International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 7 (2015) pp. 084-093

On the Sensitivity of a Bee-Inspired Algorithm to Its Internal Parameters

Dávila Patrícia Ferreira Cruz [1], Renato Dourado Maia[2] and Leandro Nunes de Castro [3]

[1] Natural Computing Laboratory (LCoN), Graduate Program in Electrical Engineering, Mackenzie University, São Paulo, Brazil
davilapatricia11@gmail.com

[2] Computer Sciences Department, State University of Montes Claros, Montes Claros, MG, Brazil
renato.dourado@unimontes.br

[3] Natural Computing Laboratory (LCoN), Graduate Program in Electrical Engineering, Mackenzie University, São Paulo, Brazil
lnunes@mackenzie.br

***Abstract*: Over the past two decades a wide range of nature-inspired clustering algorithms has been proposed in the literature with competitive performance when applied to solving real-world complex problems. One common feature of most of these algorithms is the need to set a number of internal parameters so that they can be suitably applied to these problems. These parameters are usually introduced so as to simplify or model some biological aspects of the phenomenon being modeled, but they greatly influence the performance of the proposed algorithm. The present paper takes one of these bio-inspired algorithms and investigates how its input, user-defined, parameters influence its performance. By doing that, we provide some important clues and guidelines for potential users to understand how to better set the parameters so as to take the most out of the algorithm. Two different versions of the algorithm are considered in the analysis reported here.**

***Keywords*:** swarm intelligence; optimal data clustering; dynamic size population; bee-inspired algorithms, sentitivity analysis**.**

I. Introduction

Clustering is the process of segmenting a set of objects such that those belonging to the same cluster are more similar to one another than those belonging to different clusters [1]. Its applications include, but are not restricted to, business analysis to determine groups of customers with similar behaviors, and medicine to determine groups of patients that show similar reactions to a specific medicine [2].

Swarm Intelligence [3] is one of the main Natural Computing [4] fields of research which aims at investigating and designing problem solving techniques inspired by the collective behavior of animal societies. The popularity of Swarm Intelligence has stimulated the development of several data mining algorithms [5]. The number of approaches based on Swarm Intelligence, more specifically based on bee colonies, has increased significantly over the past years [6] [7] [8] [9].

This paper presents a parametric sensitivity analysis of a specific bee-inspired clustering algorithm, called cOptBees, originally introduced in [10] [11]. This analysis was performed to evaluate the influence of its user-defined parameters on its clustering performance and to guide the adequate parameter setting according to the problem at hand. cOptBees is an adaptation of OptBees [12], an optimization algorithm inspired by the collective decision making in bee colonies, and was specially designed to solve data clustering problems. cOptBees, as well as OptBees, is able to generate and maintain the diversity of solutions by finding multiple (sub)optimal solutions in a single run. It was tested in different real-world problems and the results obtained showed high quality clusters and diversity of solutions, whilst a suitable number of clusters were automatically determined.

The present paper performs a sensitivity analysis of two different versions of the algorithm: cOptBees1, in which a bee corresponds to a partition of the data into different clusters; and cOptBees2, in which a bee represents a set of prototypes, each one representing the centroid of a cluster. After, and based on the experimental results, some general comments and guidelines are provided to point the users as to how to adequately set the input parameters for each cOptBees version.

This paper is organized as follows: Section II presents the cOptBees; Section III presents the parametric sensitivity analysis and discussion of results; and Section IV brings the conclusions and points future research.

II. cOptBees: A Bee-Inspired Algorithm for Optimal Data Clustering

cOptBees [10] [11], presented in Algorithm 1, is an algorithm that solves data clustering problems inspired by the foraging behavior of bee colonies. This algorithm is an

adaptation of OptBees, originally designed to solve continuous optimization problems and to have the ability of generating and maintaining diversity of candidate solutions in a way to find multiple local optima without compromising its global search capability [12]. In cOptBees, the active bees can play three different roles: 1) *recruiters*, responsible for attracting other bees to explore a promising region of the search space; 2) *recruited*, those recruited by recruiters to explore a promising region of the search space; or 3) *scout* bees, responsible for randomly looking for new promising regions of the space [12].

The design of cOptBees involved the proposal of two different encoding schemes for the bees and, thus, the clusters they represent. In the first scheme, which led to the algorithm named cOptBees1, each bee is represented by a vector whose length is equal to the number of objects to be clustered and each position of the vector (bee) is the label of the corresponding object, as will be detailed in Section II.A.1. In the second encoding scheme (algorithm cOptBees2), a bee corresponds to a set of prototype vectors, which altogether represent a potential partition (set of clusters) of the input data. This means that now a bee is a matrix whose number of lines is equal to the dimension of the input data plus one (because there is with one variable associated with each prototype that will indicate if it takes part in the current proposed partition or not). A bee will thus be composed of a set of prototypes,

The motivation to design these two representation schemes for cOptBees is because they result in algorithms that behave significantly differently. In the first versions the dynamics of the algorithm allows objects to be reallocated in the clusters until a satisfactory solution is found, similarly to what other search-based clustering algorithms do. The second encoding scheme is more related to self-organized clustering algorithms, such as *k*-means, aimed at finding prototypes that represent the clusters, but without explicitly solving an optimization task. The performance of these versions on a number of benchmark tasks are presented in [10] and [11], respectively.

cOptBees1 was presented in [10] and the algorithm was applied to different clustering problems, being capable of finding optimal clusters, generating and maintaining the diversity of solutions, and finding the correct number of clusters. Motivated by the results obtained by this version, the new encoding scheme, cOptBees2, was proposed in [11]. The modification in the encoding scheme required some modifications in the recruitment process as well. cOptBees2 was applied to different clustering problems and it also used to determine the centers of radial basis function (RBF) neural networks [13]. Both versions of cOptBees were compared with other algorithms from the literature, showing good results and competitive performances.

The general procedure of cOptBees is presented in Algorithm 1. The main features of cOptBees1 and cOptBees2 encoding schemes are described in the following sections.

