Clustering via Ant Colonies: Parameter Analysis and Improvement of the Algorithm

Jeffry Chavarría-Molina, Juan José Fallas-Monge and Javier Trejos-Zelaya

Abstract An ant colony optimization approach for partitioning a set of objects is proposed. In order to minimize the intra-variance, or within sum-of-squares, of the partitioned classes, we construct ant-like solutions by a constructive approach that selects objects to be put in a class with a probability that depends on the distance between the object and the centroid of the class (visibility) and the pheromone trail; the latter depends on the class memberships that have been defined along the iterations. The procedure is improved with the application of K-means algorithm in some iterations of the ant colony method. We performed a simulation study in order to evaluate the method with a Monte Carlo experiment that controls some sensitive parameters of the clustering problem. After some tuning of the parameters, the method has also been applied to some benchmark real-data sets. Encouraging results were obtained in nearly all cases.

1 Introduction

Cluster analysis, or clustering, is one of the main tools in Data Analysis and Machine Learning, since it intends to discover groups or classes in large data sets of objects described by observed variables, simplifying this way the set with a small number of clusters. Most clustering methods are based on dissimilarities, graphs, models, or densities. In our case, we will deal with dissimilarities or distances for numerical data sets. There are two main families in this case: partitioning methods and hierarchical ones, being K-means and agglomerative hierarchical methods, respectively,

J. Chavarría-Molina · J. J. Fallas-Monge
School of Mathematics, Costa Rica Institute of Technology, Cartago, Costa Rica
e-mail: jchavarria@itcr.ac.cr

J. J. Fallas-Monge
e-mail: jfallas@itcr.ac.cr

J. Trejos-Zelaya (✉)
CIMPA–School of Mathematics, University of Costa Rica, San José, Costa Rica
e-mail: javier.trejos@ucr.ac.cr

T. Imaizumi et al. (eds.), *Advanced Studies in Behaviormetrics and Data Science*,
Behaviormetrics: Quantitative Approaches to Human Behavior 5,
https://doi.org/10.1007/978-981-15-2700-5_16

the most widely used in practice. Both have local optimality problems: local minima that depend on initialization for K-means, greedy procedure for agglomerative hierarchical clustering.

Several combinatorial optimization metaheuristics have been used for cluster partitioning (Handl & Knowles [14]; Ng & Wong [21]; Sarkar, Yegnanarayana, & Khemani [23]; Trejos, Murillo, & Piza [27]). In this article, we deal with partitioning for numerical data sets, using an ant colony optimization (ACO) approach in order to overcome the local optima problem.

According to Handl and Knowles [14], published in 2006, "a few implementations of ACO have been proposed for data clustering, with the construction graph typically employed to directly represent cluster assignments (Handl & Meyer [15]; Runkler [22])".

In 2004, we published a first paper on clustering using an ant colony optimization approach (Trejos, Murillo, & Piza [28]) for the minimization of the within sum-of-squares criterion. In that method, ants were associated with partitions that were modified during the iterations, according to a probability of selection that depends on the visibility (proportional to the distance between the objects) and the pheromone trail (which depends on the fact that the objects have been classified together in the partitions). The pheromone matrix measured relation intensity between pairs of objects.

By that time, Shelokar, Jayaraman, and Kulkarni [24] published another clustering method based on ACO for minimizing the same criterion as in Trejos, Murillo, and Piza [28], with a pheromone trail but no local heuristic. The pheromone matrix relates objects and clusters, and it is defined by the inverse of the objective function. The matrix is used as a kind of adaptive memory that contains information provided by the previously found superior solutions, and is updated at the end of each iteration (Shelokar et al. [24]). This information is considered by the other ants to continue the clustering process. However, it is not clear how the authors selected the parameters to execute the ACO algorithm. They indicate that several simulations were performed to find the algorithm parameters (Shelokar et al. [24]), but they do not present details about the process. They also present a comparison among their ants algorithm and other heuristic methods such as genetic algorithm, simulated annealing, and tabu search.

Later on, Kao and Cheng in a short paper [17] improved Shelokar's algorithm introducing a local heuristic or visibility based on the inverse of the distance between objects and class centers. The pheromone trail is also defined by the inverse of the criterion and the algorithm follows almost the same steps as Shelokar algorithm (Shelokar et al. [24]), with the difference that visibility is introduced.

Neither Shelokar et al. [24] nor Kao and Cheng [17] give a detailed analysis on the choice of parameters for their methods.

In the present article, we use ACO with ants constructing partitions. The strategy is based on the traveling salesman problem (TSP) in a similar way as it was tackled in Bonabeau, Dorigo, and Theraulaz [4] with ACO, in our case for the clustering problem. It is a constructive method, in which each ant builds a partition. This part of the process is similar to the ideas presented in Kao and Cheng [17] and Shelokar et al. [24], which were previously presented; but this paper deals with three different

aims: first, developing a fitting parameters analysis studying the algorithm behavior in the clustering problem according to its parameters. Second, we introduce a local search procedure based on the K-means algorithm, to improve the basic ACO (BACO) algorithm performance. And finally, to develop a performance comparison among the K-means algorithm (KM), the BACO algorithm and the BACOK (BACO improved with the local search procedure) algorithm.

The article is organized as follows. Section 2 contains the mains concepts of clustering we use in the article, introducing the main notation we need. In Sect. 3 the artificial ant concept is explained and the ACO classical algorithm is presented. In Sect. 4 we introduce the proposed ACO algorithm. Section 5 describes the experiment performed. Sections 6 and 7 present the results and some remarks.

