Neurocomputing 159 (2015) 9–26

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

An improved bee colony optimization algorithm with an application to document clustering

Rana Forsati [a,*], Andisheh Keikha [b], Mehrnoush Shamsfard [a]

[a] Natural Language Processing (NLP) Research Lab., Faculty of Electrical and Computer Engineering, Shahid Beheshti University, G. C., Tehran, Iran
[b] Department of Computer Science and Engineering, Ryerson University, Ontario, Canada

ARTICLE INFO

Article history:
Received 23 September 2013
Received in revised form
20 November 2014
Accepted 5 February 2015
Communicated by Dorothy Ndedi Monekosso
Available online 24 February 2015

Keywords:
Swarm intelligence
Bee colony optimization
Document clustering

ABSTRACT

The bee colony optimization (BCO) algorithm is proved to be one of the fast, robust and efficient global search heuristics in tackling different practical problems. Considering BCO algorithm in this paper, we utilize it for the data clustering, a fundamental problem that frequently arises in many applications. However, we discovered some obstacles in directly applying the ancient BCO to address the clustering problem and managed to change some basic behaviors of this swarm algorithm. In particular, we present an improved bee colony optimization algorithm, dubbed IBCO, by introducing *cloning* and *fairness* concepts into the BCO algorithm and make it more efficient for data clustering. These features give BCO very powerful and balanced exploration and exploitation capabilities to effectively guide the search process toward the proximity of the high quality solutions. In particular, the cloning feature allows it to take advantage of experiences gained from previous generations when generating new solutions. The problem of getting stuck in local optima still laid bare in the proposed improved version. As a result, to overcome the shortage of this swarm algorithm in searching locally, we hybridize it with the *k*-means algorithm to take advantage of fine tuning power of the widely used *k*-means algorithm which demonstrates good result in local searches. We propose four different hybridized algorithms based on IBCO and *k*-means algorithms and investigate the clustering results and convergence behavior of them. We empirically demonstrate that our hybrid algorithms alleviate the problem of sticking in a local solution even for large and high dimensional data sets such as document clustering. The results show that proposed algorithms are robust enough to be used in many applications compared to *k*-means and other recently proposed evolutionary based clustering algorithms including genetic, particle swarm optimization, ant colony, and bee based algorithms.

1. Introduction

Clustering is one of the crucial unsupervised learning techniques for dealing with massive amounts of heterogeneous information. The aim of clustering is to group a set of data objects into a set of meaningful sub-classes, called clusters which could be disjoint or not. Clustering is a fundamental tool in exploratory data analysis with practical importance in a wide variety of applications such as data mining, machine learning, pattern recognition, statistical data analysis, data compression, and vector quantization [88]. The aim of clustering is to find the hidden structure underlying a given collection of data points. In other words, in clustering, a set of patterns, usually vectors in a multi-dimensional space are classified in such a way that patterns in same clusters have more similarity to each other than the patterns in different clusters [35,76].

Some of the most conventional clustering methods can be broadly classified into two main categories [6,36]. The first category includes the hierarchical clustering methods. A hierarchical algorithm [30,41,68,96] creates a hierarchical decomposition of the given dataset forming a dendrogram Na tree which splits the dataset recursively into smaller subsets and represent the objects in a multi-level structure. The hierarchical procedures can be further divided into agglomerative or bottom-up algorithms and divisive or top-down algorithms [87]. In the first category, each element is initially assigned to a separate cluster, the algorithm then repeatedly merges pairs of clusters until a certain stopping criterion is met [87]. On the other hand, the divisive algorithms begin with the whole set of objects and proceed to divide it into a certain number of clusters successively.

Our concern in this paper is based on partitioning clustering [10] methods which include the most practical clustering algorithms especially for large data sets. The attempt is to divide the

* Corresponding author.
E-mail addresses: r_forsati@sbu.ac.ir (R. Forsati), andisheh.keikha@ryerson.ca (A. Keikha), m_shams@sbu.ac.ir (M. Shamsfard).

http://dx.doi.org/10.1016/j.neucom.2015.02.048
0925-2312/

data set into a set of disjoint clusters without the hierarchical structure. Partitioning methods try to partition a collection of objects into a set of groups, so as to maximize a pre-defined objective value. The most popular partitioning clustering algorithms are the prototype-based clustering methods where each cluster is represented by the center of the cluster and the used objective function (a square error function) is the sum of the distances from the patterns to the center [64].

Although hierarchical methods are often said to have better quality in clustering, they usually do not provide the reallocation of objects, which may have been poorly classified in the early stages of the analysis [36] and the time complexity of them declared to be quadratic [42]. On the other hand, in recent years the partitioning clustering methods showed a lot of advantages in applications involving large datasets due to their relatively low computational requirements [42,44]. The time complexity of the partitioning technique is almost linear, which makes it widely appealing in real world problems.

Among the partitioning clustering algorithms, especially the center-based clustering algorithms, the k-means algorithm [57] is most popular thanks to its simplicity and efficiency. Although the k-means algorithm is simple, straightforward and easy to implement and works fast in most situations, it suffers from some major drawbacks that make it inappropriate for many applications. The first disadvantage is that the number of clusters K must be specified prior to application. Also, since the summary statistic is mean of the values for each cluster, so, the individual members of the cluster can have a high variance and mean may not be a good summary of the cluster members. In addition, as the number of clusters grows, for example to thousands of clusters, k-means clustering becomes untenable, approaching the $O(n^2)$ comparisons where n is the number of points. However, for relatively few clusters and a reduced set of pre-selected words, the k-means algorithm can do well [84]. Another major drawback of k-means algorithm is sensitivity to initial centers. Finally, the k-means algorithm converges to the nearest local optimum from the starting position of the search and the final clusters may not be the optimal solution [25].

In order to overcome these problems that exist in traditional partitioning clustering methods especially k-means, recently, new concepts and techniques have been proposed in this area by researchers from different fields. One of these techniques is optimization methods that try to optimize a pre-defined function that can be very useful in data clustering. Optimization techniques define a global function to capture the quality of best partitioning and try to optimize its value by traversing the search space. Therefore different artificial intelligence based clustering methods, such as statistics [24], graph theory [93], expectation-maximization algorithms [65], artificial neural networks [62,51,70], evolutionary algorithms [76,21,72], swarm intelligence algorithms [83,71,39,73] have been proposed. In principle, any general purpose optimization method can serve as the basis for this approach. The methods such as genetic algorithm [63,33,67], simulated annealing, ant colony optimization [79], particle swarm optimization [18,71,50] and harmony search [25] have been used for data clustering in the context of other meta-heuristics. Also some algorithms based on the bees behavior have been proposed for this problem such as honey bee [23,94,77], the bees algorithm [73], and artificial bee colony algorithm [94,97,89,40]. Another swarm intelligence algorithm based on the bees' behavior is bee colony optimization [54–56] which is our focus in this paper to highlight the power of this optimization algorithm in data clustering problem.

The BCO algorithm [54–56] is a nature-inspired meta-heuristic optimization method, which is similar to the way bees in nature look for food, and the way optimization algorithms search for an optimum in combinatorial optimization problems. The performance of the BCO algorithm has been compared with those of other well-known heuristic algorithms such as genetic algorithm, differential evolutional algorithm, and particle swarm optimization algorithm for unconstrained optimization problems. The bee colony algorithm has been very successful in a wide variety of optimization problems [82] in engineering and control. In fact, in optimization problems, we want to search the solution space and in the BCO algorithm, this search can be done more efficiently. Since stochastic optimization approaches are good at avoiding convergence to a locally optimal solution, these approaches could be used to find a globally near-optimal solution [45].

The BCO algorithm belongs to the class of population-based techniques which is considered to be applied to find solutions for difficult combinatorial optimization problems. The major idea behind the BCO is to create the multi-agent system capable of efficiently solving hard combinatorial optimization problems. These features increase the flexibility of the BCO algorithm and produce better solutions. The bee colony behaves to some extent similar and to some extent in a different way from bee colonies in nature. They explore through the search space looking for the feasible solutions. In order to discover superior and superior solutions, artificial bees cooperate with each other and exchange information. Also, they focus on more promising areas and gradually discard solutions from the less promising areas via collective knowledge and giving out information among themselves.

As the behavior of the k-means algorithm mostly is influenced by the number of clusters specified and the random choice of initial cluster centers, in this study, we concentrate on the latter, where the results are less dependent on the initial cluster centers chosen, hence more stabilized by introducing different algorithms based on the BCO for clustering. In summary, the present work makes the following contributions:

- A basic bee colony based clustering (BCOCLUST) algorithm which solves the clustering problem with the ancient BCO method. This basic algorithm has some problems regarding some basic behaviors of the BCO algorithm that causes the bees to follow one solution after a while and get stock in a local optimum.
- An improved BCO algorithm by introducing *cloning* and *fairness* concepts into the BCO algorithm. These modifications are aimed at increasing the explorative power of the BCO algorithm and propagation of knowledge in an optimization process, respectively. The second proposed clustering algorithm is based on the improved BCO method and referred to as IBCOCLUST which proposes a better modeling for the specific application of clustering.
- Hybrid clustering algorithms using k-means and the IBCOCLUST algorithms. Although the problem of getting stuck in the local optimum has been solved in the IBCOCLUST method, the algorithm still suffers from locating the best solution in the proximity of the found global solution. The hybrid techniques alleviate this problem by combining the fine tuning capability of the k-means in the proximity of global solution and the searching power of the IBCOCLUST in locating the global solution. The hybrid methods improve the k-means algorithm by making it less dependent on the initial parameters such as randomly chosen initial cluster centers, hence more stable. It seems that the hybrid algorithms that combine two ideas can result in an algorithm that can outperform either one individually.
- To demonstrate the effectiveness and convergence rate of IBCOCLUST and hybrid algorithms, we have applied these algorithms on various standard datasets and got very promising results compared to the k-means and GA and PSO-based clustering algorithm [57,43]. BCO and PSO algorithms fall into the same class of artificial intelligence optimization algorithms, population-based algorithms,

and they are proposed by inspiration of swarm intelligence. Besides, comparing the BCO algorithm with the PSO algorithm, the performance of BCO algorithm is also compared with a wide set of classification techniques that are also given in [82].

- Also having in mind that document clustering is one of the major challenges in information extraction, to better evaluate the functionality of our proposed algorithms, we apply them on this important application as well. The evaluation of the experimental results based on accuracy, robustness, and convergence rate shows considerable improvement and robustness of the hybridized algorithms for large scale document clustering.

Outline: The paper is organized as follows. We begin in Section 2 by thoroughly surveying the related works that are mostly aligned to our work. In Section 3 we provide background on the basic algorithms including the clustering problem and the principles of BCO meta-heuristic. The basic BCO based clustering algorithm, the improved BCO algorithm along with the hybrid algorithms are discussed in Section 4. Section 5 presents the data sets used in our experiments, empirical study of BCO parameters on convergence behavior of the BCOCLUST algorithm. It also contains the performance evaluation of the proposed algorithms compared to well-known algorithms. The experiments and analysis of the proposed algorithms on the application of document clustering is also presented in Section 6. Finally Section 7 concludes the paper.

2. More related work

Earlier in the introduction, we discussed some of the main lines of research on clustering; here, we survey further lines of study that are directly related to our work on meta-heuristic based clustering algorithms.

Genetic algorithm based clustering: The genetic algorithm (GA) is inspired by the theory of natural selection and begins with a population of solutions which tries to survive in an environment (defined with fitness evaluation). The parent population shares their properties of adaptation to the environment to the children with various mechanisms of evolution such as genetic crossover and mutation. The process continues over a number of generations to find a desirable solution [28].

The GA has been extensively utilized for clustering problem. The paper [7] was among the first initially proposed the use of basic GA for partitional clustering and in particular document clustering [11]. The standard binary encoding scheme with a fixed number of cluster centers is used for initialization of chromosomes. The reproduction operation is carried out using uniform crossover and cluster-oriented mutation (altering the bits of binary string). Ravindra Krovi [47] investigated the potential feasibility of using genetic algorithms for the purpose of clustering. A novel hybrid genetic *k*-means algorithm, dubbed GKA, proposed by [45], which finds a globally optimal partition of a given data into a specified number of clusters. This hybrid method circumvents expensive crossover operations by using a classical gradient descent algorithm that is used in clustering using the *k*-means algorithm. Using finite Markov chain theory, it was proved that the GKA converges to the global optimum. The fast genetic *k*-means algorithm [52] (FGKA) was inspired by GKA but features several improvements over GKA. The incremental genetic *k*-means algorithm (IGKA) [53] was an extension to previously proposed FGKA clustering algorithm. IGKA outperforms FGKA when the mutation probability was small. The main idea of IGKA was to calculate the objective value total within-cluster variation and to cluster centroids incrementally whenever the mutation probability was small. IGKA inherits the salient feature of FGKA of always converging to the global optimum.

Ant colony based clustering: The ant colony optimization (ACO) algorithm is inspired by ants behavior in determining the optimal path from nets to the source of food [19]. The clustering problem in its optimization formulation can be solved utilizing the ACO method as explored in [79]. In [91] a multi-ant colonies approach for clustering data consists of some parallel and independent ant colonies and a queen ant agent. Each ant colony process takes different types of ants moving speed and different versions of the probability conversion function to generate various clustering results. A number of hybrid algorithms based on ACO method are available in the literature. Initially Kuo et al. [48] have proposed ants based *k*-means algorithm, which is subsequently improved by hybridization of ACO, self-organizing maps and *k*-means in [12]. Further, Jiang et al. have developed new hybrid clustering algorithms by combining the ACO with the *k*-harmonic means algorithm in [38] and the DSBCAN algorithm in [37]. The work in [32] utilized the ACO based clustering for document retrieval and the AntClust algorithm was introduced in [49] for web session clustering.