Algorithm 1: cOptBees

1. **Input**: n_{min} (initial number of active bees); n_{max} (maximum number of active bees); ρ (inhibition radius); n_{mean} (average foraging effort); p_{min} (minimum probability of a bee being a recruiter); p_{rec} (percentage of non-recruiter bees that will be actually recruited); r_{max} (maximum number of clusters).
2. Randomly generate a swarm.
3. **while** (stopping criterion is not attained) **do**
   - 3.1. Evaluate the quality of the sites being explored by the bees.
   - 3.2. Apply local search.
   - 3.3. Determine the recruiter bees.
   - 3.4. Update the number of bees.
   - 3.5. Determine the recruited and scout bees.
   - 3.6. Perform the recruitment process.
   - 3.7. Perform the exploration process.
4. end while
5. Evaluate the quality of the sites being explored by the bees.
6. Apply local search.
7. **Output**: Bees of the swarm and their respective fitness values.

A. cOptBees 1

The cOptBees1 algorithm was proposed in [10] and will be detailed in the following sections.

1) Encoding Scheme

In cOptBees1 the swarm is represented by a matrix $\mathbf{B} \in \mathbb{R}^{n \times o}$, where n is the number of bees in the swarm and o the number of objects in the database. The initial swarm is generated randomly, respecting the maximum number of clusters, r_{max}. Each bee encodes a potential clustering and a bee can be defined as $\mathbf{b} = [b_1, b_2, \ldots, b_o]$. Each element b_i indicates the cluster of object i ($1 \leq b_i \leq r$, $r \leq r_{max}$, is the number of clusters in the dataset). Figure 1 shows the representation of a bee with $r = 3$ and $o = 9$. In this case, objects 1 to 3 belong to cluster one, objects 4 and 5 belong to cluster two and objects 6 to 9 belong to cluster 3.

1	2	3	4	5	6	7	8	9
1	1	1	2	2	3	3	3	3

Figure 1. Representation of a bee.

2) Local Search

Local search is performed by three local search operators: *exclusion*, *division* and *transformation* [14]. A single operator is applied to each bee. These operators are responsible for generating new clustering partitions at each generation, starting from the solution represented by the bee in which they are applied. The exclusion and division operators are applied with a 25% probability and the transformation operator with a 50% probability, as suggested in [14]. The operators are applied in all bees in Step 3 and work as follows:

- *Exclusion*: Randomly excludes a cluster. The objects belonging to the cluster excluded are reallocated in the cluster with the nearest centroid. This operator is applied to bees that contain more than two clusters.
- *Division*: Divides a randomly chosen cluster in two new

ones. After the division, the objects nearest to the centroid of the original cluster remain in this cluster, whilst those that are closer to the farthest object move to the new cluster.

- *Transformation*: Whenever applied, each object has a 10% probability of being changed. This operator verifies if the object is in the correct cluster by measuring its similarity to its current centroid and comparing it with its similarity to all the other centroids: if the object is in the correct cluster, its label is maintained; otherwise, it is reallocated to the cluster of the closest centroid.

3) Determination of the Recruiter Bees

The recruiter bees explore promising regions of the search space and recruit the closest bees. The number of recruited bees for each recruiter is proportional to the quality of the food sources found. Determining the recruiter bees involves three steps. In the first step, a probability p_i of being a recruiter bee is associated with each active bee:

$$p_i = \left(\frac{1 - p_{min}}{Q_{max} - Q_{min}}\right) \cdot (q_i - Q_{min}) + p_{min}, \qquad (1)$$

where q_i represents the quality of the site being explored by bee i, Q_{min} and Q_{max} represent, respectively, the minimum and maximum qualities among the sites being explored by each active bee in the current iteration (these quality values, named here fitness, are determined using the objective-function value) [12].

In the second step the bees are processed and, according to the probabilities calculated in the previous step, are now classified as recruiters or non-recruiters. In the third step, the recruiter bees are processed in accordance with the corresponding site qualities, from best to worst, and, for each recruiter bee, the other recruiters who have a high similarity are inhibited, i.e., they are classified as non-recruiters [12]. The similarity between two bees is calculated based on the objects classified in the same cluster, i.e., the greater the number of objects in common, the greater the similarity. The inhibition process happens when the similarity between two bees is greater than or equal to the inhibition radius ρ, an input parameter that represents a percentage of the maximum possible value for the similarity – the number of objects in the dataset. This process avoids that many recruiters explore the same promising regions of the search space.

4) Updating the Number of Active Bees

Updating the number of active bees aims to adapt the foraging effort in accordance with the number of recruiters and the maximum number of active bees. In a given iteration, after the determination of the recruiter bees, n_r is the number of recruiters. The number $n_d = (n_r + 1) \cdot n_{mean}$ determines the desired number of active bees, where n_{mean}, the average foraging effort, determines the desired number of non-recruiter bees for each recruiter bee. If n_d is greater than the current number of active bees, $n_{adjust} = n_d - n_{active}$ is the necessary number of bees that have to become active in order to achieve n_d active bees; if this number is less than the current number of active bees, $n_{adjust} = n_{active} - n_d$ is the necessary number of bees that have to become inactive in order to achieve n_d active bees. This process is constrained by the maximum (n_{max}) and minimum (n_{min}) numbers of active bees. If $n_d > n_{max}$, then n_d is set to n_{max}; otherwise, if $n_d < n_{min}$, then n_d is set to n_{min}. When an inactive bee becomes active, it is inserted in a random position in the search space. For the inactivation process, the bees are selected according to the corresponding site quality they explore, from the worse to the best [12].

5) Determination of the Recruited and Scout Bees

After the classification of bees as recruiters or non-recruiters, a percentage of non-recruiter bees are classified as recruited and exploit promising regions already found. The other non-recruiters are classified as scout bees, which exploit the search space to find new promising regions, reinforcing the generation and maintenance of diversity. The number of non-recruiter bees, in Step 6, is determined by $n_{nr} = n_{active} - n_r$. The number of recruited bees is $n_{rd} = [p_{rec}.n_{nr}]$, where p_{rec} is the percentage of non-recruiter bees that will be recruited and [.] denotes the nearest integer function. The number of scout bees is $n_s = n_{nr} - n_{rd}$. The process for determining the recruited bees works as follows. First, the number of recruited bees to be associated with each recruiter is determined. The relative quality of the site operated by each recruiter in relation to the others determines this number: each recruiter recruits a number of bees proportional to the quality of the site that it explores. With these numbers already determined, the non-recruiter bees are processed and associated with the most similar recruiter. After these procedures, the remaining n_s non-recruiter bees are considered scout bees [12] .