2 Clustering

Cluster analysis, or clustering, deals with finding homogeneous groups of objects such that similar objects belong to the same class and it is possible to distinguish between objects in different classes. Cluster analysis can be defined as an optimization problem in which a given function consisting of *within cluster similitary* and *among clusters dissimilarities* need to be optimized (Jafar & Sivakumar [16]; Xavier & Xavier [30]). In the numerical case, there is a set of objects $\Omega = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ such that $\mathbf{x}_i \in \mathbb{R}^p$, for all i, that is, the objects are described by p numerical or quantitative variables. The most widely used criterion (Everitt, Landau, Leese, & Stahl [9]) is the minimization of the within sum-of-squares, also known as within inertia or variance:

$$W = \frac{1}{n} \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - \mathbf{g}_k\|^2, \tag{1}$$

where K is the number of classes or clusters (number fixed a priori), $P = (C_1, C_2, \ldots, C_K)$ is a partition of Ω, and $\mathbf{g}_k$ is the barycenter or mean vector of C_k. Minimizing $W(P)$ is equivalent to maximizing the between sum-of-squares (between inertia and variance):

$$B = \sum_{k=1}^{K} \frac{|C_k|}{n} \|\mathbf{g}_k - \mathbf{g}\|^2,$$

where $\mathbf{g}$ is the overall barycenter and $|C_k|$ is the cardinality of class C_k, since the sum $I = W(P) + B(P)$ is a constant (the total inertia) (Everitt, Landau, Leese, & Stahl [9]).

The $W(P)$ function is not a convex function, thus $W(P)$ could have several local minima (Ng & Wong [21]; Sarkar et al. [23]). This feature causes the traditional clustering algorithms based on local search, such as K-means, to find mostly local minima (Trejos et al. [27]). Furthermore, the global optimization algorithms (such as linear programming, interval methods, branch, and bound methods) present a high sensitivity to relatively high-dimensional data tables, in which the algorithms'

probability for finding the optimal partition is very low. In those cases, algorithms report solutions that differ significantly from the optimum clustering (Bagirov [2]). Those features represent a challenge to try to find alternative optimization strategies, and combinatorial optimization heuristics are a viable option.

In recent years heuristic algorithms have been used to solve complex optimization problems, since their random nature is useful to efficiently avoid the convergence to local minima (Babu & Mutry [1]; Klein & Dubes [19]; Trejos et al. [27]). As particular examples of optimization heuristics used in clustering it is possible to cite simulated annealing, tabu search, genetic algorithms, particle swarm optimization, and ant colony optimization.

In the particular case of ant colony optimization, there are several contributions, as the already mentioned (Kao & Cheng [17]; Shelokar et al. [24]; Trejos et al. [28]), and some other more recent (Handl & Meyer [15]; Handl & Knowles [14]; Runkler [22]; Zhe et al. [31]).

3 Artificial Ant Colonies

The optimization approach based on ant colonies (ACO) is part of a large group based on swarm intelligence. It was proposed by Marco Dorigo in 1992, to solve several discrete optimization problems (Dorigo, DiCaro, & Gambardella [6]; Jafar & Sivakumar [16]), and since then it has been applied to several combinatorial optimization problems. This method, like every metaheuristic, depends on parameters which control several decisions taken in the process. There are several papers which develop parameters analysis for the ACO algorithm. In Gaertner and Clark [13] an empirical analysis of the sensitivity of the ACO algorithm to variations of some parameters for different instances of the TSP (traveling salesman problem) is presented. Similarly, in Wei [29] an experiment with parameter combinations is shown, in order to improve the speed of convergence of the ACO algorithm in the TSP. Also, this author indicates that at present the parameter settings and properties research of basic ant colony algorithm are mostly still in the experimental stage (Wei [29]) Meanwhile, Stützle et al. [25] provides an extensive review of available research results on parameter adaptation in ACO algorithms. They mention that ACO algorithms involve a number of parameters that need to be set appropriately, in particular α, β (both used to weigh the relative influence of the pheromone) and ρ (evaporation rate parameter, $0 \leq \rho \leq 1$). A parameter selection in the TSP context is developed in Dorigo, Maniezzo, and Colorni [8], in three different experiments. They tested the ranges: $\alpha \in \{0, 0.5, 1, 2, 5\}$, $\beta \in \{0, 1, 2, 5\}$, $\rho \in \{0.3, 0.5, 0.7, 0.99, 0.999\}$ and $Q \in \{1, 100, 10000\}$. The numbers $\alpha = 1$ and $\beta = 5$, were selected as the best values for these parameters. Parameter ρ was fixed, depending on the experiment, in 0.99, 0.99 or $\rho = 0.5$. And finally, parameter Q was found to be negligible.

In nature, the optimization developed by ants while they look for food consists basically of minimizing the distance between the nest and food. For this reason, the first application of ACO was to the TSP (Bonabeau et al. [4]). In that problem the agent should visit n cities, all interconnected, visiting all cities just one time and then returning to the departure city, minimizing the distance.

In this paper, the TSP idea is used to study the clustering optimization problem. Thus, it is necessary to introduce artificial ants; that is, agents in charge of finding a feasible solution in the search space. During this process the ant will drop artificial pheromones so that other ants can rebuild the same solution. Pheromones should be volatile (disappear in time on the trails that have not been intensified) and have to increase on the shortest trails while the number of iterations increases (Dorigo et al. [6]).