A new ant colony based method for text clustering using a validity index is introduced [90]. In this method the walking of the ants is mapped to the picking or dropping of projected document vectors with different probabilities. In another work Zhang et al. [95] suggests that the random movements of ants in the solution space lead to slow convergence. They provide a method for faster document clustering, called AFTC. The approach employs the pheromone laid by the ants to avoid randomness of movement, which lead the ants to move towards a direction with high pheromone concentration at each step. The direction of movement is the orientation where the text vectors are relatively more concentrated. A new text clustering approach named elite ant colony optimization clustering (EACOC), based on suitable retention of the elites has been introduced in [34]. The mechanism is to retain the elites that the algorithm works, in a way that in each iteration it retains a certain number of valuable solutions into the next cycle, with the purpose of improving algorithm performance. A new fully controllable ant colony algorithm (FCACA) for documents clustering has been introduced in [20]. This introduces a new version of the basic heuristic decision function that significantly improves the convergence and provides greater control over the process of the grouping data.

Particle swarm based clustering: The particle swarm optimization (PSO) algorithm is based on the swarming behavior of particles searching for food in a collaborative manner [13]. The cluster analysis using PSO was proposed in [69] for image clustering. Then, Van der Merwe and Engelbrecht [85] applied it for cluster analysis of arbitrary datasets. The algorithm in its basic form for cluster analysis consists of a swarm in a d dimensional search space in which each particle position consists of K cluster centroid vectors. A number of recent works tried to modify the PSO algorithm to make it more effective for clustering problem. In [14] the PSO algorithm was adapted to position prototypes (particles) in regions of the space that represent natural clusters of the input data set by influencing the particles' velocity update from previous position along with taking into account the past experiences. The hybrid algorithm based on *k*-means and PSO is proposed [14]. In [2] a PSO based clustering algorithm is proposed for web usage mining and clustering.

Biologically inspired based clustering: The biologically inspired algorithms comprise natural meta-heuristics derived from living phenomena and behavior of biological organisms. These algorithms encompass artificial immune systems [15] and bacterial foraging optimization [29]. These methods have recently been applied to clustering problem. In [86] a new clustering algorithm

Table 1
Summary of notations consistently used in the paper and their meaning.

Symbol	Meaning
n	The number of data objects
d	The ambient dimension of data objects
$\mathcal{D}=(\mathbf{d}_1,\ldots,\mathbf{d}_n)$	The set of objects to be clustered
K	The number of clusters
$\mathbf{A}\in\{0,1\}^{n\times K}$	The assignment matrix
$\mathcal{C}=\{\mathbf{c}_1,\mathbf{c}_2,\ldots,\mathbf{c}_K\}$	The cluster centers associated with an assignment matrix $\mathbf{A}$
$D_M(\cdot,\cdot):\mathbb{R}^d\times\mathbb{R}^d\to\mathbb{R}_+$	The Minkowski similarity measure between data points
$D_C(\cdot,\cdot):\mathbb{R}^d\times\mathbb{R}^d\to\mathbb{R}_+$	The Cosine similarity measure between data points
B	The number of bees in the hive
$\mathcal{B}=\{b_1,b_2,\ldots,b_B\}$	The set of B bees used in optimization
R	The number of recruiter bees
T	The total number of iterations
M	The number of constructive moves
S	The number of stages at each iteration of BCO algorithm

based on the mechanism analysis of bacterial foraging is proposed. It is an optimization based methodology in which a group of bacteria forage to converge to certain positions as final cluster centers by minimizing the fitness function. The initial solution space is created by assigning the bacteria positions as the randomly chosen cluster centroids in the data set. Then, the chemotaxis process defines the movement of bacteria, which represents either a tumble followed by a tumble or tumble followed by a run. Younsi and Wang [92] have developed and used a new artificial immune system algorithm for data clustering that uses the interaction between B-cells and antigens to tackle the optimization problem.

Harmony search based clustering: The harmony search (HS) algorithm is a meta-heuristic algorithm mimicking the improvisation process of musicians (where music players improvise the pitches of their instruments to obtain better harmony) [27,59]. The HS algorithm has been applied for various engineering optimization problems and in the context of clustering, novel partitional algorithms have been developed very recently. A new HS based document clustering with continuous representation is considered in [61]. In this algorithm each cluster centroid is considered as a decision variable; so each row of harmony memory, which contains K decision variables, represents one possible solution for clustering. The main drawback of the algorithm developed in [61] is its continuous representation. The continuous representation of clusters' centroid decreases the efficiency of pitch adjusting process. Another HS based document clustering with discrete representation called HKA is proposed in [60]. Furthermore, using a probabilistic analysis it has been shown that the proposed algorithm convergences to a near-optimal solution in a fairly reasonable amount of time. This algorithm codify the whole partition of the document set in a vector of length n, where n is the number of the documents. Considering the behavior of HKA, it was found that the proposed algorithm is good at finding promising areas of the search space, but not as good as k-means at fine-tuning within those areas, so it may take more time to converge. On the other hand, k-means algorithm is good at fine-tuning, but lack a global perspective. So a hybrid algorithm that combines two ideas is proposed in [26]. In the hybrid algorithm at each improvisation step a one-step k-means is leveraged to fine-tune the new solution.

3. Preliminaries

Notation: Throughout this paper, we use the following notation. We use bold-face letters to denote vectors. For any two vectors $\mathbf{d},\mathbf{d}'\in\mathbb{R}^d$, we denote by $\langle\mathbf{d},\mathbf{d}'\rangle$ the inner product between $\mathbf{d}$ and $\mathbf{d}'$, i.e., $\langle\mathbf{d},\mathbf{d}'\rangle=\sum_{i=1}^{d}d_id'_j$. We use bold upper case letter for matrices. Throughout this paper, we only consider the ℓ_2-norm which is defined as $\|\mathbf{x}\|_2=\sqrt{\sum_{i=1}^{d}x_i^2}$ for any vector $\mathbf{x}\in\mathbb{R}^d$. A summary of notations used in this paper is provided in Table 1.

3.1. The clustering problem

Clustering algorithms are commonly used to summarize large quantities of data, in a wide variety of domains. Clustering in a d-dimensional Euclidean space $\mathbb{R}^d$ is the process of partitioning a given set of n points into a fixed number of K clusters based on some similarity metric in a clustering procedure. The ith object is characterized by a real-value d-dimensional profile vector where each element corresponds to the jth real-value feature ($j=1,2,\ldots,d$). The vector space model gives us a good opportunity for defining different metrics for similarity between two data points.

Let $\mathcal{D}=\{\mathbf{d}_1,\mathbf{d}_2,\ldots,\mathbf{d}_n\}$ be a set of n objects and use $\mathbf{D}\in\mathbb{R}^{n\times d}$ to denote the corresponding data matrix. Given $\mathbf{D}$, the goal of a partitional clustering algorithm is to determine a partition $\mathcal{D}=\mathcal{D}_1\cup\mathcal{D}_2\cup\cdots\cup\mathcal{D}_K$ where each partition $\mathcal{D}_i$ is associated with a center $\mathbf{c}_i\in\mathbb{R}^d$ and denote by $\mathcal{C}=\{\mathbf{c}_1,\mathbf{c}_2,\ldots,\mathbf{c}_K\}$ the set of centers. Each data point $\mathbf{d}_i$ is assigned to the closest center, i.e., $\arg\min_{k\in\{1,2,\ldots,K\}}D(\mathbf{d}_i,\mathbf{c}_k)$ for a similarity measure $D(\cdot,\cdot)$ which will be discussed shortly. The goal is to find centers such that objects which belong to the same cluster are as similar to each other as possible, while objects which belong to different clusters are as dissimilar as possible. Over the years, two prominent ways have been used to compute the similarity between two data $\mathbf{d}$ and $\mathbf{d}'$. The first method is based on Minkowski distances which is for any two vectors $\mathbf{d}$ and $\mathbf{d}'$ is defined by

$$D_M(\mathbf{d},\mathbf{d}')=\left(\sum_{i=1}^{d}(d_i-d'_i)^p\right)^{1/p}, \tag{1}$$

which is equivalent to Euclidean distance for $p=2$.

The other commonly used similarity measure in data clustering is the cosine correlation measure [74], given by

$$D_C(\mathbf{d},\mathbf{d}')=\frac{\langle\mathbf{d},\mathbf{d}'\rangle}{\|\mathbf{d}_1\|\,\|\mathbf{d}_2\|}, \tag{2}$$

where $\|\cdot\|$ denotes the norm of a vector. This measure becomes one if both vectors are identical, and zero if there is nothing in common between them (i.e., the vectors are orthogonal to each other).

To find K centers, the problem is defined as an minimization of an objective function based on both the data points and the center locations. A popular function used to quantify the goodness of a partitioning is determined by sum of the intra-cluster distances

which is defined as [31,80]

$$f(\mathbf{D},\mathcal{C}) = \sum_{i=1}^{n} \min_{j=1,2,\ldots,K} D_*(\mathbf{d}_i, \mathbf{c}_j), \tag{3}$$

where $D_*(\mathbf{d}_i, \mathbf{c}_j)$ denotes either the Minkowski similarity or cosine distance between object $\mathbf{d}_i$ and center $\mathbf{c}_j$. The clustering problem aims to find the partitioning that has optimal adequacy with respect to the huge number of possible candidate partitioning, which is expressed in the form of the Stirling number of second kind[1] which indicates the complicated nature of the problem as

$$\frac{1}{K!}\sum_{i=1}^{K}(-1)^{K-i}\binom{K}{i}i^N.$$

It has been shown that the clustering problem is NP-hard even when the number of clusters is two [16]. This illustrates that the clustering by examining all possible partitions of n data vectors of d-dimensional into K clusters is not computationally feasible. The problem is even more demanding when additionally the number of clusters is unknown. Then the number of different combinations is the sum of the Stirling numbers of the second kind. As a result, exhaustive search methods are far too time consuming even with modern computing systems. Obviously, we need to resort to some optimization techniques to reduce the search space, but there is no guarantee that the optimal solution will be found. Although various optimization methodologies have been developed for optimal clustering, the complexity of the task reveals the need for developing efficient algorithms to precisely locate the optimum solution. In this context, this study presents a novel stochastic approach for data clustering, aiming at a better time complexity and partitioning accuracy.

3.2. The bee colony optimization meta-heuristic

The bee colony optimization (BCO) meta-heuristic has been proposed by Lucic and Teodorovic [54,55]. It is a simple and robust stochastic optimization algorithm and the basic idea is to create a colony of artificial bees capable of solving difficult combinatorial optimization problems successfully. The algorithm simulates the intelligent behavior of bee swarms. An artificial bee colony behaves to some extent like and to some extent in a different way from bee colonies found in the natural world. The performance of the BCO algorithm is compared with those of other well-known modern meta-heuristic algorithms such as genetic algorithm (GA), differential evolution (DE), and particle swarm optimization (PSO) on constrained and unconstrained problems [81,8].

The BCO is a model of collecting and processing nectar, the practice which is highly organized. Each bee decides to reach the nectar source by following a nest mate who has already discovered a patch of flowers. Each hive has a so-called dance floor area on which the bees that have discovered nectar sources, dance as a way to convince their nest mates to follow them. If a bee decides to leave the hive to get nectar, she follows one of the bee dancers to one of the nectar areas or stick to her nectar source. Upon arrival, the foraging bee takes a load of nectar and returns to the hive relinquishing the nectar to a food-store bee. After she relinquishes the food, the bee can (a) abandon the food source and become again an uncommitted follower; (b) continue to forage at the food source without recruiting nest mates; or (c) dance and thus recruit nest mates before returning to the food source. The bee opts for one of the above alternatives with a certain probability. Within the dance area, the bee dancers advertise different food sources.

The BCO algorithm consists of two alternating phases: a forward pass and a backward pass. During each forward pass, every bee is exploring the search space and creating various partial solutions. It applies a predefined number of moves (visiting a certain number of nodes), which construct and/or improve the solution, yielding a new solution. During the second forward pass, bees will visit few more nodes, expand previously created partial solutions. Having obtained new partial solutions, the bees return to the nest and start the second phase, the so-called backward pass. During the backward pass, all bees share information about their solutions. In the nest, all bees participate in a decision-making process and exchange information about quality of the partial solutions created. Bees compare all generated partial solutions.

During the backward pass, based on the quality of the partial solutions generated, every bee decides with a certain probability whether it will advertise its solution or not. The bees with better solutions have more chances to advertise their solutions. The remaining bees have to decide whether to continue to explore their own solution in the next forward pass, or to start exploring the neighborhood of one of the solutions being advertised. Similarly, this decision is taken with a probability, so that better solutions have a higher probability of being chosen for exploration. Depending on the quality of the partial solutions generated, every bee possesses certain level of loyalty to the path leading to the previously discovered partial solution. The search process is composed of iterations. The first iteration is finished when bees create for the first time one or more feasible solutions by visiting all nodes.

The two phases of the search algorithm namely the forward and backward passes, are performed iteratively until a stopping condition is satisfied. The possible stopping conditions could be, for example, the maximum number of iterations or the number of iterations without the improvement of the objective function. The best discovered solution during the first iteration is saved, and then the second iteration begins. Within the second iteration, bees again incrementally construct solutions of the problem, etc. There are one or more solutions at the end of each iteration. The analyst-decision maker prescribes the total number of iterations. The detailed steps of optimizing function $f(\mathbf{x})$ for decision variable $\mathbf{x}$ are summarized as follows. We set the number of stages S to be same as the number of decision variables. We also let B denote the number of bees in the hive, T denote the total number of iterations, and M represents the number of constructive moves during one forward.