6) Recruitment Process

Recruitment is based on the idea of recombination, as proposed in [15]. Recombination consists of combining the information present in a set of solutions to create new solutions without losing the features of previous solutions. Recombination here was adapted for the clustering problem and integer encoding. Two recombination schemes were implemented: 1) *the conciliator behavior*, which is a recombination procedure that respects features present in both bees (every bee it produces contains all the values common to its two predecessor bees); and 2) *the obsequent behavior*, which respects features present in the recruiter (for the different values of both bees, the value of the corresponding recruiter bee is maintained). For example, for bees **C** = [1 1 1 1 2 3 3] and **D** = [1 1 2 2 2 2 3], assume that **C** is a recruiter and **D** is recruited. The recombination schemes using these parents would produce the offspring shown below.

Scheme	Offspring
Conciliator	[1 1 × × 2 × 3]
Obsequent	[× × 1 1 × 3 ×]

The labels of objects represented by × are replaced by the labels of the clusters with the nearest centroid.

7) Exploration Process

In the exploration process, the scout bees are moved to a random position in the search space. By doing this, the exploration process allows the scout bees to explore new regions in the search space [12].

B. cOptBees 2

In cOptBees1 the length of a bee corresponds to the number of objects in the dataset, thus the computational effort to processing the swarm may become very high for databases composed of a large number of objects. In the second encoding scheme, cOptBees2, the swarm is represented by a set of prototypes whose structure is associated with the dimension of the data. The main adaptations of cOptBees2 when compared with cOptBees1 are presented in the sequence.

1) Encoding Scheme

In this new implementation of cOptBees, each bee is composed of a set of prototypes that encode a potential clustering. A bee is defined by a matrix $\mathbf{B}_{i\times j}$, where $i = (d + 1)$, d being the number of attributes in the input data, and j the maximum number of clusters in a clustering (r_{max}). Thus, in a given column j, lines 1 to d represent the dimensions of prototype $\mathbf{C}_j$ and the last line represents a threshold value, $L_j \in [0,1]$, that defines if the centroid $\mathbf{C}_j$ is active or not. The centroid $\mathbf{C}_j$ is active when its threshold is greater than or equal to 0.5. Figure 2 shows the matrix representation of a bee [11].

$$\mathbf{B} = \begin{bmatrix} C_{1,1} & \dots & C_{1,rMax} \\ C_{2,1} & \dots & C_{2,rMax} \\ \vdots & \dots & \vdots \\ C_{d,1} & \ddots & C_{d,rMax} \\ L_1 & \dots & L_{rMax} \end{bmatrix}$$

Figure 2. Matrix representation a Bee in cOptBees.

The swarm is composed of n bees and, for each bee, the objects in the database are associated with the nearest prototype. The initial swarm is randomly generated, respecting the maximum number of clusters, r_{max} (an input parameter introduced in cOptBees).

2) Recruitment Process

In the recruitment process, the recruiter bees attract the recruited bees to the sites they explore. This recruitment process is implemented by Eq. (2) or Eq. (3), each with 50% probability, in which α is the recruitment rate, an input parameter, $\mathbf{x}_i$ is the recruited bee, $\mathbf{y}$ is the recruiter bee, u is a random number with uniform distribution in the interval [0, 1], $\mathbf{U}$ is a vector whose elements are random numbers with uniform distribution in the interval [0, 1] ($\mathbf{U}$ has the same dimension as $\mathbf{x}_i$ and $\mathbf{y}$) and the symbol ⊗ denotes the element-wise product [11]

$$\mathbf{x}_i = \mathbf{x}_i + u \,.\, \alpha \,.\, (\mathbf{y} - \mathbf{x}_i) \quad (2)$$

$$\mathbf{x}_i = \mathbf{x}_i + \alpha \,.\, \mathbf{U} \otimes (\mathbf{y} - \mathbf{x}_i) \quad (3)$$

III. Parametric Sensitivity Analysis

The parametric sensitivity analysis of cOptBees was performed to evaluate the influence of each input parameter on its behavior. The following parameters were analyzed for both versions of the algorithm: n_{min} (initial number of bees); ρ (inhibition radius); n_{mean} (mean foraging effort); p_{min} (minimum probability of a bee being a recruiter); p_{rec} (percentage of non-recruiter bees that will be actually recruited); and r_{max} (maximum number of clusters). For cOptBees2 the parameter α (recruitment rate) was also analyzed. For both versions of the algorithm, the maximum number of bees, n_{max}, was defined as 200 bees.

The choice of the parametric configuration of cOptBees1 and cOptBees2 was based on preliminary experiments. Table I summarizes the parameters and their analyzed values. Each parameter was varied individually while the others were maintained fixed in the default values.

The sensitivity analysis was performed using the Ruspini dataset [16], shown in Figure 3, which is composed of 75 objects, each one having two integer attributes, organized in four classes. For each configuration, cOptBees was run 10 times and the following aspects were analyzed: a) mean fitness value of the best bee (solution); b) mean number of recruiters, recruited and scout bees in the final swarm; and c) mean variation in the swarm size.

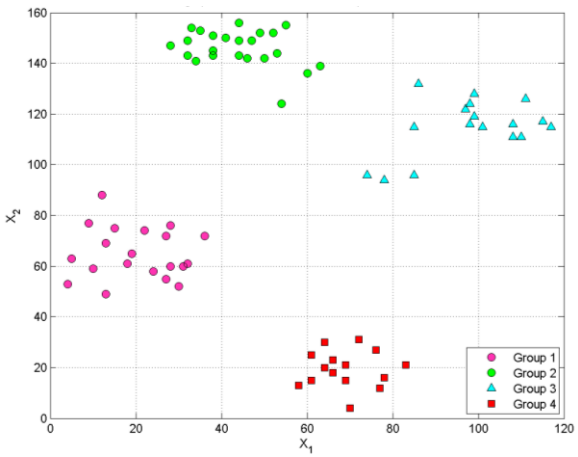


Figure 3. Graphical representation of the Ruspini dataset.