The pheromone update formula applied in the TSP is given by $\tau_{uv} = (1-\rho)\tau_{uv} + \rho\Delta\tau_{uv}$ (Barcos, Rodríguez, Álvarez, & Robusté [3]; Dorigo, Birattari, & Stützle [5]; Dorigo & Gambardella [7]), where τ_{uv} is the pheromone present on the trail from u to v, ρ is the evaporation rate, and

$$\Delta\tau_{uv} = \sum_{m=1}^{M} \Delta\tau_{uv}^{m},$$

where M is the number of ants, and $\Delta\tau_{uv}^{m}$ is the pheromone dropped by the m-th ant on the trail (u, v), normally given by

$$\Delta\tau_{uv}^{m} = \begin{cases} Q/d_m & \text{if ant } m \text{ walks across } (u, v) \\ 0 & \text{otherwise;} \end{cases}$$

where Q is a parameter to be fitted and d_m represents the total distance walked by ant m.

An alternative way to deal with pheromones is to make local updatings, that is, every time an ant goes from node u to node v, a local pheromone update is applied on the trail (u, v) (Dorigo & Gambardella [7]). A possible local update formula is $\tau_{uv} = \tau_{uv} + \dfrac{Q}{d_{uv}}$, where Q is a parameter to be fitted and d_{uv} is the distance between u and v. When all ants finish their trips, the pheromone is updated by applying the evaporation rate.

On the other hand, each ant has to decide to which node it goes from the current node. In that choice three factors are fundamental: visibility, pheromone trail, and a probabilistic factor. Thus, if T_m represents the route built by the ant m while it is on the node u, then the probability of going to the node v is given by

$$p_{uv}^{m} = \begin{cases} \dfrac{[\tau_{uv}]^{\alpha} \cdot [\eta_{uv}]^{\beta}}{\sum\limits_{s \notin T_m} [\tau_{us}]^{\alpha} [\eta_{us}]^{\beta}} & \text{if } v \notin T_m \\ 0 & \text{if } v \in T_m; \end{cases}$$

where η_{uv} is the visibility, defined by $\eta_{uv} = 1/d_{uv}$, with d_{uv} the distance from the node u to node v; τ_{uv} is the pheromone on the trail (u, v), and α and β are parameters to be fitted (Barcos et al. [3]; Dorigo et al. [6]; Kennedy & Eberhart [18]).

To stop the algorithm, Bonabeau et al. [4] proposed using a maximum iteration number. The disadvantage of this procedure is that it could stop the algorithm while

it is still improving the solutions. Also, Dorigo et al. [8] considered investigating a stagnation behavior of all ants traveling the same path. A stagnation process is present if a percentage of the ants has the same distance in their paths. Thus, it is almost certain that those ants are traveling the same path, or at least, that they are traveling paths with the same cost value.

In Algorithm 1, the classical ACO algorithm is shown.

Algorithm 1 ACO algorithm

```
Require: Initial parameters.
Set parameters and initialize pheromone trails.
while stop criterion is not satisfied do
  for t ← 1 to total of nodes do
    for m ← 1 to M do
      Move ant m to a new position.
      Update T_m.
      Update the local pheromones (optional).
    end for
  end for
  Update the global pheromones.
  Keep the best solution in this iteration if it improves the best in memory.
end while
return The best solution built.
```

4 Description of the Proposed ACO Algorithm

The method starts by defining a list of M artificial ants $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_M$, that will build a data clustering in K classes (or clusters). At the beginning, it is possible to define the best ant in the colony, denoted by $\mathbf{h}^*$, equal to $\mathbf{h}_m$ for some $m = 1, 2, \ldots, M$, because in that moment there is no comparison parameter among them; thus the assignment could be random.

For ant $\mathbf{h}_m$, with $m = 1, 2, \ldots, M$, K random points in the space of individuals (a hyperrectangle that contains all individuals) are considered, denoted by $\mathbf{g}_1^m, \mathbf{g}_2^m, \ldots, \mathbf{g}_K^m$. These points are interpreted as the initial centroids. C_k^m denotes the class k, with centroid $\mathbf{g}_k^m$, which has been built by ant m. Also, $\mathbf{h}_m$ has a tabu list L_m, which is a short term memory that contains the objects classified by $\mathbf{h}_m$. In each iteration, in order to complete the tour, ant m has to classify the objects not in L_m. When the iteration is done, all objects should be in L_m, this guarantees that the clustering process is complete.

During the clustering process, each ant randomly chooses an object that is not in its tabu list. Then, the ant should randomly select a class in which to classify the object. If ant m selects object i, then the process to choose the class uses a probabilistic roulette (see Talbi [26]). The probability that $\mathbf{h}_m$ assigns object i to class C_k^m is denoted by p_{ik}^m. To calculate this probability it is necessary to consider the following factors:

- **Visibility**: This factor is denoted by η_{ik}^m, and it consists of the visibility of $\mathbf{h}_m$, located on object $\mathbf{x}_i$, to "see" class C_k^m. The visibility is defined as the reciprocal of the distance from object $\mathbf{x}_i$ to $\mathbf{g}_k^m$, the centroid of class C_k^m. Thus, $\eta_{ik}^m := \frac{1}{d_{ik}^m}$, where $d_{ik}^m = d^2(\mathbf{x}_i, \mathbf{g}_k^m) = \left\|\mathbf{x}_i - \mathbf{g}_k^m\right\|^2$. If the visibility which $\mathbf{h}_m$ has of class C_k^m is large, then the probability of classifying $\mathbf{x}_i$ in class k is also large.
- **The pheromone trail**: The pheromone trail perceived by $\mathbf{h}_m$ on the arc from $\mathbf{x}_i$ to $\mathbf{g}_k^m$ is denoted by τ_{ik}. It quantifies pheromones that have been dropped by all ants which have classified the same object $\mathbf{x}_i$ in its respective class k. If τ_{ik} is large, then the probability of assigning class k to cluster $\mathbf{x}_i$ is going to increase.

