Clustering Analysis with Combination of Artificial Bee Colony Algorithm and *k*-Means Technique

Giuliano Armano and Mohammad Reza Farmani

Abstract—**Clustering is a popular data analysis and data mining technique. Among different proposed methods, *k*-means is an efficient clustering technique to cluster datasets, but this method highly depends on the initial state and usually converges to local optimum solution. This paper takes the advantage of a novel evolutionary algorithm, called artificial bee colony (ABC), to improve the capability of *k*-means in finding global optimum clusters in nonlinear partitional clustering problems. The proposed method is the combination of *k*-means and ABC algorithms, called *k*ABC, which can find better cluster portions. Both *k*ABC and *k*-means are run on three known data sets from the UCI Machine Learning Repository. The simulation results show that the combination of ABC and *k*-means technique has more ability to search for global optimum solutions and more ability for passing local optimum.**

Index Terms—**Artificial bee colony algorithm, *k*-means.**

I. Introduction

The process of grouping data into classes or clusters, such that the data in each cluster share a high degree of similarity while being very dissimilar to data from other clusters, is called data clustering. Attribute values which describe the objects are used for assessing the dissimilarities among clusters. And among these attributes, distance measures are the most common ones. Different areas such as data mining, machine learning, biology, and statistics include the roots of data clustering. Generally speaking, hierarchical and partitional clustering are the two main categories of traditional clustering methods [1]–[3]. In this paper, a partitional clustering method is considered. One of the most popular partitional clustering methods, which is developed about three decades ago, is the *k*-means algorithm. This algorithm is defined over continuous data and used in variety of domains. However, as *k*-means needs initial partitions to start its process, better results are given only when the initial partitions are close to the final solution. In other words, the results of this technique highly depend on the initial sate and converge to local optimal solution.

A lot of studies have been done in clustering to overcome this problem [1]–[20]. Kao *et al.* have introduced a hybrid method based on combining *k*-means, Nelder-Mead simplex, and Particle Swarm Optimization (PSO) for cluster analysis [1]. A hybrid algorithm according to the combination of Genetic Algorithm (GA), *k*-means, and logarithmic regression expectation maximization has been presented by Cao and Krzysztof [2]. Zalik has proposed the performance of correct clustering without pre-assigning the exact number of clusters within *k*-means [3]. Krishna and Murty have shown an approach called genetic *k*-means algorithm for clustering analysis [4]. A GA based method, which contains a basic mutation operator specific to clustering called distance-based mutation, has been introduced by Mualik [5]. This method is used to solve the clustering problem on real life datasets to evaluate the performance. An algorithm named HBMO has been proposed by Fathian and Amiri to solve clustering problems [6]. A GA that exchanges neighboring centers for *k*-means clustering has demonstrated by Laszlo and Mukherjee [7]. Shelokar *et al*. have introduced an evolutionary algorithm based on Ant Colony Optimization (ACO) for clustering problems [8]. A combination of two evolutionary algorithms, ACO and Simulated Annealing (SA), has been proposed by Niknam *et al*. to solve clustering problems in [9], [10]. They also have presented a hybrid evolutionary algorithm based on PSO and SA to find optimal cluster centers [11]. Zhang *et al*. [12] propose a new method called K-Harmonic Means (KHM). This method was then modified by Hammerly and Elkan [13]. The purpose of this method is to minimize the harmonic mean of all points in the data set around the central cluster. Although the KHM can reduce the initial problem, but the KHM still have the possibility of optimal local problem [14]. Shelokar introduced an evolutionary algorithm based on ACO algorithm for clustering problem [15]. Merwe *et al.* used the PSO algorithm to solve the clustering problem [16], [17]. Karaboga *et al.* used the Artificial Bee Colony (ABC) algorithm to solve the problem [18], [19]. Zou *et al.* proposed a Cooperative Article Bee Colony (CABC) algorithm to solve the clustering problem [20], in which the cooperative search technique was introduced.