1. *Initialization:* Every bee is set to be an empty solution. A feasible solution of the problem is initialized as the best solution $\mathbf{x}_*$.
2. *Optimization:* The following steps are iterated for $t = 1, 2, \ldots, T$ iterations:
   (a) Set the current stage to be one, i.e., $s \leftarrow 1$.
   (b) For every bee do the *forward* pass as
      (i) $m \leftarrow 1$ //counter for constructive moves in the forward pass.
      (ii) Evaluate all possible constructive moves.
      (iii) According to evaluation, choose one move using the roulette wheel or tournament.
      (iv) $m \leftarrow m+1$; If $m <= M$ then go to Step ii.
   (c) All bees are back at the hive and *backward* pass starts. Allow bees to exchange information about quality of the partial solution created.
   (d) Evaluate (partial) objective function value for each bee.
   (e) Sort the bees by their objective function value.

[1] The Stirling numbers of the second kind [78], that is usually denoted by $S(n,k)$ or $\{{n \atop k}\}$ is the number of ways to partition a set of n labelled objects into k nonempty unlabelled subsets. Equivalently, it captures the number of different equivalence relations with precisely k equivalence classes that can be defined on an n element set. For example, the set $\{1,2,3\}$ can be partitioned into three subsets in one way: $\{\{1\},\{2\},\{3\}\}$; into two subsets in three ways: $\{\{1,2\},\{3\}\}$, $\{\{1,3\},\{2\}\}$, and $\{\{1\},\{2,3\}\}$; and into one subset in one way: $\{\{1,2,3\}\}$. Obviously, $\{{n \atop 1}\} = \{{n \atop n}\} = 1$. The $S(n,k)$ can be explicitly calculated by $S(n,k) = (1/k!)\sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n$.

(f) Every bee decides randomly whether to continue its own exploration and become a recruiter, or to become a follower (bees with a higher objective function value have a greater chance to continue their own exploration).
(g) For every follower choose a new solution from recruiters by the roulette wheel.
(h) $s \leftarrow s+M$, and if $s < S$, go to Step (b).
(i) If the best solution $\mathbf{x}_t$ obtained during the tth iteration is better than the best-known solution, update the best known solution ($\mathbf{x}_* \leftarrow \mathbf{x}_t$)
(j) $t \leftarrow t+1$.

3. *Output:* Report the best solution $\mathbf{x}_*$ as the final solution.

4. The basic bee colony based algorithm to data clustering

In this section we first propose a pure bee colony based clustering algorithm namely BCOCLUST. We begin in Section 4.1 by giving detailed steps of the first proposed algorithm (BCOCLUST). The nature of BCO algorithm [82,8] is to force the un-loyal bees to follow only the loyal bees. In some problems like clustering, this behavior of BCO algorithm leads to getting stuck into a local optimum, since most of the bees start following a loyal bee which is foraging for a local optimum solution. To overcome this problem, we made a change in this behavior and propose Improved BCOCLUST (IBCOCLUST) algorithm in Section 4.2. To achieve even better clustering, the explorative power of IBCOCLUST is combined with the refining power of the k-means in four ways. Contrary to the localized searching property of k-means algorithm, the proposed algorithms perform a globalized search in the entire solution space. Additionally, the proposed algorithms improve k-means by making it less dependent on the initial parameters such as randomly chosen initial cluster centers, therefore, making it more stable. The details of these hybridization algorithms are described in Section 4.3.

4.1. BCOCLUST: bee colony based clustering

In order to cluster data using BCO, we must recast clustering as an optimization task that locates the optimal centroids of the clusters rather than to find an optimal partitioning, with clustering quality as the objective, and to use a suitable general purpose optimization method to find a good clustering. The principal advantage of this approach is that the objective of the clustering is explicit, which enables us to better understand the performance of the clustering algorithm on particular types of data and to use task-specific clustering objectives. It is even possible to consider several objectives simultaneously, an approach explored recently in [94]. This model offers us a chance to apply BCO algorithm on the optimal clustering of a collection of data.

Recall that the goal is to partition n vectors each of them with d dimensions into K clusters. In each iteration of our algorithm, for each cluster, all the bees leave the hive to allocate some of the data to that cluster with special probabilities and come back to the hive to see the work of other bees and decide whether or not to continue with their own decision or select an another bees solution to go on with. We consider each cluster centroid as a decision variable; so each solution extracted by a bee after each iteration, which contains K decision variables, represents one possible solution for clustering. On the other hand, each solution contains a number of candidate centroids that represents each cluster. In this case, each extracted solution contains K vectors $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K\}$.

Viewing the clustering problem as an optimization problem of such an objective function formalizes the problem to some extent. However, we are not aware of any function that optimally captures the notion of a good cluster, since for any function one can exhibit cases for which it fails. Furthermore, not surprisingly, no polynomial time algorithm is known for optimizing such cost functions. The brute force solution would be to enumerate all possible clusterings and pick the best. As the number of possible partitioning grows exponentially, this approach is not feasible. The key design challenge in objective function-based clustering is the formulation of an objective function capable of reflecting the nature of the problem so that its optimization reveals meaningful structure (clusters) in the data. The following subsections describe our modeling and bee colony operators, according to this model for clustering purpose.

4.1.1. Representation of solutions

The first attempt to solve data clustering by the BCO is how to represent solutions to exploit bee colony algorithm. To this end, we decompose data clustering into K stages where each stage targets one cluster center. For instance, the first stage represents the first cluster center; the second stage represents the second cluster center and so on. At each stage $s \in \{1, 2, \ldots, K\}$, every bee should select a subset of objects which will be allocated to the cluster s and these objects will be used to compute $\mathbf{c}_s$. We note that each individual bee consists of an encoding of a candidate solution (food source) and a fitness that indicates its quality. In order to apply it to solve clustering problem, we have used floating point arrays to encode cluster centers. Let $\mathbf{A} \in \{0,1\}^{n \times K}$ be the assignment matrix, with n rows and K columns where $[a_{ij}], 1 \le i \le n, 1 \le j \le K$ indicates whether or not the $i$the data is assigned to cluster j, i.e.,

$$a_{ij} = \begin{cases} 1 & \text{if } i\text{th data is assigned to } j\text{th cluster} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The assignment matrix $\mathbf{A} = [a_{ij}]$ has the properties that each a_{ij} must assigned exactly to one cluster (i.e., $\sum_{j=1}^{K} a_{ij} = 1$ for $i = 1, 2, \ldots, n$). An assignment that represents K nonempty clusters is a legal assignment. In this model, each food source discovered by each bee is a candidate solution and corresponds to a set of K centroids. So, the search space is the space of all matrices $\mathbf{A} \in \{0,1\}^{n \times K}$ that satisfy the constraint in which each data must be allocated to exactly one cluster and there is no empty cluster. Each stage involves optimizing one variable. We set the number of stages to be K and the number of bees to participate in the search process to be B. The algorithm proceeds in T iterations. Each bee at each stage $s = 1, 2, \ldots, K$ decides about the set of objects to be assigned to the cluster s. An example of representation of solutions is reported in Fig. 1. The number of solution components to be visited within one forward pass is set to one ($m = 1$). Therefore, at each forward pass, bees are supposed to visit a single stage.

All bees are located in the hive at the beginning of the search process. Each artificial bee allocates a subset of the data to the corresponding cluster with specified probabilities in each stage, and in this way constructs a solution of the problem incrementally. Bees are adding solution components to the current partial solution until they visit all of the K stages. The search process is composed of iterations. The first iteration is finished when bees create feasible solutions. The best discovered solution during the first iteration is saved, and then the second iteration begins. In each iteration of proposed algorithm, for each cluster (stage), all the bees leave the hive to allocate some of the data to that cluster with special probabilities and come back to the hive to see the work of other bees until that time and decide whether to continue its way or select one of the other bees solution and continue on that way.

4.1.2. Evaluation of solutions

A key characteristic of most clustering algorithms is that they use a global criterion function whose optimization drives the entire clustering process. For those clustering algorithms, the clustering problem can be stated as computing a clustering solution such that the value of a particular objective function is optimized. Our objective function is to minimize intra-cluster similarity while maximizing the inter-cluster similarity. Fitness value of each solution, which corresponds to one potential solution, is determined by sum of the intra-cluster distances (SICD) as detailed in Eq. (8).

Clearly, the smaller the sum of the distances is, the higher the quality of clustering. A food source represents a possible solution to the problem. The quantity of existing sources of nectar in the areas are explored by the bees corresponds to the quality of the solution represented by that food source. Each iteration of the BCOCLUST algorithm is detailed as the following:

Step 1. *Initialization:* The first step is the initialization. If this is not the first iteration of the algorithm and the best discovered cluster centers during the previous iterations are available, the initial cluster centers for all the stages are set to the best answer of the previous iteration. In other words, after generation of a set of initial solutions obtained in the previous iteration which described in the next steps, we rank the initial solutions based on the fitness function and set the K best of them as the initial cluster centers. Otherwise, if this is the first iteration, a set of initial cluster centers generated randomly from the data points in $\mathcal{D}$ will be set for each cluster. Each solution represents K cluster centers. The cluster centers of the ith stage are randomly selected from the uniform distribution over the set and indicate the cluster center of ith stage.

There is a loop from 1 to K where in each loop the following two steps are performed:

Step 2. *Constructive moves in the forward pass (allocate data to cluster):* In each forward pass, every artificial bee visits one stage, allocates the data to the corresponding cluster, and after that returns to the hive as detailed in Step 3. There is a loop from 1 to K where within each loop, the data will be allocated to the corresponding cluster. For each cluster, the probability of a bee choosing the data i as a member of jth cluster ($\mathcal{C}_j$), p_{ij}, is expressed using the Logit model as follows:

$$p_{ij}=\frac{\exp(-D_*(\mathbf{d}_i,\mathbf{c}_j))}{\sum_{m=1}^{n}\exp(-D_*(\mathbf{d}_m,\mathbf{c}_j))},\quad j=1,2,\ldots,K \tag{5}$$

where $D(\mathbf{d}_i,\mathbf{c}_j)$ denotes the distance of ith data to jth cluster. For all data points which have not been assigned yet, this process works in this way that for each non-allocated data, a random number is generated, if the number is less than datas' allocation probability, then the data will be allocated to cluster j with probability $1/K$ or will be set free. Within each forward pass a bee visited a certain number of nodes and created a partial solution. After solutions are evaluated (and normalized) the loyalty decision and recruiting process are performed as described in the following subsection.

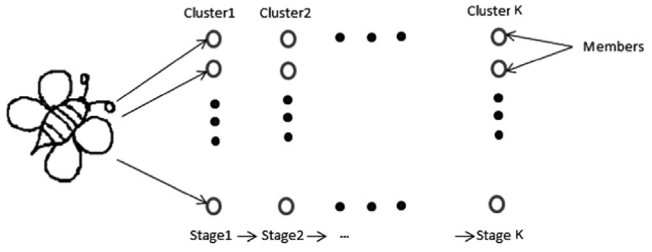

Fig. 1. Representation of BCO in clustering.

Step 3. *Backward pass (Bees partial solutions comparison mechanism):* After all of the bees completed Step 2, they will be back to hive to compare their partial solutions with themselves. We assume that every bee can obtain the information about solutions' quality generated by all other bees. In this way, bees compare all generated partial solutions. Based on the quality of the partial solutions generated, every bee decides whether to abandon the created partial solution or dance and thus recruit the nestmates before returning to the created partial solution. Depending on the quality of the partial solutions generated, every bee possesses certain level of loyalty to the previously discovered partial solution. Our criterion to decide about the goodness of discovered solution in general is sum of the distance of each vector from its cluster center for all the vectors. We want this criterion to be as minimal as possible. So as the bees are back at the hive, the probability that bth bee (during stage $s\in\{1,2,\ldots,K\}$ and iteration $t\in\{1,2,\ldots,T\}$) will be faithful to its previously generated partial solution (loyalty decision) is expressed as follows:

$$p_b(s+1,t)=e^{-O_b(s,t)/(s\times t)},\quad b=1,2,\ldots,B, \tag{6}$$

where $O_b(s,t)$ is the normalized value of $SICD_b$ defined as

$$O_b(s,t)=\frac{\mathrm{SICD}_b(s,t)-\mathrm{SICD}_{\min}(s,t)}{\mathrm{SICD}_{\max}(s,t)-\mathrm{SICD}_{\min}(s,t)} \tag{7}$$

where $\mathrm{SICD}_{\max}$ and $\mathrm{SICD}_{\min}$ denote the objective function value for the worst and the best discovered partial solution between all the bess, respectively, and SICD_b is sum of the distances of each object from its cluster center for all the objects that has been assigned by bee b as shown below:

$$\mathrm{SICD}_b(s,t)=\sum_{i=1}^{s}\sum_{j=1}^{n}D_{ji}^{b}. \tag{8}$$

$$D_{ij}^{b}=\begin{cases}D_*(\mathbf{d}_j,\mathbf{c}_i^b) & \text{if } j\text{th data is assigned to } i\text{th cluster by bee } b\\ 0 & \text{otherwise}\end{cases} \tag{9}$$

where $\mathbf{c}_i^b$ is the center for cluster i decided by bee b.

$$\mathrm{SICD}_{\min}(s,t)=\min_{i\in\{1,2,\ldots,B\}}\mathrm{SICD}_i(s,t) \tag{10}$$

$$\mathrm{SICD}_{\max}(s,t)=\max_{i\in\{1,2,\ldots,B\}}\mathrm{SICD}_i(s,t) \tag{11}$$

Let us discuss Eq. (6) in more detail. The better partial solution (i.e., lower O_b value) will increase the probability that bee b will be loyal to the previously discovered partial solution. Additionally, the greater the ordinary number of the forward passes, the higher the influence of the already discovered partial solution will be. This is expressed by the term s in the denominator of the exponent. In other words, at the beginning of the search process, bees are more brave to search the solution space. The more forward passes they make, the less the bees have courage to explore the solution space. The more we are approaching the end of the search process, the more focused the bees are on the already known partial solution.