Equation (2) shows the formula used to calculate p_{ik}^m, considering visibility and the pheromone trail, inspired by the corresponding formula used by the agent in the TSP:

$$p_{ik}^m := \frac{[\tau_{ik}]^\alpha \cdot [\eta_{ik}^m]^\beta}{\sum_{r=1}^{K} [\tau_{ir}]^\alpha \cdot [\eta_{ir}^m]^\beta}, \tag{2}$$

where α and β are parameters to be fitted.

On the other hand, when $\mathbf{h}_m$ chooses class C_k^m for object $\mathbf{x}_i$, the ant will register index i in the respective tabu list L_m. Futhermore, $\mathbf{h}_m$ should do the following processes related to the assignment.

- **Local pheromone update**: Ant $\mathbf{h}_m$ should drop a pheromone trail between object $\mathbf{x}_i$ and class C_k^m. To do this, an auxiliary pheromone matrix was defined, denoted by Γ_{aux} with size $n \times K$, such that entry ik of Γ_{aux} contains pheromones between x_i and class k. This matrix has the format presented in Table 1.

  Ant $\mathbf{h}_m$ will drop $\Delta\tau_{ik}^m$ pheromones. This quantity is defined by $\Delta\tau_{ik}^m := \frac{Q}{d_{ik}^m}$, where Q is a parameter to be fitted. Finally, the local pheromone update is done by adding $\Delta\tau_{ik}^m$ with the current entry ik of Γ_{aux}.
- **Centroid update**: The final step in this process is to update the centroid $\mathbf{g}_k^m$ of class C_k^m. One possibility is using its definition $\mathbf{g}_k^m := \frac{1}{|C_k^m|} \sum_{\mathbf{x} \in C_k^m} \mathbf{x}$. This option is not advisable because there are several unnecessary calculations. If fact, it is possible to update $\mathbf{g}_k^m$ recursively using its value in the previous iteration in case object $\mathbf{x}_i$

Table 1 Auxiliary pheromone matrix Γ_{aux}

	C_1	C_2	C_3	$\cdots$	C_K
$\mathbf{x}_1$					
$\mathbf{x}_2$					
$\mathbf{x}_3$					
$\vdots$					
$\mathbf{x}_n$					

is transferred to class C_k^m. In Trejos et al. [27] the following formula is proven and is used to update the centroids more efficiently: $\mathbf{g}_k^m := \frac{1}{|C_k^m|}\left[\left(|C_k^m| - 1\right)\mathbf{g}_k^m + \mathbf{x}_i\right]$.

After each ant has clustered one object, it should randomly select a new object that is not in its tabu list. Next, the ant should follow the process previously described. This process is done n times, clustering all objects by all ants.

When the process ends, each ant has a complete clustering of objects with the respective barycenters. Also, matrix Γ_{aux} contains pheromones that were dropped by ants. Entry ik of Γ_{aux} contains pheromone $\Delta\tau_{ik}$, which has been dropped by all ants that classified object i in its respective class k. This quantity is represented by

$$\Delta\tau_{ik} = \sum_{m=1}^{M} \Delta\tau_{ik}^m.$$

The next step is to calculate, for each ant, the within inertia. To do this, the classification done by each ant, and the respective barycenters, should be considered. Also, if one of the ants has a within inertia less than $W(\mathbf{h}^*)$ (the best inertia so far in memory), then $\mathbf{h}^*$ (the best ant in memory) is required to be updated.

Global pheromones are stored in a matrix Γ with the same structure as Γ_{aux}. At the beginning, this matrix is initialized with values close to zero (indicating pheromone absence). When the travels of all ants finish, Γ is updated in entry ik by $\Gamma_{ik} := (1-\rho)\Gamma_{ik} + \rho\Delta\tau_{ik}$, where ρ is the pheromone evaporation rate.

When the pheromone updating process is done, matrix Γ_{aux} is initialized, to be used in the next iteration. Also, tabu lists (one per ant) are initialized, to start a new classification process.

As the final step to conclude the current iteration, an intensification process done by the best ant (the ant with lowest within inertia, denoted by $\mathbf{h}^*$) is developed. $\mathbf{h}^*$ repeats her path dropping extra pheromones in arcs it visited. The intensification follows the following rule:

$$\Gamma_{ik} := \begin{cases} \Gamma_{ik} + \frac{Q}{W(\mathbf{h}^*)} & \text{if the object } i \text{ is in the class } k \text{ of } \mathbf{h}^*, \\ \Gamma_{ik} & \text{otherwise;} \end{cases}$$

where $W(\mathbf{h}^*)$ denotes the within inertia of the classification done by $\mathbf{h}^*$. This ends the current iteration and a new clustering process is started, considering the following information: the global pheromone matrix Γ, the barycenters of ants, which will be used as the initial centroids for the new classes, and the best ant $\mathbf{h}^*$.