The ABC algorithm is one of the modern swarm intelligence methods. This algorithm was first proposed by Karaboga [21]. The ABC was developed through simulation of intelligent foraging behavior of honey bees, and has been found to be robust in solving continuous nonlinear optimization problems. Since the ABC algorithm is simple in concept, easy to implement, and has fewer control parameters, it has attracted the attention of researchers and been widely used in solving many numerical [22], [23] and engineering optimization problems [24]–[26]. As mentioned earlier, the main drawback of *k*-means is that the result is sensitive to the selection of the initial cancroids and may converge to the local optimum [27]. Therefore, the initial selection of *k*-means cancroids affects the main processing of *k*-means and the partition result of the dataset as well. In the

Manuscript received September 25, 2013; revised November 10, 2013.

The authors are with IASC (Intelligent Agents and Soft-Computing) group, Department of Electrical and Electronic Engineering, University of Cagliari, Piazza d'Armi, I-09123, Cagliari, Italy (e-mail: armano@daiee.unica.it, mohammad.farmani@daiee.unica.it).

DOI: 10.7763/IJCTE.2014.V6.852

current study, the ABC algorithm is utilized to find the optimal initial cluster cancroids for *k*-means. Contrary to the localized searching of the *k*-means algorithm, the ABC performs a globalized search in the entire solution space.

The remainder of this paper is organized as follows: Section II provides a general overview of *k*-means. In Section III, the ABC algorithm is introduced. The combination of ABC and *k*-means for clustering problems is described in Section IV. Section V provides the experimental results for comparing the performance of the proposed method with the simple *k*-means algorithm. The discussion of the experiments' results is also presented in this section. The conclusion is in Section VI.

II. *K*-Means Clustering Algorithm

k-means is a simple algorithm based on the firm foundation of analysis of variances. In this method, a set of data is clustered into a predefined number of clusters. *k*-means starts with randomly initial cluster centroids and keeps reassigning the data objects in the dataset to cluster centroids based on the similarity between the data objects and the cluster centroids. The reassignment procedure will stops when a convergence criterion (e.g. the number of iteration, or no change in the cluster results after a certain number of iteration) is met. The *k*-means clustering process is described by the four following steps:

1) Randomly create *K* centroids to set an initial dataset partition.
2) Assign each sample in the given dataset to the closest cluster centroid.
3) Recalculate the centroid C_j using (1).

$$C_j = \frac{1}{n_j} \sum_{\forall d_j \in S_j} d_j \tag{1}$$

where d_j denotes the document vectors that belong to cluster S_j; C_j stand for the centroid vector; n_j is the number of document vectors that belong to cluster S_j.

4) Repeat step 2 and 3 until the convergence is achieved.

As *k*-means' performance significantly depends on the selection of the initial centroids, the algorithm may finally converge to the local optimum. Therefore, the processing of *k*-means is to search the local optimum solution in the vicinity of the initial solution and to refine partition result. The same initial cluster centroids in a dataset will always generate the same cluster results. However, if good initial centroids can be obtained using any other techniques, *k*-means would work well in refining the clustering centroids to find the optimal clustering centers.

III. Artificial Bee Colony Algorithm

Karaboga recently proposed a swarm intelligence algorithm inspired by the foraging behaviors of bee colonies [21]. This algorithm was further developed by Karaboga *et al.* [23], [24], [28], [29]. The Artificial Bee Colony (ABC) algorithm treats the search space as it were a foraging environment, so that each point in the search space corresponds to a food source (solution) that the artificial bees could exploit. The fitness of the solution is represented as the nectar amount of a food source. According to this algorithm, three kinds of bees exist in a bee colony: employed bees, onlooker bees, and scout bees. Employed bees exploit the specific food sources they have explored before and give the quality information of the food sources to the onlooker bees. Information about the food sources is received by onlooker bees, and then, a food source to exploit depending on the information of nectar quality will be chosen by them. The more nectar the food source contains, the larger probability the onlooker bees will choose it. A parameter, called “limit”, controls the employed bees whose food should be abandoned. These food sources will become scout bees, whose responsibility is to randomly search the whole environment. In the ABC algorithm, half of the colony is made of employed bees and the other half includes the onlooker bees. Each food source is exploited by only one employed bee. That is, the number of the employed bees or the onlooker bees is equal to the number of food sources. The details of the ABC algorithm are given below:

