madel.	101.3 ±9.16	198.47 ±7.44	140.3 ±6.45	**76.61 ±1.85**
Secom	82.21 ±1.73	118.38 ±5.46	97.08 ±9.59	**80.31 ±1.42**
Malac.	**42.64 ±0.52**	51.21 ±0.58	54.5 ±0.4	48.22 ±0.38
Isolet	106.4 ±25.64	174.96 ±3.42	113.6 ±4.38	**95.48 ±6.27**
Mul.F.	147.98 ±6.68	223.3 ±70.36	146.1 ±9.26	**118.1 ±6.28**
Cnae	66.85 ±3.38	117.13 ±4.94	84.13 ±3.49	**59.54 ±0.67**
Micro.	58.86 ±1.97	86.21 ±2.45	65.66 ±0.81	**58.38 ±1.35**
Gisette	2741.3 ±101.7	7931 ±261.7	4431.4 ±88	**1227 ±12.3**
Arcene	100.79 ±3.72	154.83 ±2.65	144.6 ±1.36	**99.06 ±1.46**

In this article, a new binary ABC algorithm named exBitABC that is suitable for feature selection problems is proposed. The new method has strong local search capability and can easily escape local optima. To show the effectiveness of the proposed method, 14 benchmark datasets from the UCI Machine Learning Repository were processed by using BinPSO, GA, BitABC and exBitABC algorithms, and the results were compared. For all datasets, exBitABC produced better results than BitABC in terms of validation set error, and completed processing in a shorter time. exBitABC also selected fewer features in every trial. When compared with BinPSO and GA, exBitABC yielded better performance for 13 datasets, while BitABC was better for the remaining one dataset. exBitABC also reduced the features better than both BinPSO and GA. These results demonstrate that the increased local search ability of the exBitABC algorithm improves the performance of the algorithm in terms of validation set error, feature size, and running time.

Results are only compared with BitABC within existing binary ABC methods. It is because, some binary variants of ABC algorithm are applied and compared in our previous study [41]. BitABC outperformed other methods, therefore this study focused on getting a good result from BitABC. Additionally, results are compared with GA and BinPSO because of their optimization performances. In our next study, we will compare our results some other population-based algorithms like Ant Colony Optimization, Bacterial Colony Optimization.

The main purpose of this study is, adding a limited local search process to ABC algorithm. At the new source generation formula, 'AND' and 'OR' bitwise operators and two random vectors are used. With 'OR' operator diversity of the selected features are increased. With random vectors generated from current source, it is ensured that the new source is in a certain distance to the current source. Namely, new sources are produced in the local area of the current source. Finally 'AND' operator used for obtaining a sparse vector.

There are some ABC variants based on bitwise operators in the literature. All of these are proposed for some binary problems. Bitwise operators are used for producing neighbor source. At [21] bitwise 'XOR', 'AND' and 'OR' operators are used. At [22] only 'XOR' operator is used. The 'XOR' operator prevents the new source from staying in the local area. Therefore it did not used. The main difference from these studies is the new source generation strategy ensures that the new source remains in the local area.

References

[1] Guyon, I., Elisseeff, A. 2013. An Introduction to Variable and Feature Selection. Journal of Machine Learning Research, 3, 1157–1182.

[2] Sánchez-Maroño, N., Alonso-Betanzos, A., Tombilla-Sanromán, M. 2007. Filter Methods for Feature Selection: a Comparative Study. I Proceedings of the 8th International Conference on Intelligent Data Engineering and Automated Learning, December, Berlin, Heidelberg, 178–187.

[3] Kohavi, R., John, G. H. 1997. Wrappers for Feature Subset Selection. Artificial Intelligence, 1-2, 273-324.

[4] Unler, A., Murat, A. 2010. A Discrete Particle Swarm Optimization Method for Feature Selection in Binary Classification Problems. European Journal of Operational Research, 206(3), 528-539.

[5] Cervante, L., Xue, B., Shang, L., Zhang, M. 2012. A Dimension Reduction Approach to Classification Based on Particle Swarm Optimisation and Rough Set Theory. Advances in Artificial Intelligence, 1 st ed., Springer, Berlin, Heidelberg, 313–325.

[6] Cervante, L., Xue, B., Shang, L., Zhang, M. 2013. A Multi-Objective Feature Selection Approach Based on Binary Pso and Rough Set Theory. Evolutionary Computation in Combinatorial Optimization, 7832, 25–36.

[7] Yang, J., Honavar, V.G. 1998. Feature Subset Selection Using a Genetic Algorithm. IEEE Intelligent System, 13(2), 44–49.