Algorithm 2 presents a detailed pseudocode of the BACOK. The K-means algorithm was applied (see line 19 in Algorithm 2) to each ant. The method is applied after all ants have built their respective classifications, and until the absolute difference between current inertia and previous inertia is less than 0.0001. Algorithm 3 shows how the local search strategy based on K-means works. If lines from 19 to 22 are eliminated from Algorithm 2, then BACO algorithm pseudocode is obtained. Finally, in the event that there has been no improvement, Algorithm 2 uses an iteration number (10 iterations) as stopping criterion (see line 4). Consider that, Counter

is increments in line 5, but its value must be returned to zero every time a better solution (comparing with the best in memory) is found. This stopping criterion is based on the stagnation behavior concept presented in Dorigo et al. [8].

Algorithm 2 BACOK algorithm.

Require: n (number of individuals), p (number of variables), K (number of clusters), M (number of ants), and the parameters α, β, Q and ρ.

1: Build the initial colony with m ants: $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_M$.
2: For each $m = 1, 2, \ldots, M$ define $L_m = \varnothing$, and randomly choose $\mathbf{g}_1^m, \ldots, \mathbf{g}_K^m$.
3: Counter $\leftarrow 0$
4: **while** Counter ≤ 10 **do**
5: Counter $\leftarrow$ Counter $+ 1$
6: **for** $I := 1$ **to** n **do**
7: **for** $m := 1$ **to** M **do**
8: Ant $\mathbf{h}_m$ chooses a random individual $\mathbf{x}_i$, such that $i \notin L_m$.
9: Ant $\mathbf{h}_m$ chooses $k :=$ Roulette(p_{ik}^m), where $p_{ik}^m := \frac{[\tau_{ik}]^\alpha \cdot [\eta_{ik}^m]^\beta}{\sum_{r=1}^{K} [\tau_{ir}]^\alpha \cdot [\eta_{ir}^m]^\beta}$.
10: Individual $\mathbf{x}_i$ and index i are assigned to C_k^m and L_m, respectively.
11: Let $\langle \Gamma_{aux} \rangle_{ik} := \langle \Gamma_{aux} \rangle_{ik} + \Delta\tau_{ik}^m$, where $\Delta\tau_{ik}^m = \frac{Q}{d_{ik}^m}$.
12: Let $\mathbf{g}_k^m := \frac{1}{|C_k^m|} \left[\left(|C_k^m| - 1 \right) \mathbf{g}_k^m + \mathbf{x}_i \right]$.
13: **end for**
14: **end for**
15: Let $\mathbf{h}^* :=$ BestAnt$(\mathbf{h}_1, \ldots, \mathbf{h}_M, \mathbf{h}^*)$.
16: For $i = 1, \ldots, n$ and $k = 1, \ldots, K$ let $\langle \Gamma \rangle_{ik} := \tau_{ik}$, where $\tau_{ik} := (1 - \rho) \ \langle \Gamma \rangle_{ik} + \rho \ \langle \Gamma_{aux} \rangle_{ik}$.
17: Intensify the best trail. For all $i (i = 1, \ldots, n)$, if individual i in $\mathbf{h}^*$ was classified in cluster k do $\langle \Gamma \rangle_{ik} = \langle \Gamma \rangle_{ik} + Q/W(\mathbf{h}^*)$
18: If the inertia of $\mathbf{h}^*$ improves the best inertia keeped in memory, reset Counter.
19: **for** $m := 1$ **to** M **do**
20: Apply K-means to $\mathbf{h}_m$.
21: Update $\mathbf{h}^*$ if there was an improvement from the K-means application.
22: **end for**
23: **end while**
24: **return** $\mathbf{h}^*$

5 Parameter Analysis

To develop the parameter analysis three data tables (T105, T525, and T2100) were built, with randomly generated normal variables. The data sets T105 ($n = 105$ and $p = 6$) and T525 ($n = 525$ and $p = 6$) consists of 105 and 525 objects, respectively. Both sets have seven clusters ($K = 7$), such that six classes have variance equal to $\sigma^2 = 1$, and the seventh class has $\sigma^2 = 3$. The data set T105 has a "big" class with 51 objects, and the remaining six groups with 9 objects. Meanwhile, T525 has a class with 265 objects, and the remaining objects are equitably distributed in the other groups. The $W(P)$ reference values for T105 and T525 were calculated using the Eq. (1), thereby 7.62467183 and 7.45610263 were obtained for these tables,

Algorithm 3 Local search strategy based on K-means applied in BACO.

Require: One ant $\mathbf{h}$.
1: PreviousInertia$\leftarrow -1$.
2: **while** $|\text{PreviousInertia} - W(\mathbf{h}_m)| > 0.001$ **do**
3: PreviousInertia$\leftarrow W(\mathbf{h}_m)$
4: For $\mathbf{h}$, build clusters $C_1, C_2, \ldots, C_K$, using barycenters $\mathbf{g}_1, \ldots, \mathbf{g}_K$. To do that, assign each individual $\mathbf{x}_i$ to the class with its barycenter closest to $\mathbf{x}_i$.
5: Recalculate the barycenters $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_K$ with:
$$\mathbf{g}_k = \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i, \text{ for all } k = 1, 2, \ldots, K.$$
6: **end while**
7: **return** A new ant $\widehat{\mathbf{h}}$.

respectively. Table T2100 has 2100 objects, seven clusters with the same cardinality and all classes have different variances. The $W(P)$ reference value for this set is 22.56959210.