1) Initialization phase: Each food source $x_{i,j}$ in the population is initialized by scout bees and control parameters are set. Whit $i = 1, 2, ..., SN$, $j = 1, 2, ..., D$. *SN* is the number of food sources and equals to half of the colony size. *D* is the dimension of the problem, representing the number of parameters to be optimized. The most common way to initialize food sources is the following:

$$x_{i,j} = l_j + rand(0,1)(u_j - l_j) \tag{2}$$

where l_j and u_j are lower and upper bounds of the *j*th parameter. In this phase, the fitness of food sources (objective function values) will be evaluated and additional counters which store the numbers of trails of each bee are set to 0.

2) Employed bees phase: Employed bees search for new food sources having more nectar (better fitness value) within the neighborhood of the food sources $x_{i,j}$ in their memory. After finding a neighbor food source, they evaluate its fitness. Equation (3) is used to determine a neighbor food source $v_{i,j}$:

$$v_{i,j} = x_{i,j} + \varphi\left(x_{i,j} - x_{k.j}\right) \tag{3}$$

where *k* is a randomly selected food source different from i, and *j* is a randomly selected dimension. φ is a random number which uniformly distributed in range [-1,1]. As it can be seen, a new food source *v* is determined by changing one dimension on *x*. If the new value in this dimension produced by this operation exceed its predetermined boundaries, it will set to be the boundaries.

The new food source is then evaluated and a greedy selection is applied to the original food source and to the new one. The best will be kept in memory. The trials counter of this food will be reset to zero if the food source is improved; otherwise, its value will be incremented by one.

3) Onlooker bees phase: Onlooker bees waiting in the hive receive the food source information from employed bees and then probabilistically choose their food sources depending on this information. By using the fitness values provided by employed bees, the probability values of food sources will be calculated. An onlooker bee chooses a food source depending on its probability value. That is to say, there may be more than one onlooker bee choosing a same food source if that food source has a higher fitness. The probability is calculated according to (4) as follows:

$$p_i = \frac{fitness_i}{\sum_{j=1}^{SN} fitness_j} \tag{4}$$

After food sources have been probabilistically chosen for onlooker bees, each onlooker bee finds a new food source in its neighborhood using (3). Fitness values of these new food sources will be computed and, as in the employed bees phase, a greedy selection is applied between v_i and x_i. In other words, more onlooker bees will be recruited to richer food sources.

4) Scout bees phase: In this phase, if the value of trials counter of a food source is greater than the “limit” parameter, the food source will be abandoned and the bee becomes a scout bee. According to (2), as in the initialization phase, a new food source will be produced randomly in the search space for each scout bee and the trials counter of the bee will be reset to zero.

The three employed, onlooker, and scout bees' phases will be repeated until the termination criterion is met and the best food source (i.e. the one that shows the best optimal value) will be selected as final solution.

IV. Combination of ABC and *K*-Means

As mentioned in the previous sections, *k*-means is computationally light and converges after a limited iterations. However, the studies conducted by the researchers confirm that the algorithm is highly dependent on the initialization of centroids and usually gets stuck in local optimums. It was also mentioned that the ABC algorithm performs a global search in the entire solution space. If given enough time, ABC can generate good and global results. Therefore, here, we propose a new combined algorithm to use the merits of the two *k*-means and ABC algorithms for solving clustering problems. The proposed algorithm does not depend on the initial centroids and can avoid being trapped in a local optimum solution as well.