The modified Silhouette was used as quality or fitness function [17] [18]. The Silhouette for an object x_i is calculated by:

$$s(x_i) = \frac{c(x_i) - a(x_i)}{\max\{a(x_i), c(x_i)\}}, \quad (2)$$

where $a(x_i)$ represents the dissimilarity between x_i and its centroid, and $c(x_i)$ represents the dissimilarity between x_i and the closest centroid.

The next sections present the results obtained by the two versions of cOptBees and the analysis for each parameter.

TABLE I. DEFAULT AND TEST VALUES ASSESSED IN THE SENSITIVITY ANALYSIS OF CUPTBEES1 AND CUPTBEES2.

Parameter	*cOptBees*1		*cOptBees*2	
	Default value	Values tested	Default value	Values tested
n_{min}	50	5; 25; 50; 75; 100	50	5; 25; 50; 75; 100
ρ	0.2	0.05; 0.1; 0.2; 0.4; 0.8	0.2	0.1; 0.2; 0.3; 0.4; 0.5
n_{mean}	10	5; 10; 15; 20; 25	10	5; 10; 15; 20; 25
p_{min}	0.25	0.05; 0.15; 0.25; 0.35; 0.45	0.01	0.01; 0.1; 0.15; 0.20; 0.25
p_{rec}	0.9	0,1; 0,3; 0,5; 0.7; 0.9	0.7	0.1; 0.3; 0.5; 0.7; 0.9
r_{max}	8	2; 4; 8; 16; 32	8	2; 4; 8; 16; 32
α	-	-	0.5	0.15; 0.25; 0.75; 0.9

A. *Initial Number of Bees (n_{min})*

This parameter defines the number of bees that initially explore the search space. The initial swarm is generated with the minimal number of bees (n_{min}), as explained in Sections II.A.1 and II.B.1, and the number of bees in the swarm is updated in each iteration based on the number of recruiters and the foraging effort required. It is expected that the higher its value, the higher the capability of exploring the search space and, consequently, the higher the probability of finding promising regions in the first iterations. However, the computational effort increases proportionally.

For all executions and all n_{min} values tested in both versions, the best fitness was 0.8158 and all objects were grouped correctly.

Figure 4 shows the influence of the initial number of bees in the final swarm size and the number of recruiters, recruited and scout bees. This parameter does not directly influence the final swarm size, because the number of bees in the swarm is updated at each iteration based on the number of recruiters. In this case, the variation of the swarm size is produced by the variation in the number of recruiters.

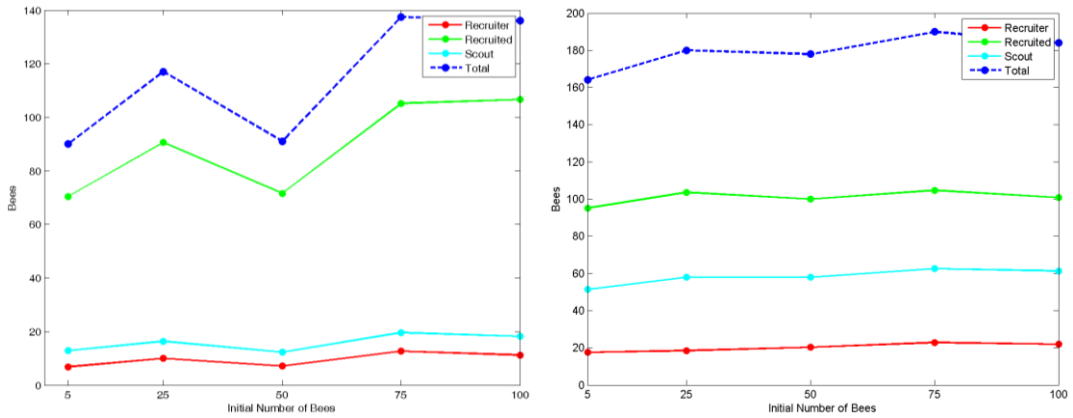
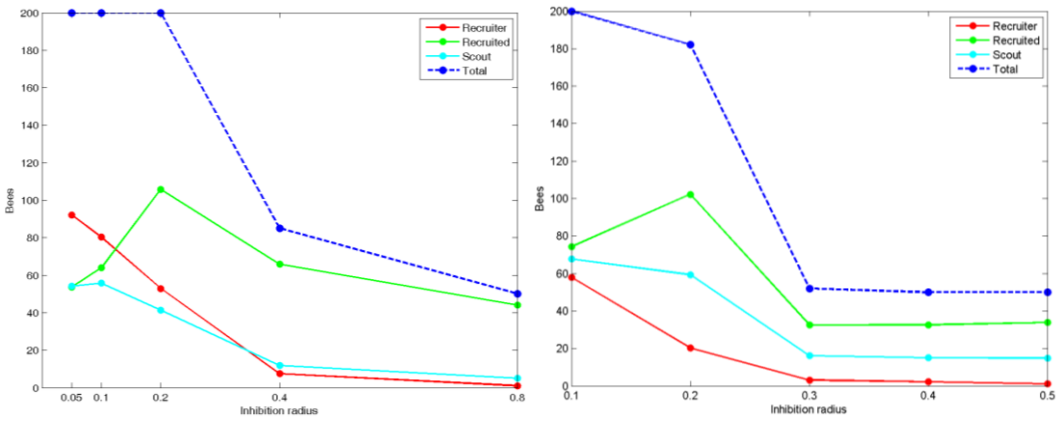


Figure 4. Sensitivity in relation to the initial number of bees. Swarm size and number of recruiters, recruited and scout bees as a function of n_{min} to a) cOptBees1, and b) cOptBees2.

B. *Inhibition Radius (ρ)*

The inhibition radius avoids that many recruiters explore the same region of the search space. The number of recruiters is directly proportional to ρ: the larger the ρ of a recruiter, the larger the region explored by it and, therefore, the lower the number of recruiters in the swarm.