The Algorithm 2 has four parameters that should be fitted, with the aim of achieving good performance. Parameters α and β control the relative weights assigned to pheromone concentration and ant visibility, respectively. Meanwhile, ρ represents the pheromone evaporation rate, used to update the pheromone matrix. Finally, parameter Q is a *pheromone amplification constant*.

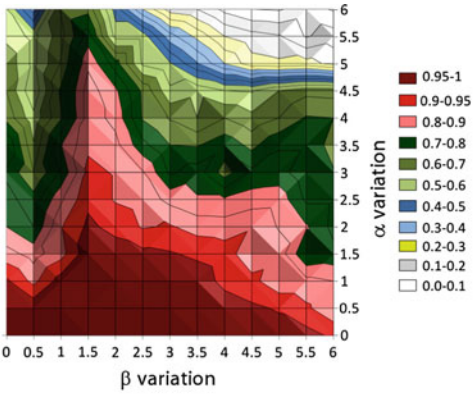
To develop the parameter analysis tables T105 and T525 were used, and for each table, and for each parameter combination, 200 multistart runs were done. Based on the ranges presented in Dorigo et al. [8], in the current experiment a further analysis was developed, using $\rho \in \{0.1, 0.2, \ldots, 0.9\}, \alpha, \beta \in \{0, 0.5, 1, 1.5, \ldots, 6\}$, and $Q \in \{50, 100, 150, \ldots, 500\}$. In total $9 \times 13 \times 13 \times 10 = 15210$ combinations were run for each table. This analysis used $M = 10$ (the number of ants).

The pictures in Fig. 1 show some examples of the 90 contour maps built with the performance percentages (each percentage represents how many times the algorithm scores the $W(P)$ reference value, in the 200 runs) obtained with table T105, for the different parameter combinations. For example, Fig. 1a shows the contour map for $\rho = 0.1$, $Q = 50$ and $\alpha, \beta \in \{0, 0.5, 1, 1.5, \ldots, 6\}$. This analysis showed that $\rho = 0.5$ was the best option, because the best performance zone for $\rho = 0.5$ (the darker red zone in Fig. 1b) is better (largest area) than those of the remaining ρ values.

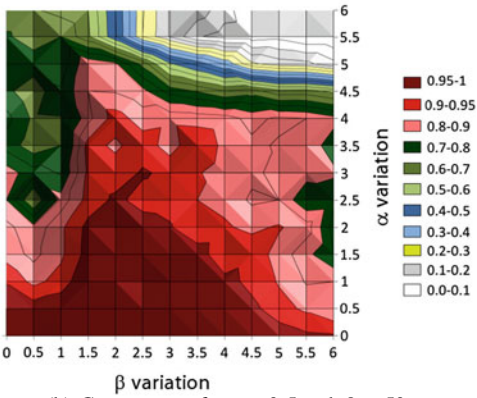
On the other hand, very similar contour maps were obtained when ρ was fixed, and Q varied from 50 to 500 (10 contour maps per each ρ value). This showed evidence that Q was not an important parameter in this experiment. And this coincides with the observation presented in Dorigo et al. [8], which indicates that Q has a negligible influence in the algorithm. Therefore, the parameter Q was fixed at 250 (the range middle value), but also could be fixed at 100, as they did.

Next, an analysis for α and β was developed with tables T105 and T525, using $\rho = 0.5$, $Q = 250$, and $\alpha, \beta \in \{0, 0.25, 0.5, 0.75, \ldots, 6\}$. Figure 2 shows the contour maps obtained in this process. This analysis was not enough to determine optimum values for α and β. Figure 2a and b only suggest that the best performance is probably obtained when $1.5 \leq \beta \leq 5$ and $0 < \alpha \leq 2.5$. For this reason, an extra analysis was

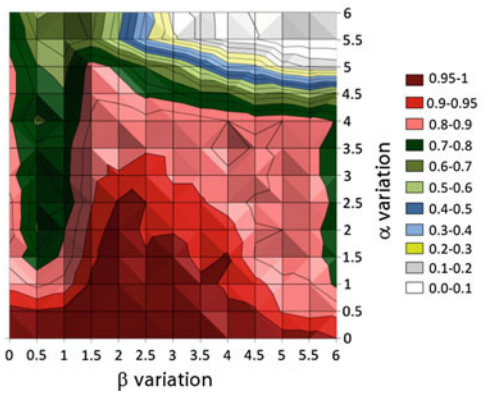
Fig. 1 Some examples of contour maps created with the performance percentages, for $Q = 50$, $\rho = 0.1, 0.5, 0.9$, and variants values for α and β. Analysis done with table T105



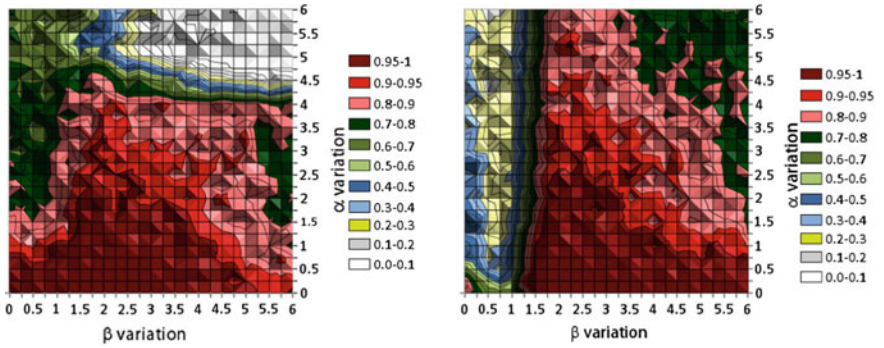
(a) Contour map for $\rho = 0.1$ and $Q = 50$