4.1.3. Recruiting process

At the beginning of a new stage, if a bee does not want to expand the previously generated partial solution, the bee will go to the dancing area and will follow another bee. Within the dance area the bee dancers (recruiters) advertise different partial solutions. We have assumed in this paper that the probability that the partial solution of bee b would be chosen by any uncommitted bee is equal to:

$$p_b=e^{-\gamma O_b(s,t)},\quad b=1,2,\ldots,R \tag{12}$$

Table 2
The statistics of general purpose benchmark data sets used in our first set of experiments.

Dataset	# Attributes (d)	# Classes (K)	# Samples (n)
Iris	4	3	150
Wine	13	3	178
Glass	9	6	214
Wisconsin breast cancer	9	2	683
Vowel	3	6	871

where $\gamma \in (0,1)$ and R denotes the number of recruiters. The probability p_b is used in a roulette wheel selection or tournament selection algorithm and one of the bees is selected.

Using Eq. (12) and a random number generator, every uncommitted follower joins one bee dancer (recruiter). Recruiters fly together with their recruited nest mates in the next forward pass along the path discovered by the recruiter. So the bee that wants to continue another partial solution will set its partial solution exactly as the selected bee but will continue the algorithm independently. At the end of this path, all bees are free to independently search the solution space and generate the next iteration of constructive moves. It can be inferred from Eq. (7) that if a bee has discovered the lowest distance in stage s of iteration t, the bee will fly along the same partial path with the probability equal to one. The higher the sum of the distance of a discovered path the smaller the probability of flying again along the same path.

Step 4. *Allocation of non-allocated data:* After the loop of K clusters has finished there might be some data that has been not allocated to any cluster yet for each bee. These vectors will be allocated to each cluster with a greedy algorithm which means each vector belongs to the cluster that its cluster center is the nearest center to that vector.

Step 5. *Setting the cluster centers (computing the centroid of clusters):* At last, the cluster centers as the centroid of the vectors belong to each cluster for each bee are computed as follows: each solution extracted by each bee corresponds to a clustering with assignment matrix $\mathbf{A}$. Let $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K\}$ be the set of K centroids for assignment matrix $\mathbf{A}$. The kth centroid is computed as

$$c_k = \sum_{i=1}^{n} a_{ki}\mathbf{d}_i \Big/ \Big[\sum_{j=1}^{n} a_{kj}\Big]. \tag{13}$$

Step 6. *Selecting the best answer:* In this phase, among all generated solutions, the best one is determined and is used to update the global best. The global best will be used for setting the cluster centers for all the stages in next iteration. At this point, all B solutions are deleted, and the new iteration starts. The BCO proceeds iteratively until a stopping condition is met.

4.2. IBCOCLUST: improved BCOCLUST algorithm

A major shortcoming of the BCOCLUST algorithms that leads to unreasonable results is the low diversity of solutions in the course of search process. This phenomenon is the consequence of the nature of the algorithm where all the bees start to follow the one whose answer is the best among others (i.e., exploitation) after a few iterations, and so the answer will converge to a local optimized one. To overcome this problem, we present the Improved BCOCLUST (IBCOCLUST) algorithm which is based on two major modifications we have made to the original BCO algorithm: fairness and cloning. The main insight which underlines the proposed improved algorithm and the cloning and fairness ideas stems from, is as follows. For meta-heuristics algorithm such as BCO the main tricky point is to balance the trade-off between exploration and exploitation. On one hand, a high value for the exploitation rate forces the algorithm to mostly stick to the exiting solutions found by the bess (i.e., exploitation) and consequently leading to less exploration of the solution space. On the other hand, by choosing a small value for exploitation, the algorithm performs a random behavior in the solution space (i.e., exploration), hence losing all the information collected during the past rounds which deteriorates the effectiveness of the algorithm. The cloning idea is to give an erasable amount of exploitation to the BCO algorithms which is lacked in the original algorithms. The detailed description of these two features is as follows.

Table 3
Model selection for the number of bees. For each data set we fix the value of other parameters including the maximum number of iteration and γ, and evaluate the performance of the IBCOCLUST algorithm for different number of bees. Recall that T denotes the maximum number of iterations.

Constant Configuration	Data set	Number of bees	SICD
$\gamma = 1$	Iris	2	123.76
$T = 1000$		5	122.18
		7	122.16
		8	122.15
		10	122.15
		12	**97.23**
		15	97.23
		16	97.35
		17	97.34
		20	97.23
		24	97.23
		25	97.34
		30	97.23
$\gamma = 0.01$	Glass	2	250.44
$T = 1000$		3	246.66
		6	248.92
		8	228.64
		9	219.85
		12	220.56
		15	218.72
		16	220.13
		17	219.74
		18	**214.74**
		20	214.85
		25	220.96
		30	217.68

Fairness: With this modification, we aim at giving chance to every bee to be followed. In particular, in the improved algorithm after a bee decides to follow another bee, it is not forced to follow only loyal bees and it may consider both loyal and non-loyal ones to follow. In other words, there is no restriction in following just loyal bees and every bee can follow any other bee. It is obvious that the chance that a loyal bee is selected is much more than the chance that a recruiter is selected due to their fitness that is included in the probability of selection. By giving chance to non-loyal bees to be followed, even with a small probability, the algorithm is able to keep the diversity of solutions in a reasonable level and therefore having a much better explorative power.

Cloning: Another improvement we have made is to resolve the forgiveness characteristic of the BCO algorithm. In the standard BCO, the iterations of the algorithm are independent and no propagation of knowledge happens between different iterations. But since the best solution of each iteration, potentially carries all the information the algorithm learns in the course of that specific iteration, it would be better to incorporate this knowledge in next iteration. To this end, we propose the following novel idea to propagate the information during the optimization. Let $\mathcal{C}_*^{t-1} = \{\mathbf{c}_*^{t-1,1}, \mathbf{c}_*^{t-1,2}, \ldots, \mathbf{c}_*^{t-1,K}\}$ denote the best clustering solution obtained till iteration $t-1$. In the tth iteration, we add a specific bee to the set of bees called *cloning bee* that behaves as follows. The cloning bee is similar to other bees with

Table 4
Model selection for parameter γ. For each data set we fix the value of other parameters including the number of bees and the maximum number of iteration and evaluate the performance of the algorithm for different values of γ. Recall that T and B denote the maximum number of iterations and the number of bees, respectively.

Constant configuration	Data set	Variable parameter	Value	SICD
$B=12$ $T=1000$	Iris	γ	0.0001	97.33
			0.0005	97.34
			0.001	97.23
			0.005	97.33
			0.01	97.34
			0.05	97.33
			0.1	97.35
			0.5	97.45
			1	**97.23**
$B=12$ $T=1000$	Wine	γ	**0.0001**	**16,453.28**
			0.0005	16,460.80
			0.001	16,455.28
			0.005	16,460.00
			0.01	16,476.80
			0.05	16,460.80
			0.1	16,497.59
			0.5	16,465.50
			1	16,567.45
$B=12$ $T=1000$	Vowel	γ	0.0001	149,786.85
			0.0005	149,786.85
			0.001	150,769.34
			0.005	149,786.85
			0.01	14,998.34
			0.05	149,786.85
			0.1	**149,466.23**
			0.5	150,093.05
			1	150,693.05
$B=18$ $T=1000$	Glass	γ	0.0001	218.98
			0.0005	218.90
			0.001	248.55
			0.005	223.90
			0.01	**218.26**
			0.05	223.84
			0.1	223.94
			0.5	223.90
			1	223.14

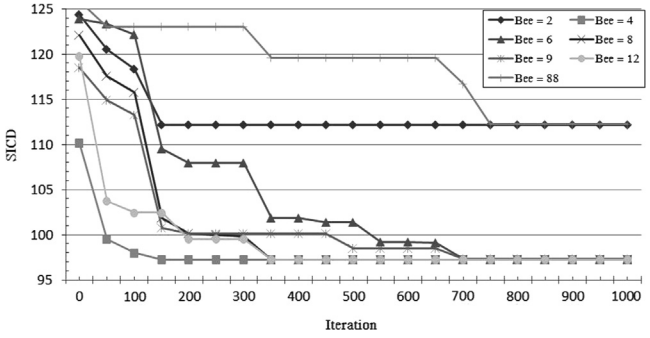

Fig. 2. Convergence behavior of IBCOCLUST on Iris dataset with $\gamma=1$ and different number of bees.

the difference that in the forward pass, it is forced to follow the decision of $\mathcal{C}_*^{t-1}$. More specifically, at stage s of iteration t, the cloning bee's decision is the subset of data points in $\mathcal{D}$ that are assigned to sth cluster by $\mathcal{C}_*^{t-1}$, i.e., the cluster represented by $\mathbf{c}_*^{t-1,s}$.

Remark 1. The main insight that underlines the cloning and fairness ideas stems from is as follows. For the meta-heuristics algorithm such as BCO the main tricky point is to balance the

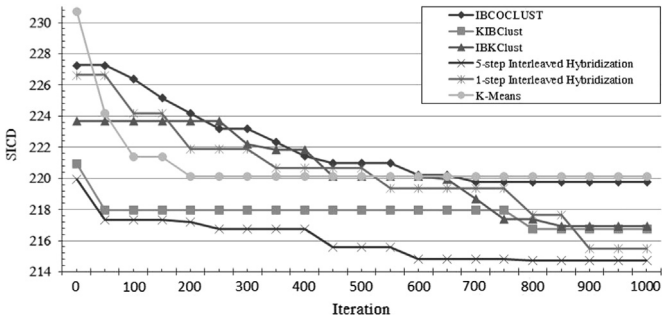

Fig. 3. Convergence behavior of proposed algorithms on glass data set with number of bees $B=9$ and $\gamma=0.01$.

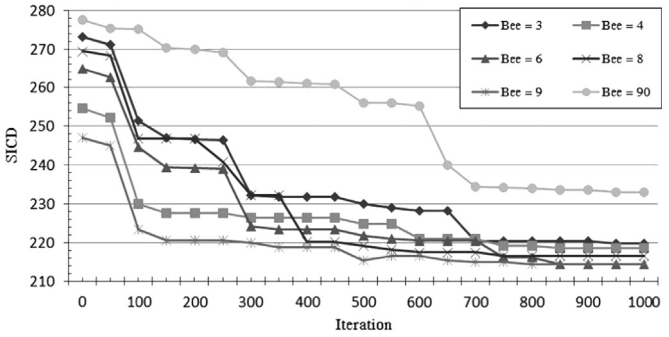

Fig. 4. Convergence behavior of IBCOCLUST on glass data set with $\gamma=0.01$.

trade-off between exploration and exploitation. On one hand, a high value for the exploitation rate forces the algorithm to mostly stick to the exiting solutions found by the bees (i.e., exploitation) and consequently leading to less exploration of the solution space. On the other hand, by choosing a small value for exploitation, the algorithm performs a random behavior in the solution space (i.e., exploration), hence losing all the information collected during the past rounds which deteriorates the effectiveness of the algorithm. The cloning idea is to give a reasonable amount of exploitation to the BCO algorithms which is lacked in the original algorithms. In other words a chance is given to learn from the experiences of other bees based on the quality of the solutions they have found to reduce the amount of search done be all bees.

The other parts of the algorithm such as initialization, selecting the bees to be loyal or non-loyal are exactly same as the BCOCLUCT. The pseudo-code of the IBCOCLUST is demonstrated in Algorithm 1.

Algorithm 1. IBCOCLUST.

Input: The number of iterations T, number of clusters K, number of bees B, the data set $\mathcal{D}=\{\mathbf{d}_1,\mathbf{d}_2,\ldots,\mathbf{d}_n\}$

Initialize $\mathcal{C}_*\leftarrow$A random valid clustering,
for $t=1,\ldots,T$ **do**
  for $b=1,\ldots,B$ **do**
    if $t>0$ **then**
      Set the initial cluster centroid of the b to $\mathcal{C}_*$
    else
      Select a random data as the centroid of each cluster of each b
    end if
  end for
  for $s=1,\ldots,K$ **do**
    for $b=1,\ldots,B$ **do**
      Select point **d** by tournament selection

13: $r \leftarrow U(0,1)$ where U is the uniform random generator
14: **if** $r < \frac{1}{K}$ **then**
15: **Allocate** selected point to the current cluster
16: **end if**
17: **end for**
18: **for** $b = 1, \ldots, B$ **do**
19: **Calculate** the probability of sticking to its solution
20: $r \leftarrow U(0,1)$
21: **if** $r <$ sticking to its solution**then**
22: The specified bee will stick to its own solution
23: **else**
24: Select another bee by tournament selection and choose its solution
25: **end if**
26: **end for**
27: **end for**
28: **Allocate** non-assigned data points with a greedy algorithm
29: Select the best solution at iteration t and set to be the $\mathcal{C}_*$
30: **end for**
Return $\mathcal{C}_*$

4.3. Bee colony k-means clustering

The IBCOCLUST algorithm performs a global search for solutions whereas the k-means clustering procedure performs a local search. In a local search, the solution obtained is usually located in the proximity of the solution obtained in the previous step. For example, the k-means clustering algorithm uses the randomly generated seeds as the initial clusters centroids and refines the position of the centroids at each iteration. The refining process of the k-means algorithm indicates that the algorithm only explores the very narrow proximity, surrounding the initial randomly generated centroids and its final solution depends on these initially selected centroids. So, the IBCOCLUST and k-means algorithms have complementary strong and weak points, the IBCOCLUST is good at finding promising areas of the search space, but not as good as k-means at fine-tuning within those areas. On the other hand, the k-means algorithm is good at fine-tuning, but lacks a global perspective. It seems a hybrid algorithm that combines the IBCOCLUST with k-means can result in an algorithm that can outperform either one individually.