In cOptBees1, for ρ equals to 0.05, 0.1, 0.2 and 0.4 the algorithm obtained a fitness value equals to 0.8158 and all objects were grouped correctly in all executions. For ρ = 0.8, in the first execution the algorithm found seven groups, with 13 objects incorrectly classified and fitness value equals to 0.7564. For the other values of ρ the algorithm grouped all objects correctly again and the fitness value was maximal. In cOptBees2, for all executions and all ρ values tested, the best fitness was the maximal one and all objects were grouped correctly.



(a) (b)

Figure 5. Sensitivity in relation to the inhibition radius. Swarm size and number of recruiters, recruited and scout bees as a function of ρ. a) cOptBees1. b) cOptBees2.

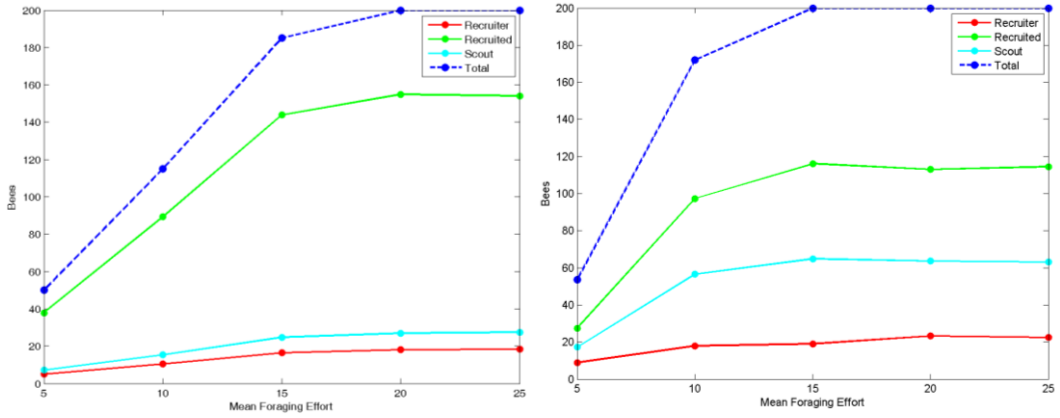
Figure 5 presents the swarm size and the number of recruiters, recruited and scout bees for each value of ρ. The results show that the number of recruiters decreases as ρ increases. For $\rho = 0.05$, cOptBees1 assigned 92.2 recruiters and cOptBees2, 141. For cOptBees1, for $\rho = 0.8$ the mean number of recruiters decreased considerably: the number of recruiters was one and all recruited bees were attracted to exploit the same region. In this case, the effort was concentrated in the search for the global optimum. For $\rho = 0.4$ the mean number of recruiters was 7.4 for cOptBees1 and 2.2 for cOptBees2, and the mean number of recruited bees was 65.8 for cOptBees1 and 32.7 for cOptBees2, which indicates a stronger exploitation and search for local optima. For lower values, such as $\rho < 0.2$, the number of recruiters was higher than the number of recruited bees. Hence, there are promising regions that may not be well exploited.

The results indicate that the number of recruiters is inversely proportional to the inhibition radius. For $\rho = 0.8$ this value decreases substantially and the algorithm found just one recruiter in both cases. The number of recruiters, in turn, directly influences the swarm size together with the n_{mean} parameter. The lower the number of recruiters (promising regions), the lower the number of recruited bees required to the exploitation processes and, consequently, the lower the number of bees in the final swarm (swarm size). In both cases, for ρ lower than 0.1 the swarm size reached the maximum number of bees, defined as an input parameter, and for the higher value tested the number of bees in the swarm was 50, the minimal number of bees defined by n_{min} parameter.

C. Mean Foraging Effort (n_{mean})

The mean foraging effort controls the swarm size based on the number of recruiters at each iteration (see Section II.A.4). For all executions the best fitness was 0.8158 and all objects were grouped correctly.

The value of n_{mean} influences the swarm size depending on the number of recruiters. The higher the value of this parameter, the higher the foraging effort spent to exploit each region found by recruiters, i.e., the higher the number of bees attracted to exploit each region. Thus, the higher the number of recruiters, the higher the number of recruited bees to explore all regions found. As shown in Figure 6, the higher the number of n_{mean}, the higher the number of bees in the final swarm. Figure 6 presents the swarm size and the number of recruiters, recruited and scout bees for each value of n_{mean}.



(a) (b)

Figure 6. Sensitivity in relation to the mean foraging effort. Swarm size and number of recruiters, recruited and scout bees as a function of n_{mean}. a) cOptBees1. b) cOptBees2.

D. Minimum Probability of a Bee Being a Recruiter (p_{min})

The probability p_i of a bee being a recruiter is a function of the minimum probability of a bee being a recruiter, p_{min}. For $p_{min} = 0$, the worst bee (i.e., the one with minimum fitness) is not able to become a recruiter; whilst $p_{min} = 0.5$ indicates that the worst bee has a 50% probability of becoming a recruiter. The probability of a bee exploring a low quality region becoming a recruiter increases with higher values of p_{min}. Therefore, higher values for p_{min} allow the selection of recruiters located in non-promising regions. This fact can compromise the quality of the solutions found. For all executions the best fitness was 0.8158 and all objects were grouped correctly.

The p_{min} value directly influences the number of recruiters and, consequently, the swarm size. For cOptBees1, as p_{min} increases, the number of recruiters also increases: for $p_{min} = 0.05$ the mean number of recruiters was 3.2 and for $p_{min} = 0.25$ the mean number of recruiters was 38.5. On the other hand, considering the parametric configuration used, the cOptBees2 does not present great variations of the mean number of recruiters: for $p_{min} = 0.01$ the mean number of recruiters was 18.5; and for $p_{min} = 0.25$ the mean number of recruiters was 27.3.