In the proposed algorithm, each food source in the search environment represents a set of centroids; that is, a food source represents a possible solution for clustering, and the position x_i is constructed as:

$$x_i = (C_{i1}, C_{i2}, \ldots, C_{iK}) \tag{5}$$

where K is the number of clusters, C_{ij} is the jth centroid of the ith food source. To measure the overall clustering quality of each food source, a clustering criterion function should be defined. In this work, a simple criterion inspired by sum of square error called *distortion* is used as the clustering criterion function. *Distortion* is the total sum of the squared distance between all samples and their cluster centers and defined as follows:

$$E = \sum_{j=1}^{K} \sum_{z_i \in C_j} \left\| z_i - C_j \right\|^2 \tag{6}$$

where z_i represents ith pattern belongs into cluster C_j. Here, the goal is to obtain a partitioning of the data set, such that E is minimized. The *distortion* criterion is valid for cluster sample dense as well as the small differences in the number of various clustering samples. The procedure for the proposed algorithm can be summarized as follows:

Setp 1: Randomly Initialize the positions of food sources (each food source being a set of centroids) and use the *k*-means algorithm to finish clustering task for all produced positions and compute the fitness value of each group of centroids using (6).

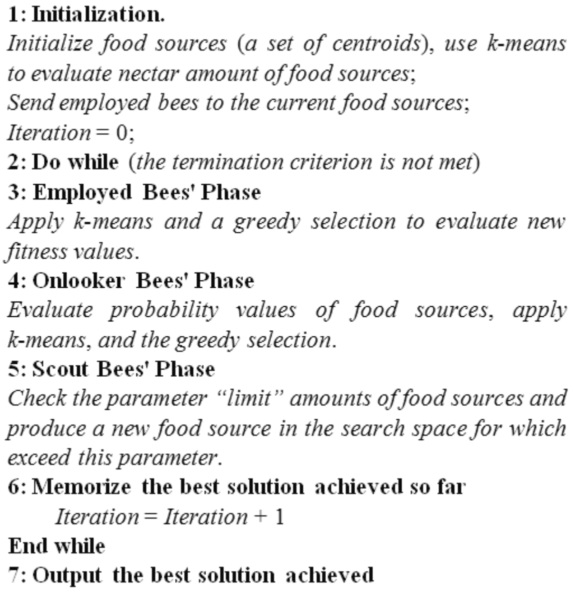
Setp 2: Search for new food sources and update the place of food sources by employed bees. Apply the *k*-means algorithm and a greedy selection to evaluate new fitness values and compare them with the original ones. Better food sources will be delivered to onlooker bees.

Step 3: Calculate probability values of food sources and update their place according to the probability values by onlooker bees. Again, the *k*-means algorithm and a greedy selection will be applied to finish clustering, evaluate new fitness values using (6) and compare them with the original ones to update them.

Step 4: Check the trial counter of food sources and produce a new food source (set of centroids) in the search space for which exceed the “limit” parameter amount.

Step 5: Repeat from step 2 until the termination criterion is met.

The pseudo of the proposed algorithm, named ABC*k*, is illustrated in Fig. 1.



```
1: Initialization.
Initialize food sources (a set of centroids), use k-means
to evaluate nectar amount of food sources;
Send employed bees to the current food sources;
Iteration = 0;
2: Do while (the termination criterion is not met)
3: Employed Bees' Phase
Apply k-means and a greedy selection to evaluate new
fitness values.
4: Onlooker Bees' Phase
Evaluate probability values of food sources, apply
k-means, and the greedy selection.
5: Scout Bees' Phase
Check the parameter "limit" amounts of food sources and
produce a new food source in the search space for which
exceed this parameter.
6: Memorize the best solution achieved so far
    Iteration = Iteration + 1
End while
7: Output the best solution achieved
```

Fig. 1. Pseudo of the ABC*k* algorithm.

V. Exprimental Results and Discussion

The experimental results comparing the ABC*k* with the *k*-means algorithm are provided for three real-life datasets (Iris, Wine, and Contraceptive Method Choice (CMC)). They are briefly summarized below:

Iris data (n = 150, d = 4, K = 3). These data with 150 random samples of flowers from the iris species setosa, versicolor, and virginica used by Fisher [30]. From each species there are 50 observations for sepal length, sepal width, petal length, and petal width in cm.