[8] Raymer, M. L., Punch, W. F., Goodman, E. D., Kuhn, L. A., Jain, A. K. 2000. Dimensionality Reduction Using Genetic Algorithms. IEEE Transactions on Evolutionary Computation, 4(2), 164–171.

[9] Ahmed, S., Zhang, M., Peng, L. 2014. Improving Feature Ranking for Biomarker Discovery in Proteomics Mass Spectrometry Data Using Genetic Programming. Connection Science, 26(3), 215–243.

[10] Nemati, S., Basiri, M. E., Ghasem-Aghaee, N., Aghdam, M. H. 2009. A Novel ACO-GA Hybrid Algorithm for Feature Selection in Protein Function Prediction. Expert System Application, 36(10), 12086–12094.

[11] Wen, L., Yin, Q., Guo, P. 2008. Ant Colony Optimization Algorithm for Feature Selection and Classification of Multispectral Remote Sensing Image. IEEE International Geoscience and Remote Sensing Symposium (IGARSS2008), 7-11 July, Boston, USA, 923-926.

[12] Jensen, R. 2006. Performing Feature Selection with ACO. Swarm Intelligence in Data Mining, Springer, Berlin, Heidelberg, 45-73.

[13] Nakamura, R., Pereira, L., Costa, K., Rodrigues, D., Papa, J. 2012. BBA: a Binary Bat Algorithm for Feature Selection. Conference on Graphics, Patterns and Images, 22–25 Aug, Ouro Preto, 291-297.

[14] Oduntana, I. O., Toulouse, M., Baumgartner, R., Bowman, C., Somorjai, R., Crainic, T. G. 2008. A Multilevel Tabu Search Algorithm for the Feature Selection Problem in Biomedical Data. Computers & Mathematics with Applications, 55, 1019–1033.

[15] Chuang, L. Y., Yang, C. H. 2009. Tabu Search and Binary Particle Swarm Optimization for Feature Selection using Microarray Data. Journal of Computational Biology, 16(12), 1689-1703.

[16] Balabina, R. M., Smirnov, S. V. 2011. Variable Selection in Near-Infrared Spectroscopy: Benchmarking of Feature selection Methods on Biodiesel Data. Analytica Chimica Acta, 692, 63–72.

[17] Ustunkar, G., Ozogur-Akyuz, S., Weber, G. W., Friedrich, C. M., Aydin Son, Y. 2011. Selection of Representative SNP Sets for Genome-Wide Association Studies: a Metaheuristic Approach. Optimization Letters, 6(6), 1207–1218.

[18] Hancer, E. 2019. Differential Evolution for Feature Selection: a Fuzzy Wrapper-Filter Approach. Soft Computing, 23(13), 5233-5248.

[19] Li, T., Dong, H., Sun, J. 2019. Binary Differential Evolution Based on Individual entropy for Feature Subset Optimization. IEEE Access, 7, 24109-24121.

[20] Öztürk, C., Hançer, E., Karaboğa, D. 2015. A Novel Binary Artificial Bee Colony Algorithm Based on Genetic Operators. Information Science, 297, 154-170.

[21] Jia, D., Duan, X., Khan, M. K. 2014. Binary Artificial Bee Colony Optimization Using Bitwise Operation (BitABC). Computers and Industrial Engineering, 76, 360–365.

[22] Kiran, M. S., Gündüz, M. 2013. XOR Based Artificial Bee Colony algorithm for Binary Optimization. Turkish Journal of Electrical Engineering & Computation Sciences, 21, 2307–2328.

[23] Kashan, M. H., Nahavandi, N., Kashan, A. H. 2012. DisABC: A New Artificial Bee Colony Algorithm for Binary Optimization. Applied Soft Computing, 12, 342-352.

[24] Öztürk, C., Hançer, E., Karaboğa, D. 2014. Dynamic Clustering With Improved Binary Artificial Bee Colony-IDisABC. Applied Soft Computing, 28, 69-80.

[25] Hançer, E., Xue, B., Karaboğa, D., Zhang, M. 2015. A Binary ABC Algorithm Based on Advanced Similarity scheme for Feature Selection. Applied Soft Computing, 36, 334-348.

[26] Singhal, P. K., Noresh, R., Sherma, V. 2015. A Novel Strategy-Based Hybrid Binary Artificial Bee Colony Algorithm for Unit Commitment Problem. Arabian Journal for Science and Engineering, 40(5), 1455–1469.