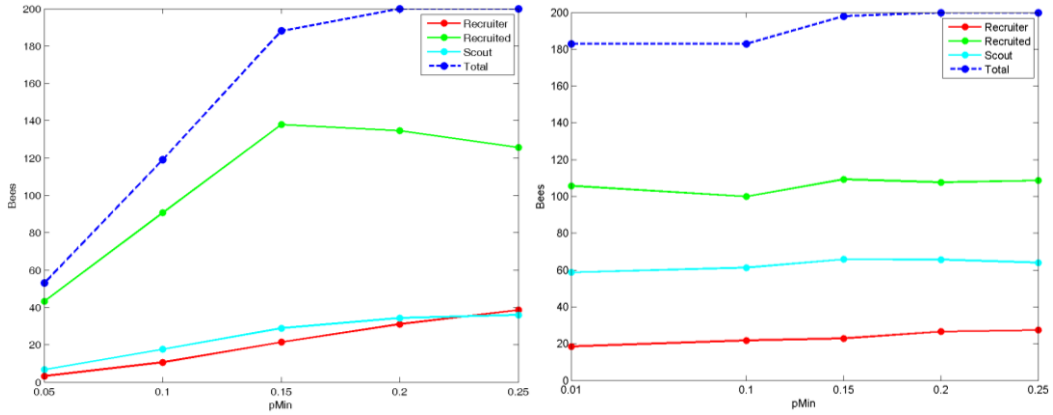
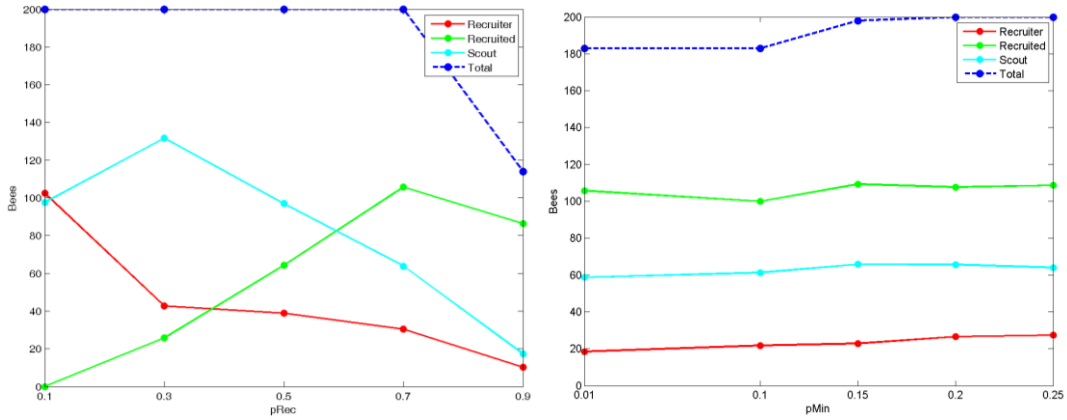


Figure 7. Sensitivity in relation to the minimum probability of a bee being a recruiter. Swarm size and number of recruiters, recruited and scout bees as a function of p_{min} to a) cOptBees1, and b) cOptBees2.

E. Percentage of Non-Recruiters that Will Be Actually Recruited (p_{rec})

Parameter p_{rec} determines the percentage of non-recruiter bees that will be recruited, and controls the exploration and exploitation efforts. Higher values for p_{rec} indicate higher exploitation and lower values indicate higher exploration [12]. For $p_{rec} = 0.1$, as shown in Figure 8(a), the average number of recruiters was 102.5 for cOptBees1 and the effort was totally dedicated to exploration, thus all bees were recruited to exploit promising regions. In this case, the algorithm obtained a fitness equals to –0.0506, which indicates that the objects are not well grouped. For $p_{rec} = 0.1$, cOptBees2 presents similar behavior, the average number of recruiters was 28.8 and effort was concentered in the exploration and search of new promising regions. In this case, the average number of recruited bees was 2 and the average number of scout bees was 169.2. The cOptBees1 presented the best results for values of p_{rec} higher than 0.5. For $p_{rec} = 0.5$, 0.7 and 0.8 the cOptBees1 obtained a fitness equal to 0.8158 and was able to find the best solution. cOptBees2 obtained the best fitness for all configurations and all objects were grouped correctly. Figure 8 presents the distribution of bees in the swarm. As p_{rec} is increased the number of recruited bees increases and, consequently, the number of scout bees decreases, indicating intensification of the exploitation in regions found by recruiters.



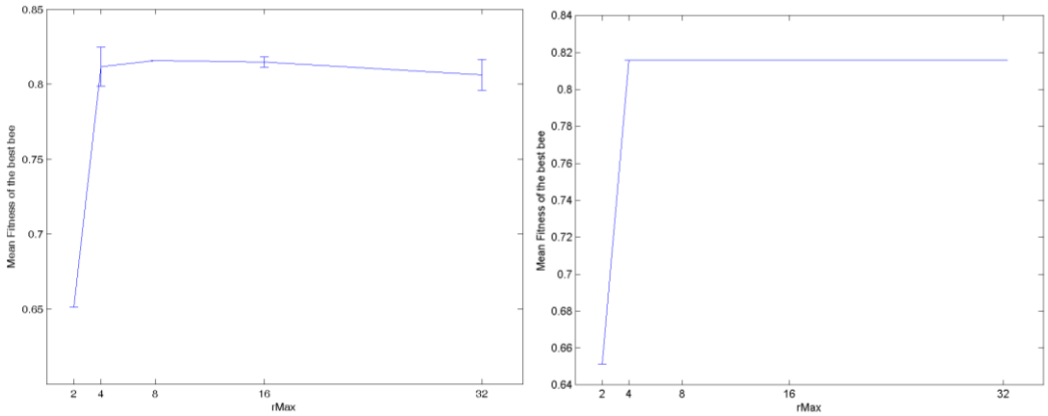
(a) (b)

Figure 8. Sensitivity in relation to the mean foraging effort. Swarm size and number of recruiters, recruited and scout bees as a function of p_{rec} to (a) cOptBees1, and (b) cOptBees2.

F. Maximum Number of Clusters (r_{max})

Parameter r_{max} indicates the maximum number of clusters that can be found in the dataset. The values tested for r_{max} were 2, 4, 8, 16 and 32. This parameter directly influences the performance of the algorithm. For r_{max} smaller than the number of clusters in the dataset, the solutions present low fitness values and, consequently, the number of bees that become recruiters is lower, as presented in Figure 9. As already mentioned, the number of recruiters influences the swarm size: the lower the number of recruiters, the lower the number of bees in the final swarm.