Wine data (n = 178, d = 13, K = 3). These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars [31]. The analysis determined the quantities of 13 constituents found in each of the three types of wines. There are 178 instances with 13 numeric attributes in wine data set. All attributes are continuous and there is no missing attributes.

Contraceptive Method Choice (CMC) data (n = 1473, d = 10, K = 3). These data are a subset of the 1987 National Indonesia Contraceptive Prevalence Survey [32]. The samples are married women who were either not pregnant or do not know if they were at the time of interview. The problem is to predict the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socioeconomic characteristics.

In this paper, in the ABC*k* algorithm, the colony size, “limit” parameter, and number of iteration, are set to 10, 100, and 20, respectively. The comparison of results for each dataset based on the best solution found in 100 distinct runs of each algorithm and the convergence processing time taken into attain the best solution. The algorithms are implemented on a Intel Core i7, 2.4 GHz, 8 GB RAM computer. The quality of the respective clustering will also be compared, where the quality is measured by the following two criteria:

1) the distortion criterion as defined in (6). Clearly, the smaller the sum is, the higher the quality of clustering is.
2) The *F*-measure which uses the ideas of precision and recall from information retrieval [33]. Each class i (as given by the class labels of the used benchmark dataset) is regarded as the set of n_i items desired for a query; each cluster j (generated by the algorithm) is regarded as the set of n_j items retrieved for a query; n_{ij} gives the number of elements of class i within cluster j. For each class i and cluster j precision and recall are then defined as $p(i,j) = (n_{ij} / n_j)$ and $r(i,j) = (n_{ij} / n_i)$ and the corresponding value under the F-measure is $F(i,j) = ((b^2+1).p(i,j).r(i,j)) / (b^2.p(i,j)+r(i,j))$, where b = 1 is chosen here to obtain equal weighting for $p(i,j)$ and $r(i,j)$. The overall F-measure for the dataset of size n is given by:

$$F^{*}=\sum_{i}\frac{n_i}{n}MAX_{j}\{F(i,j)\} \qquad (7)$$

Obviously, the bigger *F*-measure is, the higher the quality of clustering is.

The simulation results given in Table I to Table III show that ABC*k* is much more precise than the *k*-means algorithm. In other words, it provides the optimum value and small standard deviation in compare to those of obtained by *k*-means. For instance, the results obtained on the Iris dataset show that ABC*k* converges to the global optimum of 97.33 in all times while the average and standard deviation amounts of *k*-means are 102.73 and 10.52. The standard deviation of the fitness function for the proposed algorithm is 0. Table II shows the results of algorithms on the Wine dataset. The average optimum values, which are obtained by ABC*k* and *k*-means in all runs, are 16574.49 and 16890.16, respectively. As it is presented, the ABC*k* noticeably resulted in a smaller standard deviation value in comparison with *k*-means.

Table III provides the results of algorithms on the CMC dataset. As seen, the ABC*k* is far superior in term of the standard deviation value, the worst global solution, and the best global solution. On CMC dataset, ABC*k* obtains the average optimum and standard deviation values of 5711.27 and 3.41, respectively. While *k*-means results in 5864.22 and 51.32.

In terms of computational costs, *k*-means needs much lower evaluation times, but the results are less than satisfactory. Indeed, the illustrated results show that the number of function evaluations of *k*-means are significantly less than those of ABC*k*.

The simulation results of the tables also illustrate that the average of F-measure of the proposed algorithm is better than or equal to those obtained by the *k*-means algorithm on the all datasets. F-means is an indication that shows how the clusters are spatially well separated and the accuracy presents the ability of both algorithms to cluster the data into different partitions, correctly. To conclude, the simulation results demonstrate that ABC*k* converges to global optimum solutions with smaller standard deviation and more function evaluations which is caused by the statistical behavior of all nature inspired optimization algorithms.