The following lemma shows that the k-means algorithm always improves the objective function.

Lemma 1. *The k-means algorithm monotonically decreases the objective until local convergence, i.e.,* $f(\mathbf{D}, \mathcal{C}^{t+1}) \leq \mathbf{f}(\mathbf{D}, \mathcal{C}^t)$.

Proof. Let $\mathcal{C}^t = (\mathbf{c}_1^t, \ldots, \mathbf{c}_K^t)$ be the cluster centers at the tth iteration of the k-means algorithm which partitions the data points into the sets $\mathcal{D}_1^t, \ldots, \mathcal{D}_K^t$, where $\mathcal{D}_i^t, i \in [K]$ is the set of points assigned to ith cluster based on $\mathcal{C}_t$. The objective function for this clustering is as

$$f(\mathbf{D}, \mathcal{C}^t) = \sum_{k=1}^{K} \sum_{\mathbf{d} \in \mathcal{D}_k} \|\mathbf{d} - \mathbf{c}_k^t\|^2$$

which is the sum of the distances of data points to the centers of the assigned clusters. Let $\mathcal{C}^{t+1} = (\mathbf{c}_1^{t+1}, \ldots, \mathbf{c}_K^{t+1})$ denote the solution at $(t+1)$ th iteration by applying two steps of the k-means algorithm. Similarly, let $\mathcal{D}_1^{t+1}, \ldots, \mathcal{D}_K^{t+1}$ denote the set of the data points assigned to each cluster at the end of $(t+1)$ th iteration.

We consider the effect of each step of k-means algorithm separately. The reassignment step results in a non-increasing objective since the distance between a data point and its newly assigned cluster mean never increases the objective. Similarly, the mean update step results in an increasing objective since the mean is the best representation of a cluster in terms of the squared Euclidean distance, i.e.,

$$\sum_{k=1}^{K} \sum_{\mathbf{d} \in \mathcal{D}_k^{t+1}} \|\mathbf{d} - \mathbf{c}_k^{t+1}\|^2 \leq \sum_{k=1}^{K} \sum_{\mathbf{d} \in \mathcal{D}_k^{t}} \|\mathbf{d} - \mathbf{c}_k^{t}\|^2.$$

The fact that the algorithm will converge locally follows from the fact that the objective function cannot increase and there are only a finite number of possible clustering of data. □

We note however that the number of iterations required to reach convergence can be exponentially large, and furthermore, there is no any non-trivial lower bound on the gap between the value of the k-means objective of the algorithms output and the minimum possible value of that objective function.

Although Lemma 1 indicates that the k-means algorithm monotonically decreases the objective, but it might find locally optimal solutions with respect to the clustering error. This is due to the non-convex nature of criterion functions (sum of the squared Euclidean distance) in terms of both centers and assignment of data points, the iterative relocation methods are often trapped into one of the local minima. As mentioned before, the problem in general in NP-hard (see e.g., few recent results on proving this claim [3,17,58]). Hence the quality of a final clustering solution depends and is very sensitive to the initial configuration and the obtained partition is often only suboptimal (not the globally best partition). This deficiency becomes more serious for applications such as which intensifies the hardness of the problem due to low quality of text data, high-dimension and sparseness properties of documents.

To illustrate this fact, we provide a simple setting which shows the hardness of the problem and sub-optimality of the k-means method. To do so, consider a clustering problem over real line with five clusters with centers $\mathcal{C} = \{c_1, c_2, \ldots, c_5\}$. For the simplicity of exposition we assume that $c_1 < \cdots < c_5$ and every two consecutive centers are located in a distance of Δ. We assume that there is a ball of radius δ around each center and n data points are distributed in these balls uniformly at random. Hence for the optimal clustering of these n points, the sum of squared Euclidean distance of points to cluster centers is $O(n\delta^2)$, because the distance of each point to its cluster center is at most δ.

We utilize the k-means algorithm to cluster these points. To this end, we initialize the k-means algorithm by choosing five data points at random as the initial centers of the clusters. There is a chance that no data point from first cluster, two data points from the third cluster, and on data point from the remaining clusters have been chosen as centers. In the first round of k-means, all points in clusters 1 and 2 will be assigned to the cluster centered at c_1. The two centers in cluster 3 will end up sharing that cluster. And the centers in clusters 4 and 5 will move roughly to the centers of those clusters. Thereafter, no further changes will occur. This local optimum has cost $\Omega(n\Delta^2)$. We note that this cost can be made arbitrarily far away from the optimum cost by setting the distance between the consecutive centers, i.e., Δ, large enough. As this example illustrates, despite the convergence of k-means algorithm to local minimum, the initialization of k-means algorithm significantly affects the final result. In this simple one dimensional problem, the only scenario that k-means algorithm would generate reasonable result with a cost close to the optimum cost is the case where in the initialization step we sample on data point from each cluster. It is not hard to see that for large number of data points compared to the number of clusters, i.e., n/K, the probability of this case is inversely proportional to the Stirling number in Eq. (4) and is very small.

Motivated by the above example on the poor performance of the k-means algorithm, we propose four different versions of the hybrid clustering, depending on the stage where we carry out the k-means algorithm. The hybrid clustering approaches use k-means algorithm

Table 5
The configuration of parameters for different baseline algorithms which are compared to the proposed algorithms.

GA		ACO		PSO		CABC		HSCLUST	
Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
Population	50	Ants (R)	50	Population	100	Colony size	100	HMS	$2 \times K$
Crossover rate	0.8	Probability for max trial	0.98	Min and max inertia	0.7, 0.9	Upper bounce	5	HMCR	0.9
Mutation rate	0.001	Local search probability	0.01	Acceleration factor (c_1)	2	Limit	100	PAR_{min}	0.09
Max number of iterations	1000	Evaporation rate	0.01	Acceleration factor (c_2)	2	Maximum cycle number	1000	PAR_{max}	0.99
		Max number of iterations	1000	Max number of iterations	1000			Max Number of Iterations	1000
				V_{min}	-0.05				
				V_{max}	0.05				

Table 6
SICD comparison. Hybrid I and Hybrid II indicate the one step hybridization and the interleaved hybridization of IBCOCLUST and *k*-means algorithms, respectively.

Data	Measure	GA	ACO	*k*-means	PSO	CABC	IBCOCLUST	IBKCLUST	KIBCLUST	Hybrid I	Hybrid II
Iris	Average	139.98	97.17	106.05	103.51	—	97.27	97.33	96.4	96.38	95.14
	Best	125.19	97.1	97.33	96.66	—	97.22	97.33	96.4	96.33	95.10
Wine	Average	16,530.53	16,530.53	18,161	16,311	16,449.8	16,460.55	16,460.9	16,460.6	16,458.1	16295.9
	Best	16,530.50	16,530.50	16,555.68	16,294.00	16,433.37	16,460	16,460.55	16,453.28	16,433.37	16,295
Glass	Average	—	—	260.4	291.33	223.68	225.19	223.4	226.34	226.59	221.5
	Best	—	—	215.68	271.29	212.32	214.85	214.78	217.97	214.78	214.71
Cancer	Average	—	—	2988.3	3334.6	2964.4	2976.89	2976.33	2980.15	2977.59	2976.24
	Best	—	—	2987	2976.3	2964.4	2976.24	2976.06	2980.15	2976.24	2976.11
Vowel	Average	—	—	159,242.87	168,477	—	150,881.16	152,575.13	150,751.38	151,688.49	150,892.17
	Best	—	—	149,422.3	163,882	—	149,466.61	149,466.61	150,469.89	149,490.88	149,473.9

Table 7
CEP comparison. Hybrid I and Hybrid II indicate the one step hybridization and the interleaved hybridization of IBCOCLUST and *k*-means algorithms, respectively.

Algorithm	Iris	Wine	Glass	Cancer
VFI	0	5.77	41.11	7.34
Ridor	0.52	5.1	31.66	6.33
NBTree	2.63	2.22	24.07	7.69
MultiBoost	2.63	17.77	53.7	5.59
Bagging	0.26	2.66	25.36	4.47
Kstar	0.52	3.99	17.58	2.44
RBF	9.99	2.88	44.44	20.27
MlpAnn	0	1.33	28.51	2.93
BayesNet	2.63	0	29.62	4.19
PSO	2.63	2.22	39.5	5.8
ABC	0	0	41.5	2.81
IBCOCLUST	2.63	5.5	32.19	4.49
IBKCLUST	2.63	5.2	30.93	4.49
KIBCLUST	2.63	5.5	28.12	4.49
Hybrid I	2.63	4.31	28.12	3.7 5
Hybrid II	2.63	3.96	26.76	3.7

to replace the refining stage in the IBCOCLUST algorithm. The hybrid algorithms combine the power of the IBCOCLUST algorithm with the speed of a *k*-means and the global searching stage and the local refine stage are accomplished by those two modules, respectively. The IBCOCLUST finds the region of the optimum, and then the *k*-means takes over to find the optimum centroids. We need to find the right balance between local exploitation and global exploration.

4.3.1. The sequential hybridization

Two kinds of hybridization of *k*-means and IBCOCLUST are proposed in this section. One of them applies *k*-means and then IBCOCLUST (improved bee colony + *k*-means clustering) whereas the other one changes the order of these algorithms (*k*-means + improved bee colony clustering). We need to find the right balance between local exploitation and global exploration. The global searching stage and the local refine stage are accomplished by those two modules, respectively.

IBKClust: Hybridization of Improved Bee colony and *k*-means Clustering. In the initial stage, the IBCOCLUST module is executed for a short period (50–100 iterations) to discover the vicinity of the optimal solution by a global search and at the same time to avoid consuming high computation. When the IBCOCLUST is completed or shows a negligible trend of improvement after many iterations, the result from the IBCOCLUST module is used as the initial seed of the *k*-means module. The *k*-means algorithm will be applied for refining and generating the final result.

KIBClust: hybridization of *k*-means and improved bee colony clustering. First *k*-means module is performed on the data, and then the results of *k*-means are given to the IBCOCLUST as the initial answers and the IBCOCLUST will continue its process.

4.3.2. The interleaved hybridization

In this hybrid algorithm, the local method is integrated into the IBCOCLUST. For instance, after every *L* iterations, the *k*-means uses the best vector from the IBCOCLUST as its starting point. Then center of clusters is updated if the locally optimized vectors (obtained by *k*-means) have better fitness value than those in IBCOCLUST and this procedure repeated until stop condition.

4.3.3. The integrated hybridization

To improve the algorithm a one-step *k*-means algorithm is introduced. In each iteration, after that a new clustering solution

Table 8
The quality of different hybrid clustering algorithms on five data sets measured in terms of SICD.

Data	Measure	K-NM-PSO	K-PSO	K-GA	K-HS	K-ABC	IBCOCLUST Hybrid (II)
Iris	Average	96.67	96.76	97.10	96.22	96.29	95.14
	Best	96.66	96.66	96.10	96.10	96.19	95.10
Wine	Average	16,293.00	16,296.00	16,298.70	16,296.10	16,296.10	16,294.90
	Best	16,292.00	16,292.00	16,295.00	16,292.20	16,292.50	16,292.00
Glass	Average	200.50	221.55	221.70	221.76	221.89	221.35
	Best	199.68	213.37	215.70	214.80	215.30	214.71
Cancer	Average	2964.70	2965.80	2968.00	2975.30	2970.45	2967.24
	Best	2964.50	2964.50	2965.56	2964.50	2964.60	2963.11
Vowel	Average	150,895.61	150,990.65	150,992.45	150,908.56	150,903.45	150,892.17
	Best	149,496.40	149,486.34	149,556.01	149,478.45	149,498.4	149,473.90

Table 9
The statistics of document data sets used in our experiments.

Dataset	Label	Description	# Documents (n)	# of Clusters (K)
Politics	Polotics	Random topics of politics	176	6
TREC	Newspaper	Various articles from certain topics	873	8
DMOZ	DMOZ	Selected documents among 14 topics	697	14
20 Newsgroup	Message	Collected from 10 different usenet newsgroups	9249	10
WebAce	Wap	Web pages listed in the subject hierarchy of Yahoo!	1560	20

is generated with applying the IBCOCLUST, we use *k*-means to reassign each data to the cluster with the nearest centroid. If the result of *k*-means has better fitness than the generated solution by the IBCOCLUST, then we replace it with a candidate solution of the IBCCOCLUST. In each iteration, one iteration of the IBCOCLUST and then one iteration of *k*-means are performed, and the process will continue until the fixed number of iterations has finished.

5. Experimental results on basic data sets

To better understand the performance of the proposed algorithms, we divide the conducted experiments into two different settings. In the first setting, we apply the proposed algorithms to few well-known general purpose benchmark data sets and compare them with the baseline algorithms. In the second setting, we compare the proposed algorithms to the state-of-the-art algorithms on document clustering. The main reason for splitting our experiments into two parts is the unique challenges that exist in clustering document data sets due to the high-dimensionality and spareness characteristics of text data which makes it much challenging problem. We note that different algorithms might achieve totally different performance on these two types of data sets. The focus of this section is on general purpose data sets. The application of proposed algorithms to five different document data sets will be discussed in Section 6.

We begin by introducing the data sets we have used in our experiments followed by comparing the proposed algorithms to a number of well-known algorithms according to their quality and rate of convergence. We also compare the proposed hybrid algorithms to other hybrid evolutionary methods proposed in the literature.