For r_{max} = 2 the mean number of clusters found by the best bee in all runs was 2. For r_{max} = 4 and r_{max} = 8 the number of clusters was 4 for all executions for both versions. For r_{max} = 16, the mean number of clusters found was 4.7 for cOptBees1 and 4.0 for cOptBees2. For r_{max} = 32 the mean number of clusters found was 7.4 for cOptBees1 and 4.0 for cOptBees2. The best performance of cOptBees1 was obtained for r_{max} = 8, with a fitness value of 0.8158. Considering the tested values, the cOptBees2 obtained the best fitness value for r_{max} higher than 4. It was observed that the maximum number of clusters, r_{max}, does not influence directly the distribution of bees in the swarm and final swarm size. Figure 9 presents the plot of the best fitness for each value tested. Figure 10 presents the number of recruiters, recruited and scout bees, as well as the total number of bees in the final swarm.



(a) (b)

Figure 9. Sensitivity in relation to the maximum number of clusters. Plot of the best fitness related to r_{max} for (a) cOptBees1, and (b) cOptBees2.

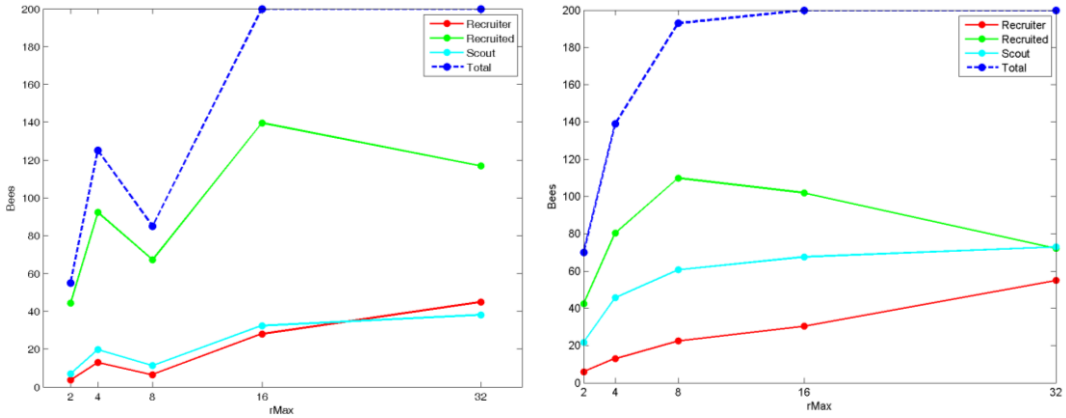


Figure 10. Sensitivity in relation to the maximum number of clusters. Swarm size and number of recruiters, recruited and scout bees as a function of r_{max} for (a) cOptBees1 and (b) cOptBees2.

G. Recruitment Rate

The recruitment rate is an input parameter of cOptBees2 and it defines how many of the recruited bees had approached the recruiters, as explained in Section II.B.2. This parameter does not directly influence the distribution of bees in the swarm. For all values tested cOptBees2 was able to group all objects correctly and the fitness value was 0.8158. Figure 11 presents the number of recruiters, recruited and scout bees, as well as the total number of bees in the final swarm. As shown in Figure 11, as the recruitment rate increases, so does the number of recruiters. For α = 0.15, 15.1 bees were classified as recruiters and for α = 0.9 the mean number of recruiters was 24.4. As already mentioned, the number of recruiters influences directly the total number of bees in the swarm, as shown in Figure 11. For α = 0.9 the number of bees in the swarm reached the maximum number of bees, defined as 200 bees.

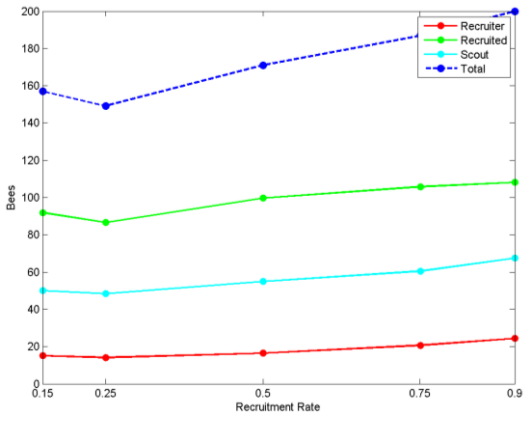


Figure 11. Sensitivity in relation to the recruitment rate. Swarm size and number of recruiters, recruited and scout bees as a function of α.

IV. Discussion

Both cOptBees versions present efficient mechanisms to control the swarm size and adjust the swarm effort to exploit or explore the search space. The sensitivity analysis is important to show the behavior of cOptBees in relation to each input parameter, to investigate the impact of each parameter in the performance of cOptBees, and to guide the parametric configuration according to the problem at hand. Some parameters, such as p_{min} and ρ, have a greater impact on the algorithm performance.

The dynamic variation of the swarm size allows the adaptation of the computational effort for each problem. This variation is based on the number of recruiters (promising regions found). The number of bees in the final swarm is directly influenced by the number of recruiter bees (number of promising regions).

The number of recruited and scout bees is influenced by the number of recruiters, p_{rec}. This input parameter must be set according to the problem: higher values produce a larger number of recruited and are adequate in problems where the intense exploitation of promising regions is needed; whilst lower values result in larger numbers of scout bees and are used in problems that require more exploration than exploitation. The number of recruiter bees directly impacts the diversity of solutions, since the recruiters represent the best and diverse solutions found by the algorithm.

The inhibition radius directly influences the number of promising regions and allows the algorithm to search for global or local optima, generating diverse solutions. The quality of the solutions found by the recruiter bees can be influenced by the maximum number of clusters, p_{rec} and p_{min}.

V. Conclusions and Future Research

This paper presented a sensitivity analysis of two versions of cOptBees, an algorithm inspired by the foraging behavior of bee colonies for performing optimal data clustering. The analysis was performed to increase the understanding of the influences of the many parameters in the behavior of the algorithm. To assess the impact of each input parameter in the performance of cOptBees, the algorithm was tested with different configurations. Each parameter was varied individually while the other ones were maintained fixed.

The quality of the clusters found by cOptBees, based on an internal quality function, and the final swarm size are directly proportional to the number of promising regions, represented

by recruiter bees, and the foraging effort in these regions. The number of recruiters, in turn, is directly influenced by the minimum probability of a bee being a recruiter, p_{min}, and by the inhibition radius, ρ.