TABLE I: Resaults Obtained on Iris Data for 100 Runs

	Best	Worst	Average	Std. Dev.	CPU time	F-measure
k-means	97.33	123.97	102.73	10.52	~ 0.1 sec	87.33%
ABC*k*	97.33	97.33	97.33	0	~ 23.7 sec	89.25%

TABLE II: Resaults Obtained on Wine Data for 100 Runs

	Best	Worst	Average	Std. Dev.	CPU time	F-measure
k-means	16555.68	16923.11	16890.16	718.65	~ 0.2 sec	70.22%
ABC*k*	16436.95	16678.52	16574.49	188.13	~ 31.1 sec	71.47%

TABLE III: Resaults Obtained on CMC Data for 100 Runs

	Best	Worst	Average	Std. Dev.	CPU time	F-measure
k-means	5840.44	5934.50	5864.22	51.32	~ 0.4 sec	39.71%
ABC*k*	5700.82	5764.27	5711.27	3.41	~ 121.1 sec	42.31%

VI. Conclusion

The clustering problem is a very important problem and has attracted much attention of many researchers. The k-means algorithm is a simple and efficient clustering method that has been applied to many engineering problems; nevertheless it suffers from several drawbacks due to its choice of initializations. This paper has developed a combined algorithm for solving the clustering problem which is based on the combination of artificial bee colony algorithm and k-means technique. The algorithm has been implemented and tested on several well known real datasets and preliminary computational experience is very encouraging. In other word it has been proved that the ABC*k* algorithm will definitely converge to optimal solution in almost runs. The ABC*k* clustering algorithm developed in this paper can be applied when the number of clusters is known a prior.

References

[1] Y. T. Kao, E. Zahara, and I. W. Kao, "A hybridized approach to data clustering," *Expert Systems with Applications*, vol. 34, pp. 1754–1762, April 2008.

[2] D. N. Cao and J. C. Krzysztof, "GAKREM: a novel hybrid clustering algorithm," *Information Sciences*, vol. 178, pp. 4205–4227, November 2008.

[3] K. R. Zalik, "An efficient k-means clustering algorithm," *Pattern Recognition Letters*, vol. 29, pp. 1385–1391, July 2008.

[4] K. Krishna and M. N. Murty, "Genetic k-means algorithm," *IEEE Trans. on System Man Cybernetics*, vol. 29, pp. 433–439, June 1999.

[5] U. Mualik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique," *Pattern Recognition*, vol. 33, pp. 1455–1465, September 2000.

[6] M. Fathian and B. Amiri, "A honey-bee mating approach on clustering," *The International Journal of Advanced Manufacturing Technology*, vol. 38, pp. 809-821, September 2008.

[7] M. Laszlo and S. Mukherjee, "A genetic algorithm that exchanges neighboring centers for k-means clustering," *Pattern Recognition Letters*, vol. 28, pp. 2359–2366, December 2007 .

[8] P. S. Shelokar, V. K. Jayaraman, and B. D. Kulkarni, "An ant colony approach for clustering," *Analytica Chimica Acta*, vol. 509, pp. 187–195, May 2004.

[9] T. Niknam, J. Olamaie, and B. Amiri, "A hybrid evolutionary algorithm based on ACO and SA for cluster analysis," *Journal of Applied Science*, vol. 8, pp. 2695–2702, September 2008.

[10] T. Niknam, B. Bahmani Firouzi, and M. Nayeripour, "An efficient hybrid evolutionary algorithm for cluster analysis," *World Applied Sciences Journal*, vol. 4, pp. 300–307, June 2008.

[11] T. Niknam, B. Amiri, J. Olamaie, and A. Arefi, "An efficient hybrid evolutionary optimization algorithm based on PSO and SA for clustering," *Journal of Zhejiang University Science A*, vol. 10, pp. 512-519, April 2009.

[12] B. Zhang B. M. Hsu, and U. Dayal, "K-Harmonic means – A data clustering algorithm," Technical Report, HPL-1999-124, Hewlett-Packard Laboratories, 1999.

[13] G. Hammerly and C. Elkan, "Alternatives to the k-means algorithm that find better clusterings," in *Proc. the 11th ACM International Conference on Information and Knowledge Management*, Virginia, USA, 2002, pp. 600–607.