(b) Contour map for $\rho = 0.5$ and $Q = 50$



(c) Contour map for $\rho = 0.9$ and $Q = 50$



(a) Results obtained with table T105 (b) Results obtained with table T525

Fig. 2 Contour maps created with the performance percentages, with the fixed values $\rho = 0.5$ and $Q = 250$

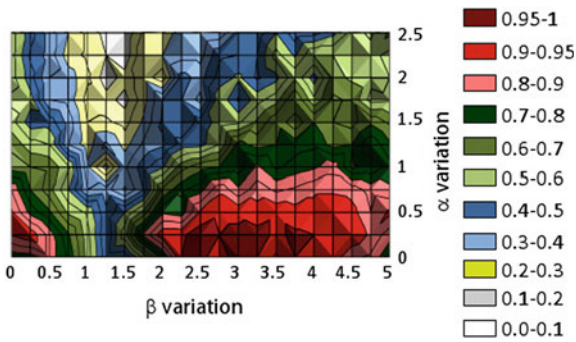


Fig. 3 Contour maps created with the performance percentages, with the fixed values $\rho = 0.5$ and $Q = 250$, in table T2100

developed with table T2100. Figure 3 shows that any combination for α and β in the dark red region could be taken. Therefore, for this experiment the combination $\beta = 2.5$ and $\alpha = 0.25$ was selected. Summarizing, the parameters were chosen as $\alpha = 0.25$, $\beta = 2.5$, $\rho = 0.5$, and $Q = 250$.

6 Extra Data Sets, Results, and Discussion

A personal computer with 8 GB of RAM memory and an Intel Core i7-4712MQ CPU@2.30 GHz processor, was used in this experiment. In order to develop a comparison among the algorithms BACO, BACOK and KM, five real-life data sets were downloaded from the website of UCI repository of machine learning databases (UCI [20]): iris ($n = 150$, $K = 3$ and $p = 4$), wine ($n = 178$, $K = 3$ and $p = 13$), glass identification ($n = 214$, $K = 6$, $p = 9$), red wine quality ($n = 1599$, $K = 3$, $p = 11$) and white wine quality ($n = 4898$, $K = 3$, $p = 11$) data sets. In *glass* data set the first attribute was not considered as a variable, because it is an identification number (for this reason $p = 9$). Furthermore, K was fixed at 6 because the type of glass number 4

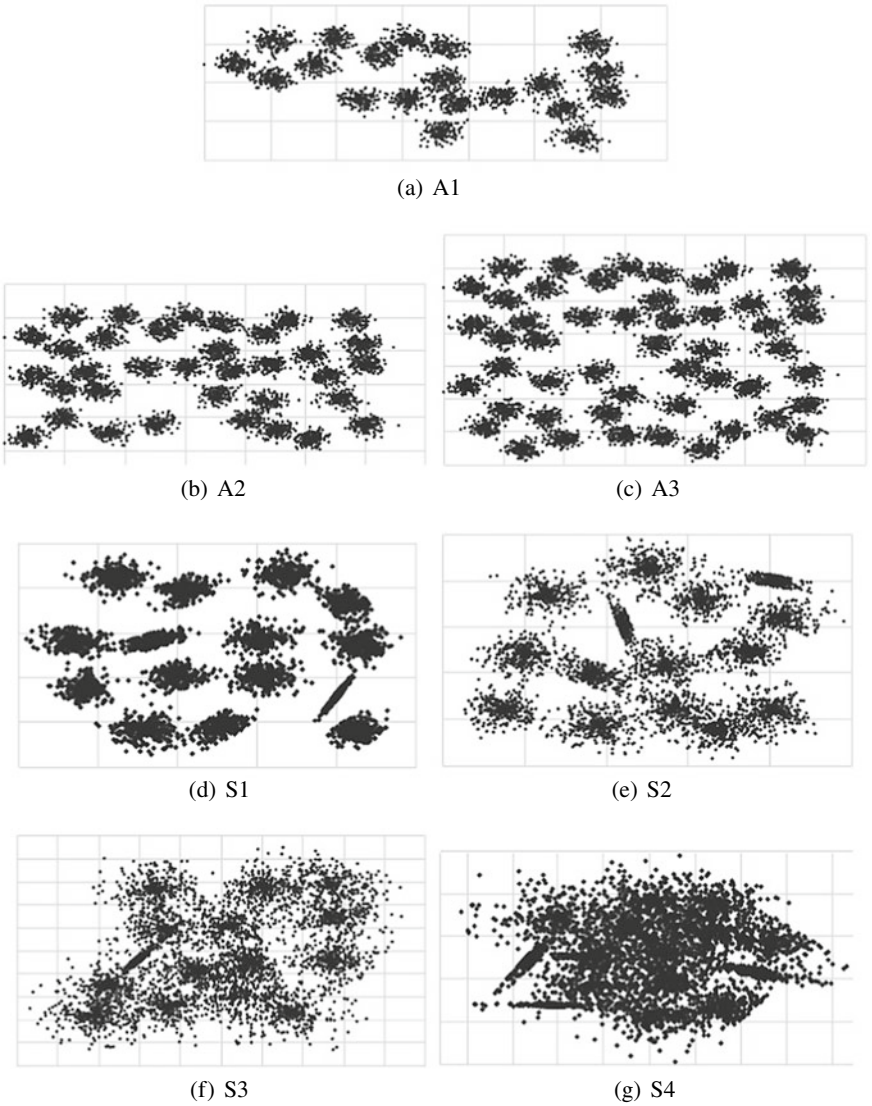
is not present in this data set (in total, there are 7 types of glass). In *wine quality* (both tables), the attribute number 12 was not considered because it is an output variable. Additionally, two groups (A and S) of bidimensional synthetic data sets were considered (downloaded from Fränti & Sieranoja [12]), which are described on Fränti and Sieranoja [11]. Group A (3 sets) varies the number of clusters, and the group S varies the overlapping among the clusters (4 sets). All cases use $p = 2$. Table 2 summarizes the main features of these sets and Fig. 4 shows a bidimensional representation for each set. Also, the *ground truth* centroids for these data sets are available on Fränti and Sieranoja [12]; hence, it was possible to analyze if the proposed BACOK algorithm was generating a reasonable clustering for the data. The *centroid index* (CI) presented in Fränti, Rezaei, and Zhao [10] is a cluste- level similarity measure, based on the cluster centroids, which can be used to compare one clustering against other solution or the ground truth, if is available. The algorithm BACOK was executed 100 times on sets A1, A2, A3, S1, S2, S3, and S4, and the best solution found, in each case, was compared with the ground truth solution, using the CI value. In all cases, the CI value was equal to zero, therefore according to Fränti and Sieranoja [11], our algorithm is properly clustering those datasets. This experiment was made with 20 ants ($M = 20$) and the parameters $\alpha = 0.25$, $\beta = 2.5$, $\rho = 0.5$, and $Q = 250$. Finally, using for each set the centroids of the best solution and the definition of $W(P)$ (see Eq. 1), the best within inertia for each set (W_{best}) was calculated (see column number 4 on Table 2).