Important future research include the investigation of the impact of the parameters on the diversity of solutions, the assessment on different types of clustering tasks, and the use of various internal and external measures in the assessments.

Acknowledgments

The authors thank Capes, CNPq, MackPesquisa, Fapesp #2013/12005-7 and Fapemig for the financial support.

References

[1] J. Han and M. Kamber, Data Mining: Concepts and Techniques, San Francisco, 2006.

[2] R. Elmasri and S. B. Navathe, Sistemas de Banco de Dados, São Paulo: Pearson, 2006.

[3] J. Kennedy, R. Eberhart and Y. Shi, Swarm Intelligence, Morgan Kauffman Publishers, 2001.

[4] L. N. De Castro, Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications, Chapman & Hall/CRC, 2006.

[5] D. Martens, B. Baesens and T. Fawcett, "Editorial Survey: Swarm Intelligence for Data Mining," *Machine Learning*, vol. 82, no. 1, pp. 1-42, 2011.

[6] D. Karaboga and B. Akay, "A survey: algorithms simulating bee swarm intelligence," *Artificial Intelligence Review*, vol. 31, no. 1, pp. 61-85, 2009.

[7] D. Karaboga and C. Ozturk, "Expert Systems with Applications," *Applied Soft Computing*, vol. 11, no. 1, p. 652–657, 2011.

[8] C. Zhang, D. Ouyang and J. Ning, "An artificial bee colony approach for clustering," *Expert Systems with Applications*, vol. 37, no. 7, p. 4761–4767, 2010.

[9] X. Yan, Y. Zhu, W. Zou and L. Wang, "A new approach for data clustering using hybrid artificial bee colony algorithm," *Neurocomputing*, vol. 95, p. 241–250, 2012.

[10] D. P. F. Cruz, R. D. Maia, L. N. De Castro and A. Szabo, "A Bee-Inspired Algorithm for Optimal Data Clustering," in *IEEE World Congress on Computational Evolutionary*, Cancún, 2013.

[11] D. P. F. Cruz, R. D. Maia and L. N. De Castro, "A New Encoding Scheme for a Bee-Inspired Optimal Data Clustering Algorithm," in *1st BRICS Countries Congress (BRICS-CCI)*, Recife, 2013.

[12] R. D. Maia, L. N. De Castro and W. M. Caminhas, "Collective Decision-Making by Bee Colonies as Model for Optimization-the OptBees Algorithm," *Applied Mathematical Sciences*, vol. 7, no. 87, pp. 4327-4351, 2013.

[13] D. P. F. Cruz, R. D. Maia, L. A. d. Silva and L. N. d. Castro, "A Bee-Inspired Data Clustering Approach to Design RBF Neural Network Classifiers," in *Advances in Intelligent Systems and Computing*, Springer International Publishing, 2014, pp. 545-552.

[14] E. BORGES, D. G. FERRARI and N. L. DE CASTRO, "Opt-aiNet: An Artificial Immune System for Optimal Clustering," *Revista Mackenzie de Engenharia e Computação*, vol. 11, no. 1, pp. 117-140, 2012.

[15] C. Cotta and P. Moscato, "The Parameterized Complexity of Multiparent Recombination," in *6th Metaheuristics International Conference*, Vienna, 2005.

[16] D. M. Hamby, "A review of techniques for parameter sensitivity analysis of environmental models," *Environmental Monitoring and Assessment*, vol. 32, no. 2, pp. 135-154, 1994.

[17] E. H. Ruspini, "Numerical methods for fuzzy clustering," *Information Sciences*, vol. 2, no. 3, p. 319–350, 1970.

[18] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, p. 53–65, 1987.

[19] E. R. Hruschka, L. N. De Castro and R. J. G. B. Campello, "Evolutionary algorithms for clustering gene-expression data," in *IEEE 13th International Conference on Data Mining*, Brighton, 2004.

[20] J. E. Gadau and J. Fewell, Organization of Insect Societies: from genome to sociocomplexity, Cambridge: Harvard University Press, 2009.

Author Biographies

Dávila Patrícia Ferreira Cruz has a B.Sc in Computer Engineering from Faculty of Science and Technology of Montes Claros – FACIT (Bee-inspired algorithm to solve clustering problems). She was teacher and coordination assistant at Education Technological Center of Montes Claros from 2008 to 2013. She is currently M.Sc. student at Graduate Program in Electrical Engineering at the Mackenzie University and she is a member of Natural Computing Laboratory (LCoN). Her main research lines are Swarm Intelligence emphasizing bee-inspired algorithms and Data Mining, emphasizing Data Clustering and Data Classification tasks.

Renato Dourado Maia has a B.Sc in Control and Automation Engineering, a M.Sc. and a Ph.D. in Electrical Engineering, all from the Federal University of Minas Gerais. He is currently an Adjunct Professor at the Institute of Agricultural Sciences of the Federal University of Minas Gerais, and at the Computer Science Department of the State University of Montes Claros, and the Science and Technology Faculty of Montes Claros, where he is the coordinator of the Control and Automation Engineering course. He has experience in Engineering of Control and Automation, Computer Science and his main research line is Natural Computing emphasizing bioinspired computation, Control and Monitoring Systems, Optimization and Data Analysis.

Leandro Nunes de Castro has a B.Sc. in Electrical Engineering from the Federal University of Goias, an M.Sc. and a Ph.D. in Electrical Engineering from Unicamp and an MBA in Strategic Business Management from the Catholic University of Santos. He was a Research Associate at the University of Kent at Canterbury from 2011 to 2002, a Research Fellow at Unicamp from 2002 to 2006, a Visiting Lecturer at the Universiti Teknologi Malaysia in September 2005, a Visiting Professor at Unicamp in 2012, and a Visiting Researcher at the University of Salamanca in 2014. He is currently a Professor at the Graduate Program in Electrical and Computer Engineering at the Mackenzie University, where he founded and leads the Natural Computing Laboratory (LCoN). His main research lines are Natural Computing and Data Mining, emphasizing Artificial Immune Systems, Neural Networks, Swarm Intelligence, Evolutionary Algorithms and real-world applications.