5.1. Data sets

In this work, five benchmark data sets from the UC Irvine (UCI) machine learning repository [94] which is a well-known database repository, are used to evaluate the performance of the proposed BCO based algorithms. The data sets and their main statistics are summarized in Table 2.

5.2. Experimental setup

In the next step, the proposed BCO based algorithms are applied to the data sets summarized in Table 2. The Euclidean distance measure is used as the similarity metric in each algorithm. It should be emphasized at this point that the results shown in the rest of the paper for every dataset, are the average of over 30 independent runs of the algorithms (to make a fair comparison), each run with randomly generated initial solutions and different seeds of the random number generator. Also, for an easy comparison, the algorithms proceed over 1000 iterations in each run since the 1000 generations are enough for convergence of the algorithms. No parameter needs to be set up for the *k*-means algorithm. For each data set optimum number of bees is decided empirically which will be elaborated in the next subsection. We would like to emphasize that our experimental results revealed that the optimum number of bees is a factor of the number of features in the data set.

5.3. Empirical study of the impact of different parameters on convergence of IBCOCLUST

The aim of this section is to study the effect of two important parameters, namely the number of bees and the parameter γ on the result attained by different proposed algorithms. The SICD value of the obtained solution is the value of fitness function. The algorithm we use to evaluate is the IBCOCLUST which was described in Section 4.2.

5.3.1. The impact of number of bees on clustering quality

Figs. 2 and 4 report the fitness of solutions measured by SICD for different values of the number of bees. We can see that decreasing the number of bees leads to premature convergence and increasing

Table 10
Normalized SICD comparison. Hybrid I and Hybrid II indicate the one step hybridization and the interleaved hybridization of IBCOCLUST and *k*-means algorithms, respectively.

Data	Criteria	*k*-means	HSCLUST	IBCOCLUST	IBKCLUST	KIBCLUST	Hybrid I	Hybrid II
Politics	Euclidean	0.73254	0.6524	0.6043	0.6174	0.6162	0.597	0.5821
	Cosine	0.7690	0.6732	0.6235	0.6246	0.6419	0.6382	0.601
Newspaper	Euclidean	0.6815	0.3682	0.5445	0.5729	0.6323	0.5684	0.492
Cosine		0.7263	0.6723	0.6716	0.6392	0.5978	0.6093	0.5744
DMOZ	Euclidean	0.4587	0.3952	0.2947	0.4543	0.4593	0.3758	0.3669
Cosine		0.4821	0.4092	0.3374	0.3419	0.3563	0.3655	0.3367
Message	Euclidean	0.8325	0.7612	0.7423	0.7896	0.7747	0.7739	0.7511
Cosine		0.9206	0.8843	0.8958	0.8737	0.884	0.8624	0.8759
Wap	Euclidean	0.8652	0.8147	0.8234	0.7951	0.8103	0.7968	0.7893
Cosine		0.8753	0.82	0.8433	0.7814	0.7882	0.7695	0.7532

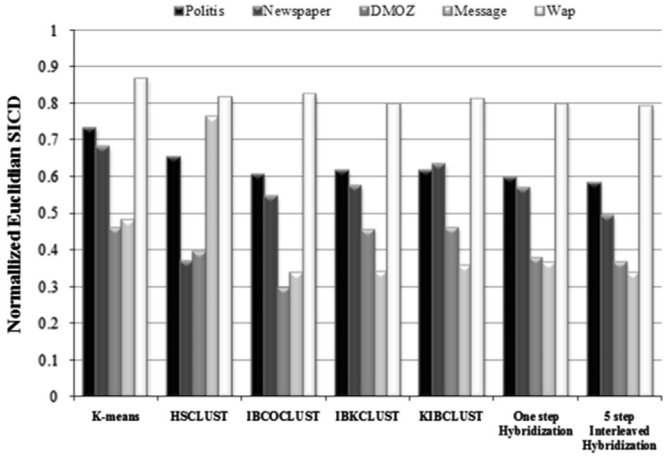

Fig. 5. Normalized Euclidean SICD comparison.

Table 11
Comparison of proposed hybrid algorithms to other baseline algorithms in document clustering measured in terms of F-measure quality.

Dataset	Politics	Newspaper	DMOZ	Message	Wap
k-means	0.6035	0.5117	0.5423	0.3236	0.4302
GA	0.6822	0.6427	0.7194	0.5213	0.4982
HSCLUST	0.7655	0.7824	0.72445	0.6692	0.58963
IBCOCLUST	0.8317	0.7256	0.7668	0.6845	0.63291
IBKCLUST	0.8152	0.8324	0.7731	0.75993	0.6022
KIBCLUST	0.7916	0.8212	0.7563	0.6954	0.7225
Hybrid I	0.7928	0.8349	0.7433	0.7166	0.7098
Hybrid II	0.8846	0.8826	0.8574	0.7902	0.7342

the number of bees leads to significant improvements in the initial phase of the clustering. Note that when the time or the number of iterations is fixed as shown in Figs. 2 and 4, increasing the number of bees may deteriorate the quality of the clustering. In general we can say, the larger number of bees, the more time (or iterations) is needed for algorithm to find the optimal solution but usually higher quality is achieved.

As it can be seen in Figs. 2 and 4, a very small number of bees lead to less exploration in the search space this potentially leads to the algorithm to get stuck in a local optimum. On the other hand, as the number of iterations is finite, increasing the number of bees may deteriorate the quality of the clustering. In general, using a mild number of bees seems to be a good and logical choice with the advantages of converging to the best result. In addition, our empirical studies demonstrate that with a linear relation between number of bees and the number of feature, better results are reached.

To decide the best setting for the number of bees, the value of the parameter γ and the number of iterations are set fixed and the SICD of IBCOCLUST for different values of number of bees is evaluated. Table 3 shows the effect of increasing the number of bees, when the number of generations is constant with the value of 1000. By producing high number of bees, it will be guaranteed that, most of the possible solutions that are available in the solution space will be searched. As it can be inferred from Table 3, by increasing the number of bees, better clustering results can be acquired, but it has some fluctuations. Its worst value is gained when number of bees is 2, while the best value was achieved when the number of bees was linear function of the number of feature in each cluster. Table 3 indicates that, when the amount of bees are few, they seek the solution space in low depth, while by increasing their number solution space will be explored in more depth.

5.3.2. *The impact of parameter γ on clustering quality*

We repeat the same process to decide the value of the parameter γ in our experiments. Similar to the experiments conducted for the number of bees, we fix the value of other parameters, i.e., the number of bees and the maximum number of iterations, and vary the value of parameter γ. The results of this experiments of four data sets are reported in Table 4. From the results in Table 4 few conclusions are in order. First, we note that the value of parameter γ should be tuned based on the data set at hand to obtain the best results. Second, we can observe that for two Wine and Glass data sets, the value of γ is much smaller than the best setting of this parameter for other two data sets. Comparing Wine and Glass data sets to Iris and Vowel data sets in terms of number of features, this fact demonstrates that for data sets with a large number of features the value of γ must be chosen smaller. Finally, based on the results in Table 4, it seems reasonable to do a model checking on this parameter by performing a grid search on the candidate values of γ in the $(0, 1]$ interval with the scales of 0.01 to find the best possible value.

5.4. *Convergence analysis*

Here we present experiments to investigate the effectiveness of the proposed algorithms in tens of their convergence rate to the optimal solution.

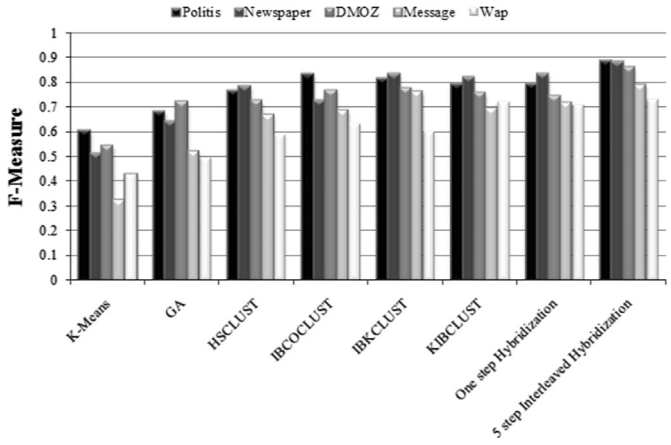

Fig. 6. F-measure comparison.

Fig. 3 illustrates the convergence behavior of the proposed algorithms and the *k*-means algorithm on the Glass data set with the number of bees $B=9$ and $\gamma=0.01$. Fig. 3 illustrates that the reduction of SICD value in IBCOCLUST follows a smooth curve from its initial vectors to final optimum solution with no sharp moves. Another noteworthy point in Fig. 3 is that SICD has the lowest final value for L-step Interleaved Hybridization among the other algorithms. The sequence of other algorithms with respect to their SICD values are Interleaved Hybridization, Sequential Hybridization and IBCOCLUST. It can be inferred from Fig. 3 that hybrid algorithms overcome IBCOCLUST disadvantage by incorporating two-step hybrid algorithms. The algorithm uses BCO to get close to optimal solution, but since it does not fine-tune this result, it uses the *k*-means algorithm to fine tunes that. The results show that the hybrid approaches outperform the component algorithms (*k*-means and IBCOCLUST) in terms of the quality of generated clusters. As it can be seen from Fig. 3 the IBCOCLUST takes more time to reach the optimal solution than the *k*-means. This is because the *k*-means algorithm may be trapped in local optimums. Although the *k*-means algorithm is more efficient than the IBCOCLUST with respect to execution time, the IBCOCLUST generates much better clustering than the *k*-means algorithm.

5.5. Comparison to other baseline algorithms

In this part of experiments, we evaluate and compare the performances of the proposed algorithms according to their quality of generated clusters with *k*-mean, PSO based clustering (PSO) [43], GA based clustering (GA) [67], ACO based clustering (ACO) [79] and cooperative artificial bee colony based clustering (CABC) [97] algorithms. The algorithmic parameters used in this set of experiments for each baseline algorithm is reported in Table 5. The setting of parameters for ACO, GA, PSO and CABC is the same as their original paper.

To evaluate the quality of clustering obtained by different algorithms, we use two metrics, namely Classification Error Percentage (CEP) and SICD where the first measure has been chosen from external quality measures and SICD has been selected from internal measures. CEP expresses the clustering results from an external export view as shown in Eq. (14), whereas SICD examines how much the clustering satisfies the optimization constraints

$$CEP = \frac{\text{of misclassified objects}}{\text{size of test dataset}} \times 100 \tag{14}$$

We now report and discuss the results for each measure separately in the following subsections.

5.5.1. SICD based evaluation

Table 6 reports the SICD value of algorithms applied to the mentioned data sets. The smaller the SICD value, the more compact the clustering solution is. Looking at Table 6, we can see that the results obtained by different proposed algorithms are comparable. It is noticeable from Table 6 that the PSO algorithm outperforms the proposed algorithms in Wine dataset but in other datasets the proposed algorithms have better SICD than PSO and other well-known algorithms. In comparing the proposed algorithms to each other we can say that the 5 step Interleaved Hybridization outperforms the others.

5.5.2. CEP based evaluation

In order to make a better evaluation of clustering, as a primary measure of quality, we used the widely adopted CEP measure. Since the benchmark data sets have their nominal partitions known to the user, we also compute the mean number of misclassified data points. This is the average number of objects that were assigned to clusters other than according to the nominal classification. In Table 7, we report the misclassification errors (with respect to the nominal classification) for the experiments conducted for different algorithms.

5.6. Comparison to other hybrid clustering methods

In addition to the basic evolutionary-based clustering algorithms, we compare the proposed algorithms to other state-of-th-art hybrid algorithms. The hybrid models use both evolutionary based algorithms and the *k*-means algorithm simultaneously. These algorithms include a hybrid technique based on combining the *k*-means algorithm, NelderMead simplex search, and particle swarm optimization (K-NM-PSO) [22], a hybrid technique of *k*-means and particle swarm optimization (K-PSO) [1], a hybrid approach based on genetic algorithm and *k*-means algorithm called K-GA [45], harmony *k*-means algorithm K-HS [60] and hybrid algorithm for data clustering using ABC and *k*-means algorithm, dubbed K-ABC [46]. A brief description of these algorithm is given below for completeness:

1. *K-PSO* [1]: The Hybrid PSO algorithm, first uses the *k*-means clustering to seed the initial swarm, and then uses PSO algorithm to refine the clusters formed by *k*-means. In this approach, *k*-means is used to calculate the distance from each item to the cluster centers. In this approach, the result of the *k*-means algorithm is used as one of the particles, while the remaining particles are initialized randomly.
2. *K-NM-PSO* [22]: This algorithm is a hybrid technique based on combining the *k*-means algorithm, NelderMead simplex search, and particle swarm optimization. It clusters arbitrary data by evolving the appropriate cluster centers in an attempt to optimize a given clustering metric. The hybrid algorithm first executes the *k*-means algorithm, which terminates when there is no change in centroid vectors. This algorithm randomly generate 3 N particles, or vertices, and NM-PSO is then carried out to its completion.
3. *K-HS* [60]: The hybrid K-HS algorithm combines the harmony search algorithm with a one-step *k*-means algorithm. Each row of the harmony memory in this algorithm has a discrete representation. This algorithm codifies the whole partition of the data in a vector of length *n*, where *n* is the number of data. In this algorithm at each improvisation step a one-step *k*-means is included to fine-tune the new solution.
4. *K-GA* [45]: This algorithm is a hybrid approach based on genetic and *k*-means algorithms called the genetic *k*-means algorithm for clustering analysis which defines a basic mutation operator specific to clustering called distance-based mutation. The genetic

operators that are used in K-GA are the selection, the distance based mutation and the k-means operator. The representation of GA is to consider a chromosome of length n and allow each allele in chromosome to take values from $\{1, 2, \ldots, K\}$. The mutation changes an allele value depending on the distances of the cluster centroids from the corresponding data point.