[27] Yurtkurtaran, A., Emel, E. 2016. A Discrete Artificial Bee Algorithm for Single Machine Scheduling Problem. International Journal of Production Research, 54(22), 6860-6878.

[28] Zhang, X., Zhang, X. 2016. A Binary Artificial Bee Colony Algorithm for Constructing Spanning Trees in Vehicular ad Hoc Networks. Ad Hoc Networks, 58, 198-204.

[29] Zhang, S., Gu, X. 2015. An Effective Discrete Artificial Bee Colony Algorithm for Flow Shop Scheduling Problem with Intermediate Buffers. Journal of Central South University, 22, 3471–3484.

[30] Tasgetiren, M. F., Pan, Q., Suganthan, P. N., Chen, A. 2011. A Discrete Artificial Bee Colony Algorithm for the Total Flow Time Minimization in Permutation Flow Shops. Information Sciences, 181, 3459–3475.

[31] Zhang, H., Ye, D. 2015. Key-Node-Based Local Search Discrete Artificial Bee Colony Algorithm for Obstacle-Avoiding Rectilinear Steiner Tree

Construction. Neural Comput & Applications, 26, 875–898.

[32] Ye, D., Chen, Z. 2015. A New Approach to Minimum Attribute Reduction Based on Discrete Artificial Bee Colony. Soft Computing, 19, 1893–1903.

[33] Ribas, I., Companys, R., Tort-Martorell, X. 2015. An Efficient Discrete Artificial Bee Colony Algorithm for the Blocking Flow Shop Problem with Total Flowtime Minimization. Expert Systems with Applications, 42, 6155–6167.

[34] Han, Y. Y., Gong, D., Sun, X. A Discrete Artificial Bee Colony Algorithm Incorporating Differential Evolution for the Flow-Shop Scheduling Problem with blocking. Engineering Optimization, 47, 927–946.

[35] Schiezaro, M., Pedrini, H. 2013. Data Feature Selection Based on Artificial Bee Colony Algorithm. Journal on Image and Video Processing, 47.

[36] Özmen, Ö., Batbat, T., Özen, T., Sinanoğlu, C., Güven, A. 2018. Optimum Assembly Sequence Planning System Using Discrete Artificial Bee Colony Algorithm. Mathematical Problems in Engineering, 2018, 340764.

[37] Wei, L., Hanning, C. 2012. BABC: A Binary Version of Artificial Bee Colony Algorithm for Discrete Optimization. International Journal of Advancements in Computing Technology, 4(14), 307-314.

[38] Tran, D. C., Wu, Z. 2014. New Approaches for Binary Artificial Bee Colony Algorithm for Solving 0-1 Knapsack Problem. Advances in Information Sciences and Service Sciences, 4(22), 464-471.

[39] Kıran, M. S. 2015. The Continues Artificial Bee Colony Algorithm for Binary Optimization. Applied Soft Computing, 33, 15-23.

[40] Mandala, M., Gupta, C. P. 2014. Binary Artificial Bee Colony Optimization for GENCO's Profit Maximization under Pool Electricity Market. International Journal of Computer Applications, 90, 34-42.

[41] Ozger, Z. B., Bolat, B., Diri, B. 2016. A Comparative Study on Binary Artificial Bee Colony Optimization Methods for Feature Selection. INnovations in Intelligent SysTems and Applications (INISTA), 2-5 Aug., Romaina, 1-4.

[42] Karaboga, D., Akay, B. 2009. A Survey: Algorithms Simulating Bee Swarm Intelligence. Artificial Intelligence Review, 31, 61-85.

[43] Mirjalili, S., Lewis, A. 2013. S-Shaped Versus v-Shaped Transfer Functions for Binary Particle Swarm Optimization. Swarm Evolution Computation, 9, 1–14.

[44] Sivanandam, S., Deepa, S. 2008. Genetic Algorithm Implementation Using Matlab. Introduction to Genetic Algorithms, Berlin: Heidelberg, 211-262.

[45] Mernik, M., Liu, S. H., Karaboga, D., Črepinšek, M. 2015. On Clarifying Misconceptions When Comparing Variants of the Artificial Bee Colony Algorithm by Offering a New Implementation. Information Sciences, 291, 115-127.

[46] Draa, A. 2015. On the Performances of the Flower Pollination Algorithm–Qualitative and Quantitative Analyses. Applied Soft Computing, 34, 349-371.

[47] Črepinšek, M., Liu, S. H., Mernik, L., Mernik, M. 2016. Is a Comparison of Results Meaningful from the Inexact Replications of Computational experiments?. Soft Computing, 20(1), 223-235.