[14] F. Yang, T. Sun, and C. Zhang, "An efficient hybrid data clustering method based on k-harmonic means and particle swarm optimization," *Expert Systems with Applications*, vol. 36, pp. 9847–9852, August 2009.

[15] P. S. Shelokar, V. K. Jayaraman, and B. D. Kulkarni, "An ant colony approach for clustering," *Analytica Chimica Acta*, vol. 509, pp. 187–195, May 2004.

[16] V. D. Merwe and A. P. Engelbrecht, "Data clustering using particle swarm optimization," in *Proc. IEEE Congress on Evolutionary Computation (CEC 2003*, Canbella, Australia, 2003*)*, pp. 215-220.

[17] M. Omran, A. P. Engelbrecht, and A. Salman, "Particle swarm optimization method for image clustering," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 19, pp. 297–321, May 2005.

[18] D. Karaboga and C. Ozturk, "A novel clustering approach: artificial bee colony (ABC) algorithm," *Applied Soft Computing*, vol. 11, pp. 652–657, January 2011.

[19] C. Zhang, D. Ouyang, and J. Ning, "An artificial bee colony approach for clustering," *Expert Systems with Applications*, vol. 37, pp. 4761–4767, July 2010.

[20] W. Zou, Y. Zhu, H. Chen, and X. Sui, "A clustering approach using cooperative artificial bee colony algorithm," *Discrete Dynamics in Nature and Society*, vol. 2010, pp. 16, October 2010.

[21] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical Report, TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[22] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, pp. 459–471, November 2007.

[23] D. Karaboga and B. Basturk, "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems," *Lecture Notes in Computer Science*, vol. 4529, pp. 789–798, May 2007.

[24] D. Karaboga, B. Akay, and C. Ozturk, "Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks," *Lecture Notes in Computer Science*, vol. 4617, pp. 318–329, April 2007.

[25] A. Baykasoglu, L. Ozbakır, and P. Tapkan, "Artificial bee colony algorithm and its application to generalized assignment problem, Swarm Intelligence: Focus on Ant and Particle Swarm Optimization," *I Tech Education and Publishing*, Vienna Austria, pp. 113–144, 2007.

[26] M. F. Tasgetiren, Q. K. Pan, P. N. Suganthan, and A. H. L. Chen, "A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops," *Information Sciences*, vol. 181, pp. 3459–3475, August 2011.

[27] S. Z. Selim, M. A. Ismail, "K-means type algorithms: A generalized convergence theorem and characterization of local optimality," *IEEE Trans. on Pattern Analysis Machine Intelligence*, vol. 6, pp. 81–87, January 2009.

[28] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing*, vol. 8, pp. 687–697, January 2008.

[29] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, pp. 108–132, August 2009.

[30] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annual Eugenics*, vol. 7, pp. 179-188, 1936.

[31] M. Forina *et al*., "PARVUS, An extendible package for data exploration, classification and correlation. Institute of pharmaceutical and food analysis and technologies," Via Brigata Salerno, 16147 Genoa, Italy.

[32] T. S. Lim, W. Y. Loh, and Y. S. Shih, "A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms," *Machine Learning*, vol. 40, pp. 203-228, September 2000.

[33] A. Dalli, "Adaptation of the F-measure to cluster-based Lexicon quality evaluation," in *Proc. the EACL 2003 Workshop on Evaluation Initiatives in Natural Language Processing: are evaluation methods, metrics and resources reusable*, Budapest, 2003*?*, pp. 51-56.

Giuliano Armano is currently an associate professor of computer engineering at the University of Cagliari, and Head of the "Intelligent Agents and Soft-Computing" (IASC) group. His educational background ranges from expert systems to machine learning, whereas his current research activities focus on classifier ensemble methods, hierarchical classification, and feature dimensionality reduction (applied in particular to bioinformatics and information retrieval tasks).

Mohammad Reza Farmani is a Ph.D. student in the IASC (Intelligent Agents and Soft-Computing) group, in Department of Electrical and Electronic Engineering (DIEE), University of Cagliari. His research interests include evolutionary and bio-inspired computation, machine learning, data mining, and bioinformatics.