In Table 8, SICD values of hybrid version of IBCOCLUST is compared to other hybrid evolutionary methods proposed in the literature. In Iris, Wine and Vowel data sets, the proposed algorithm had the lowest rates of SICD that makes the algorithm as a distinct one, while in other datasets such as Glass and Cancer proposed algorithm had a similar or superior performance comparing to other competitors. The behavior of the proposed algorithms is varied in different datasets, but what is very common among most of the datasets is the superiority of the proposed algorithm, over the conventional variations, in most datasets.

6. Experimental results on document clustering

We now turn to comparing the proposed algorithms to the state-of-the-art algorithms on document clustering. Document clustering is a crucial and important application in Information Retrieval [25] and characteristics of this type of data such as high-dimensionality and sparseness introduces new challenges to clustering problem and makes it harder compared to other types of data. Having this in mind, we chose this application for evaluating and comparing the proposed algorithms on different document data sets. In this section, a brief introduction to the problem of document clustering is given, the data sets are introduced and the algorithms are compared with k-means, a GA based algorithm, and harmony search based document clustering (referred to as HSCLUST) [25].

6.1. Document clustering

In document clustering the vector space model is used to represent documents in which each vector is set of the documents features such as words, terms and N-grams. These vectors are used in similarity measure between documents as well. We use the same notation as Section 3, where in document clustering $\mathcal{D}$ would be the set of n documents in which $d_i, i = 1, 2 \ldots, d$ is the ith documents. These real values can be determined as the words frequency or can have other relevant measures like frequency and inverse document frequency (TF-IDF) which is the most widely used weighting schema [75]. Having in mind that assuming all of the words in a document will result in a very high vector dimension (i.e., large d), a preprocessing phase is applied to eliminate the unnecessary words and reduce the vector dimension [44]. The similarity measure is the two well-known similarities namely Euclidean and Cosine measures which are introduced in Section 3 in Eqs. (1) and (2). Also the performance of a clustering is measured, using the introduced SICD fitness function as defined in Eq. (3).

6.2. Document data sets

For the application of document clustering five different datasets with different characteristics are used. The first dataset namely Politics is a dataset consisting of random topics in politics which is collected in 2006. The TRED dataset is collected among different topics from San Jose Mercury newspaper including topics such as computers, electronics, health, medical, research, and technology. The DMOZ dataset is collected among 14 topics in which for each topic some web pages are selected and included in the data. The 20 Newsgroup dataset is collection of 1000 messages from each of the 10 different Usenet newsgroups resulting in 10,000 messages which become 9249 after preprocessing. The last dataset WebAce is from the WebACE project (WAP) [9,66]. The details about these datasets are demonstrated in Table 9.

The feature vector of each document is its words, however assuming all of the words in a document makes the feature set too much big for text mining. Therefore, to overcome this problem a preprocessing approach is necessary to reduce the dimension of the feature set. To this aim, the common words (e.g. function words: “a”, “the”, “in”, “to”; pronouns: “I”, “he”, “she”, “it”) are eliminated from the documents, also different forms of a stem are determined as one.

6.3. Experimental setup

The entire proposed algorithms are applied to the introduced datasets. The parameters are tuned as mentioned in Section 5.2. The results shown in the rest of paper are the average over 30 runs of the algorithms. Based on the achievements in Section 5.3 the number of bees is indicated as a factor of the number of features for each dataset.

6.4. Quality of clustering

In this part of experiments we compare the proposed algorithms according to their quality of generated clusters with few well known and efficient clustering algorithms including k-mean, harmony search and GA based clustering algorithms [25,4].

For evaluation of clustering quality we used the two greatly applicable metrics of SICD and F-measure. The SICD metric measures the external quality while the F-measure examines the internal quality of the clustering.

6.4.1. SICD based evaluation

Table 10 demonstrates the comparison of normalized SICD for five datasets using both cosine and Euclidean similarity measures. As it can be seen the result obtained by our proposed algorithms outperforms k-means in all datasets and in average 5 step Interleaved Hybridization outperform all the algorithms proposed. Fig. 5 depicted this comparison for Euclidian similarity measure.

6.4.2. F-measure based evaluation

Another evaluation metric we utilize to compare the quality of clustering resulted in our proposed algorithms is F-measure [5]. It is defined as the harmonic means of precision and recall from information retrieval. In our measurement each cluster is supposed as if it were the result of a query and each class as if it were the preferred set of documents for a query. The recall and precision of that cluster for each specified class can then be calculated. For a specified cluster of documents $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K\}$, to assess the quality of $\mathcal{C}$ with respect to an ideal cluster $\mathcal{C}_* = \{\mathbf{c}_1^*, \mathbf{c}_2^*, \ldots, \mathbf{c}_K^*\}$ (categorization by human) we first compute precision and recall as

$$P(\mathcal{C}, \mathcal{C}_*) = \frac{|\mathcal{C} \cap \mathcal{C}_*|}{|\mathcal{C}|} \quad \text{and} \quad R(\mathcal{C}, \mathcal{C}_*) = \frac{|\mathcal{C} \cap \mathcal{C}_*|}{|\mathcal{C}|} \tag{15}$$

Then we define:

$$F(\mathcal{C}, \mathcal{C}_*) = \frac{2 \times P(\mathcal{C}, \mathcal{C}_*) \times R(\mathcal{C}, \mathcal{C}_*)}{P(\mathcal{C}, \mathcal{C}_*) + R(\mathcal{C}, \mathcal{C}_*)} \tag{16}$$

Table 11 shows the details of F-measures compared to k-means and GA. Again the 5 step Interleaved Hybridization has the best performance according to F-measure and all other algorithms outperform k-means and GA as well. Fig. 6 shows a better overview on this comparison.

7. Conclusion

In this paper, we applied the bee colony optimization (BCO) algorithm to the clustering problem. The shortcomings of basic pure BCO based clustering were examined and refined in IBCOCLUST which is the improved version of the basic algorithm. In particular, the improved algorithm is a novel modification of the BCO optimization algorithm by introducing the fairness and cloning properties which are aimed at increasing the explorative power of the BCO algorithm and propagation of knowledge in an optimization process, respectively.

Additionally, four different hybridization methods have been proposed as well, which are basically the composition of IBCOCLUST and the *k*-means algorithms in different manners: (1) IBKCLUST in which the IBCOCLUST is applied before the *k*-means, (2) KIBCLUST in which *k*-means is applied before the IBCOCLUST, (3) One step Hybridization IBCOCLUST+*k*-means in which at each iteration both algorithms are applied simultaneously, and (4) *k* step Interleaved Hybridization in which in each iteration *k* steps of each algorithm is applied simultaneously. The performances of all of the proposed algorithms are compared with well-known methods which are widely used by the researchers in two applications of data and document clustering. The results of the experiments show an impressive improvement in comparison to others and can indicate that the Improved Bee Colony algorithm can successfully be applied to clustering for the purpose of clustering. It also shows that the *k* step Interleaved Hybridization method results in best solution, since in each iteration the steps of IBCOCLUST augment the search space by searching more globally, and after that the *k*-means algorithm has another *k* steps chance to find the locally optimum in the global solution area that IBCOCLUST provided for it. As this process goes on, the algorithm has a chance to investigate many local optima in many global solution areas that are the best global solution areas among other areas and therefore gain the best performance.

This work leaves few directions, both theoretically and empirically, as future work. In our setting, the number of clusters and the number of data points were assumed to be fixed in advance and as a result, a static matrix for the assignment of data points to the clusters was sufficient for our optimization purpose. When the number of clusters is not known or the data points can be dynamically added or removed, this static structure would not be sufficient and a dynamic data structure is necessary. We note that considering the hardness of the clustering problem even for a fixed number of clusters and data points, the dynamic problem is much more challenging and requires careful investigating. It would be interesting to further examine this issue as a future work.

Acknowledgment

The authors would like to thank the Associate Editor and anonymous reviewers for their immensely insightful comments and helpful suggestions on the original version of this paper.

References

[1] Alireza Ahmadyfard, Hamidreza Modares, Combining pso and k-means to enhance data clustering, in: International Symposium on Telecommunications, IST 2008, IEEE, 2008, pp. 688–691.
[2] Shafiq Alam, Gillian Dobbie, Patricia Riddle, Particle swarm optimization based clustering of web usage data, in: Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, vol. 03, IEEE Computer Society, 2008, pp. 451–454.
[3] Daniel Aloise, Amit Deshpande, Pierre Hansen, Preyas Popat, Np-hardness of euclidean sum-of-squares clustering, Mach. Learn. 75 (2) (2009) 245–248.
[4] Sanghamitra Bandyopadhyay, Ujjwal Maulik, An evolutionary technique based on k-means algorithm for optimal clustering in rn, Inf. Sci. 146 (1) (2002) 221–237.
[5] Arindam Banerjee, Chase Krumpelman, Joydeep Ghosh, Sugato Basu, Raymond J. Mooney, Model-based overlapping clustering, in: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, ACM, Chicago, IL, USA, 2005, pp. 532–537.
[6] Pevel Berkhin, A survey of clustering data mining techniques, 2006, pp. 25–71.
[7] James C. Bezdek, Srinivas Boggavarapu, Lawrence O. Hall, Amine Bensaid, Genetic algorithm guided clustering, in: Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, IEEE, 1994, pp. 34–39.
[8] Salim Bitam, Mohamed Batouche, E.-g. Talbi, A survey on bee colony algorithms, in: IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), IEEE, 2010, pp. 1–8.
[9] Daniel Boley, Maria Gini, Robert Gross, Eui-Hong Sam Han, Kyle Hastings, George Karypis, Vipin Kumar, Bamshad Mobasher, Jerome Moore, Document categorization and query generation on the world wide web using webace, Artif. Intell. Rev. 13 (5–6) (1999) 365–391.
[10] Paul S. Bradley, Usama Fayyad, Cory Reina, Scaling em (expectation–maximization) clustering to large databases, Microsoft Research, 1998.
[11] Arantza Casillas, M.T. González De Lena, R. Martínez, Document clustering into an unknown number of clusters using a genetic algorithm, in: Text, Speech and Dialogue, Springer, České Budéjovice, Czech Republic, 2003, pp. 43–49.
[12] Sheng-Chai Chi, Chih Chieh Yang, Integration of ant colony som and k-means for clustering analysis, in: Knowledge-Based Intelligent Information and Engineering Systems, Springer, Bournemouth, UK, 2006, pp. 1–8.
[13] Maurice Clerc, James Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (1) (2002) 58–73.
[14] Sandra C.M. Cohen, Leandro N. de Castro, Data clustering with particle swarms, in: CEC 2006, IEEE Congress on Evolutionary Computation, IEEE, 2006, pp. 1792–1798.
[15] Dipankar Dasgupta, Advances in artificial immune systems, Computational intelligence magazine, IEEE 1 (4) (2006) 40–49.
[16] Sanjoy Dasgupta, The Hardness of k-Means Clustering, Department of Computer Science and Engineering, University of California, San Diego, 2008.
[17] Sanjoy Dasgupta, Yoav Freund, Random projection trees for vector quantization, IEEE Trans. Inf. Theory 55 (7) (2009) 3229–3242.
[18] Ivanoe De Falco, Antonio Della Cioppa, Ernesto Tarantino, Facing classification problems with particle swarm optimization, Appl. Soft Comput. 7 (3) (2007) 652–658.
[19] Marco Dorigo, Vittorio Maniezzo, Alberto Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 26 (1) (1996) 29–41.
[20] Piotr Dziwiński, Łukasz Bartczuk, Janusz T. Starczewski, Fully controllable ant colony system for text data clustering, in: Swarm and Evolutionary Computation, Springer, Zakopane, Poland, 2012, pp. 199–205.
[21] Emanuel. Falkenauer, Genetic Algorithms and Grouping Problems, John Wiley & Sons, Inc., 1998.
[22] Shu-Kai S. Fan, Yun-chia Liang, Erwie Zahara, Hybrid simplex search and particle swarm optimization for the global optimization of multimodal functions, Eng. Optim. 36 (4) (2004) 401–418.
[23] Mohammad Fathian, Babak Amiri, Ali Maroosi, Application of honey-bee mating optimization algorithm on clustering, Appl. Math. Comput. 190 (2) (2007) 1502–1513.
[24] Edward W. Forgy, Cluster analysis of multivariate data: efficiency versus interpretability of classifications, Biometrics 21 (1965) 768–769.
[25] Forsati Rana, Mahdavi Mehrdad, Shamsfard Mehrnoush, Meybodi Mohammad Reza, Efficient stochastic algorithms for document clustering, Inf. Sci. 220 (2013) 269–291.
[26] Rana Forsati, MohammadReza Meybodi, Mehrdad Mahdavi, AzadehGhari Neiat, Hybridization of k-means and harmony search methods for web page clustering, in: Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, vol. 01, IEEE Computer Society, Sydney, Australia, 2008, pp. 329–335.
[27] Zong Woo Geem, Joong Hoon Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2) (2001) 60–68.
[28] David Edward Goldberg, et al., Genetic Algorithms in Search, Optimization, and Machine Learning, vol. 412, Addison-wesley, Reading Menlo Park, 1989.
[29] Julie Greensmith, Uwe Aickelin, The dendritic cell algorithm (Ph.D. thesis), Nottingham Trent University, 2007.
[30] Sudipto Guha, Rajeev Rastogi, Kyuseok Shim, Cure: an efficient clustering algorithm for large databases, 27(2) (1998) 73–84.
[31] Zülal Güngör, Alper Ünler, *k*-harmonic means data clustering with simulated annealing heuristic, Appl. Math. Comput. 184 (2) (2007) 199–209.
[32] Julia Handl, Bernd Meyer, Improved ant-based clustering and sorting in a document retrieval interface, in: Parallel Problem Solving from Nature PPSN VII, Springer, 2002, Granada, Spain, pp. 913–923.
[33] Hong He, Yonghong Tan, A two-stage genetic algorithm for automatic clustering, Neurocomputing 81 (2012) 49–59.
[34] Fu Hui, Chen Chao-tian, Liu Xiao-Yong, An improvement text clustering algorithm based on ant colony, in: 1st International Conference on Information Science and Engineering (ICISE), IEEE, 2009, pp. 782–785.
[35] Anil K. Jain, Data clustering: 50 years beyond k-means, Pattern Recognit. Lett. 31 (8) (2010) 651–666.
[36] Anil K. Jain, M. Narasimha Murty, Patrick J. Flynn, Data clustering: a review, ACM Comput. Surv. (CSUR) 31 (3) (1999) 264–323.

[37] Hua Jiang, Jing Li, Shenghe Yi, Xiangyang Wang, Xin Hu, A new hybrid method based on partitioning-based dbscan and ant clustering, Expert Syst. Appl. 38 (8) (2011) 9373–9381.
[38] Hua Jiang, Shenghe Yi, Jing Li, Fengqin Yang, Xin Hu, Ant clustering algorithm with *k*-harmonic means clustering, Expert Syst. Appl. 37 (12) (2010) 8679–8684.
[39] Yucheng Kao, Kevin Cheng, An aco-based clustering algorithm, in: Ant Colony Optimization and Swarm Intelligence, Springer, Brussels, Belgium, 2006, pp. 340–347.
[40] Dervis Karaboga, Celal Ozturk, A novel clustering approach: artificial bee colony (abc) algorithm, Appl. Soft Comput. 11 (1) (2011) 652–657.
[41] George Karypis, Eui-Hong Han, Vipin Kumar, Chameleon: hierarchical clustering using dynamic modeling, Computer 32 (8) (1999) 68–75.
[42] Michael Steinbach George Karypis, Vipin Kumar, A comparison of document clustering techniques, in: TextMining Workshop at KDD2000 (May 2000), 2000.
[43] James Kennedy, Particle swarm optimization, in: Encyclopedia of Machine Learning, Springer, 2010, pp. 760–766.
[44] James F. Kennedy, James Kennedy, Russel C. Eberhart, Swarm Intelligence, Morgan Kaufmann, 2001.
[45] K. Krishna, M. Narasimha Murty, Genetic k-means algorithm, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 29 (3) (1999) 433–439.
[46] M. Krishnamoorthi, A.M. Natarajan, Abk-means: an algorithm for data clustering using abc and k-means algorithm, Int. J. Comput. Sci. Eng. 8 (4) (2013) 383–391.
[47] Ravindra Krovi, Genetic algorithms for clustering: a preliminary investigation, in: Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences, vol. 4, IEEE, 1992, pp. 540–544.
[48] R.J. Kuo, H.S. Wang, Tung.-Lai. Hu, S.H. Chou, Application of ant k-means on clustering analysis, Comput. Math. Appl. 50 (10) (2005) 1709–1724.
[49] Nicolas Labroche, Nicolas Monmarché, Gilles Venturini, Antclust: ant clustering and web usage mining, in: Genetic and Evolutionary ComputationGECCO 2003, Springer, Chicago, IL, 2003, pp. 25–36.
[50] Chaoshun Li, Jianzhong Zhou, Pangao Kou, Jian Xiao, A novel chaotic particle swarm optimization based fuzzy clustering algorithm, Neurocomputing 83 (2012) 98–109.
[51] Shu-Hsien Liao, Chih-Hao Wen, Artificial neural networks classification and clustering of methodologies and applications–literature analysis from 1995 to 2005, Expert Syst. Appl. 32 (1) (2007) 1–11.
[52] Yi Lu, Shiyong Lu, Farshad Fotouhi, Youping Deng, Susan J. Brown, Fgka: a fast genetic k-means clustering algorithm, in: Proceedings of the 2004 ACM symposium on Applied computing, ACM, Nicosia, Cyprus, 2004, pp. 622–623.
[53] Yi Lu, Shiyong Lu, Farshad Fotouhi, Youping Deng, Susan J. Brown, Incremental genetic k-means algorithm and its application in gene expression data analysis, BMC Bioinf. 5 (1) (2004) 172.
[54] Panta Lucic, Dusan Teodorovic, Bee system: modeling combinatorial optimization transportation engineering problems by swarm intelligence, in: Preprints of the TRISTAN IV Triennial Symposium on Transportation Analysis, 2001, pp. 441–445.
[55] Panta Lucic, Dusan Teodorovic, Transportation modeling: an artificial life approach, in: Proceedingsof 14th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2002, IEEE, 2002, pp. 216–223.
[56] Panta Lučić, Dušan Teodorović, Computing with bees: attacking complex transportation engineering problems, Int. J. Artif. Intell. Tools 12 (03) (2003) 375–394.
[57] James MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, California, USA, 1967, p. 14.
[58] Meena Mahajan, Prajakta Nimbhorkar, Kasturi Varadarajan, The planar k-means problem is np-hard, in: WALCOM: Algorithms and Computation, Springer, Kolkata, India, 2009, pp. 274–285.
[59] M. Mahdavi, Mohammad Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Appl. Math. Comput. 188 (2) (2007) 1567–1579.
[60] Mehrdad Mahdavi, Hassan Abolhassani, Harmony k-means algorithm for document clustering, Data Min. Knowl. Discov. 18 (3) (2009) 370–391.
[61] Mehrdad Mahdavi, M. Haghir Chehreghani, Hassan Abolhassani, Rana Forsati, Novel meta-heuristic algorithms for clustering web documents, Appl. Math. Comput. 201 (1) (2008) 441–451.
[62] Jianchang Mao, Anil K. Jain, Artificial neural networks for feature extraction and multivariate data projection, IEEE Trans. Neural Netw. 6 (2) (1995) 296–317.
[63] Ujjwal Maulik, Sanghamitra Bandyopadhyay, Genetic algorithm-based clustering technique, Pattern Recognit. 33 (9) (2000) 1455–1465.
[64] Boris Mirkin, Mathematical Classification and Clustering: From How to What and Why, Springer, 1998.
[65] Tom M. Mitchell, Machine learning and data mining, Commun. ACM 42 (11) (1999) 30–36.
[66] Jerome Moore, Eui-Hong Han, Daniel Boley, Maria Gini, Robert Gross, Kyle Hastings, George Karypis, Vipin Kumar, Bamshad Mobasher, Web page categorization and feature selection using association rule and principal component clustering, IBM shared research report/University of Minnesota (Minneapolis, Mn.), 98:3, 1997.
[67] Chivukula A. Murthy, Nirmalya Chowdhury, In search of optimal clusters using genetic algorithms, Pattern Recognit. Lett. 17 (8) (1996) 825–832.
[68] Clark F. Olson, Parallel algorithms for hierarchical clustering, Parallel Comput. 21 (8) (1995) 1313–1325.
[69] M. Omran, A. Salman, A.P. Engelbrecht, Image classification using particle swarm optimization, in: Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning, vol. 2002, Singapore, 2002. pp. 370–374.
[70] Nikhil R. Pal, James C. Bezdek, E.C.-K. Tsao, Generalized clustering networks and Kohonen's self-organizing scheme, IEEE Trans. Neural Netw. 4 (4) (1993) 549–557.
[71] Sandra Paterlini, Thiemo Krink, Differential evolution and particle swarm optimisation in partitional clustering, Comput. Stat. Data Anal. 50 (5) (2006) 1220–1247.
[72] Sandra Paterlini, Tommaso Minerva, Evolutionary approaches for cluster analysis, in: Soft Computing Applications, Springer, 2003, pp. 165–176.
[73] D.T. Pham, S. Otri, A. Afify, Massudi Mahmuddin, H. Al-Jabbouli, Data clustering using the bees algorithm, 2007.
[74] Gerard Salton, Automatic text analysis, Science 168 (3929) (1970) 335–343.
[75] Gerard Salton, Christopher Buckley, Term-weighting approaches in automatic text retrieval, Inf. Process. Manag. 24 (5) (1988) 513–523.
[76] Manish Sarkar, B. Yegnanarayana, Deepak Khemani, A clustering algorithm using an evolutionary programming-based approach, Pattern Recognit. Lett. 18 (10) (1997) 975–986.
[77] Mohammad Ali Shafia, Mohammad Rahimi Moghaddam, Rozita Tavakolian, A hybrid algorithm for data clustering using honey bee algorithm, genetic algorithm and k-means method, J. Adv. Comput. Sci. Technol. Res. 1 (2) (2011).
[78] Henry. Sharp Jr., Cardinality of finite topologies, J. Comb. Theory 5 (1) (1968) 82–86.
[79] P.S. Shelokar, Valadi K. Jayaraman, Bhaskar D. Kulkarni, An ant colony approach for clustering, Anal. Chim. Acta 509 (2) (2004) 187–195.
[80] Marc Teboulle, A unified continuous optimization framework for center-based clustering methods, J. Mach. Learn. Res. 8 (2007) 65–102.
[81] Dušan. Teodorović, Swarm intelligence systems for transportation engineering: principles and applications, Transp. Res. Part C: Emerg. Technol. 16 (6) (2008) 651–667.
[82] DUŠAN Teodorovic, Tatjana Davidovic, Milica Selmic, Bee colony optimization: the applications survey, ACM Trans. Comput. Logic 1529 (2011) 3785.
[83] Chi-Ho Tsang, Sam Kwong, Ant colony clustering and feature extraction for anomaly intrusion detection, in: Swarm Intelligence in Data Mining, Springer, 2006, pp. 101–123.
[84] Shivakumar Vaithyanathan, Byron Dom, Model selection in unsupervised learning with applications to document clustering, in: Proceedings of the Sixteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., CA, USA, 1999, pp. 433–443.
[85] D.W. Van der Merwe, Andries Petrus Engelbrecht, Data clustering using particle swarm optimization, in: CEC'03, The 2003 Congress on Evolutionary Computation, vol. 1, IEEE, 2003, pp. 215–220.
[86] Miao Wan, Lixiang Li, Jinghua Xiao, Cong Wang, Yixian Yang, Data clustering using bacterial foraging optimization, J. Intell. Inf. Syst. 38 (2) (2012) 321–341.
[87] Shuting Xu, Jun Zhang, A parallel hybrid web document clustering algorithm and its performance study, J. Supercomput. 30 (2) (2004) 117–131.
[88] Xu Xiaohua, Lu Lin, He Ping, Pan Zhoujin, Chen Ling, Improving constrained clustering via swarm intelligence, Neurocomputing 116 (2013) 317–325.
[89] Xiaohui Yan, Yunlong Zhu, Wenping Zou, Liang Wang, A new approach for data clustering using hybrid artificial bee colony algorithm, Neurocomputing 97 (2012) 241–250.
[90] Yan Yang, Mohamed Kamel, Fan Jin, A model of document clustering using ant colony algorithm and validity index, in: Proceedings of 2005 IEEE International Joint Conference on Neural Networks, IJCNN'05, vol. 5, IEEE, 2005, pp. 2730–2735.
[91] Yan Yang, Mohamed S. Kamel, An aggregated clustering approach using multi-ant colonies algorithms, Pattern Recognit. 39 (7) (2006) 1278–1289.
[92] Reda Younsi, Wenjia Wang, A new artificial immune system algorithm for clustering, in: Intelligent Data Engineering and Automated Learning–IDEAL 2004, Springer, Exeter, UK, 2004, pp. 58–64.
[93] Charles T. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters, IEEE Trans. Comput. 100 (1) (1971) 68–86.
[94] Changsheng Zhang, Dantong Ouyang, Jiaxu Ning, An artificial bee colony approach for clustering, Expert Syst. Appl. 37 (7) (2010) 4761–4767.
[95] Fuzhi Zhang, Yujing Ma, Na Hou, Hui Liu, An ant-based fast text clustering approach using pheromone, in: FSKD'08, Fifth International Conference on Fuzzy Systems and Knowledge Discovery, vol. 2, IEEE, 2008, pp. 385–389.
[96] Tian Zhang, Raghu Ramakrishnan, Miron Livny, Birch: an efficient data clustering method for very large databases, in: ACM SIGMOD Record, vol. 25, ACM, 1996, pp. 103–114.
[97] Zou, Wenping and Zhu, Yunlong and Chen, Hanning and Sui, Xin, A clustering approach using cooperative artificial bee colony algorithm, Discrete Dynamics in Nature and Society., vol. 2010, 2010 (Hindawi Publishing Corporation).

Rana Forsati obtained her Ph.D. degree from Shahid Beheshti University, Tehren, Iran in summer 2014. She was a member of the NLP Research Laboratory of Electrical and Computer Engineering department. She also spent a year as a visiting research scholar at University of Minnesota at the Department of Computer Engineering from March 2013 to March 2014. Her research interests include machine learning, soft computing and data mining with applications in natural language processing and recommender systems.

Andisheh Keikha is a PhD student of Computer Engineering at Ryerson University. She received the M.Sc. degree from Shahid Beheshti University, Tehran, Iran. Her research interests include data mining, soft computing with applications in natural language processing.

Mehrnoush Shamsfard has received her BS and MS both on computer software engineering from Sharif University of Technology, Tehran, Iran. She received her PhD in Computer Engineering – Artificial Intelligence from Amir Kabir University of Technology in 2003. Dr. Shamsfard has been an assistant professor at Shahid Beheshti University since 2004. She is the head of NLP research Laboratory of Electrical and Computer Engineering faculty. Her main fields of interest are natural language processing, ontology engineering, text mining and semantic web.