Table 3 summarizes the results obtained with the three algorithms. The performance of each algorithm is represented by a percentage, and this corresponds to the number of times in which the algorithm scored the W_{best} value in 100 multistart runs. The algorithm BACO also used $M = 20, \alpha = 0.25, \beta = 2.5, \rho = 0.5$, and $Q = 250$. Meanwhile, the KM algorithm iterates until the difference between two consecutive within inertias is less than 0.001. The symbol "-" used in Table 3 means the algorithm did not attend the W_{best} reference value in any of the 100 runs. Also, the standard deviation of inertia, the average time and the standard deviation of time, in those 100 executions, are presented in Table 3.

Table 3 shows how the algorithm BACOK performed very good on the available data sets. This final comparison is valuable because it reinforces one of the princi-

Table 2 Main features for sets on group A ans S

Data set	n	K	$W(P)$ reference value
A1	3000	20	4048752.50
A2	5250	35	3864140.31
A3	7500	50	3858322.01
S1	5000	15	1783523123.37
S2	5000	15	2655821898.14
S3	5000	15	337791436.87
S4	5000	15	3140628447.25



(a) A1

(b) A2

(c) A3

(d) S1

(e) S2

(f) S3

(g) S4

Fig. 4 Two-dimensional representation for the datasets on groups A and B

Table 3 Performance comparison among BACO, BACOK, and KM

Data set	W_{best}	Performance-Standard deviation of inertia			Average time-Standard deviation of time (s)		
		BACO	KM	BACOK	BACO	KM	BACOK
Iris	0.52136	34% – 0.00104	5% - 0.19743	100% - 0	0.16467 - 0.05200	0.00027 - 0.00008	0.13550 - 0.02911
Wine	13318.48	8% - 41.3007	100% - 0	100% - 0	0.25580 - 0.08328	0.00048 - 0.00004	0.31380 - 0.04476
Glass	1.570377	-	-	93% - 0.0001	-	-	0.64427 - 0.17442
Red wine Q	247.2075	-	1% - 0.00717	100% - 0	-	0.00794 - 0.00072	3.28075 - 0.25986
White wine Q	560.4186	-	83% - 0.00022	100% - 0	-	0.03349 - 0.00778	16.59486 - 2.48081
A1	4048752.50	-	-	80% - 244433.51	-	-	14.84363 - 4.49093
A2	3864140.31	-	-	90% - 70488.07	-	-	62.97577 - 15.71993
A3	3858322.01	-	-	50% - 139284.45	-	-	164.00306 - 54.02449
S1	1783523123.37	-	-	100% - 0	-	-	11.94175 - 3.86356
S2	2655821898.14	-	-	100% - 0	-	-	12.65522 - 0.93671
S3	3377914369.87	-	-	87% - 1390.99	-	-	22.69879 - 5.31946
S4	3140628447.25	-	-	10% - 7221.77	-	-	30.60434 - 8.88218

pal contributions of this paper: the BACO and KM algorithms did not show good results in most all data sets, but our algorithm uses the potential of K-means to improve the algorithm BACO, and then significantly better results were obtained. That hybridization process presented in algorithm BACOK reveals how the K-means algorithm itself could not work well, but it can be used to improve other heuristic algorithms. Finally, the lowest performance reported by algorithm BACOK was in the set S4, which has the highest level of overlap (see Fig. 4).

7 Conclusions

We have presented a hybrid clustering method based on the ant colony optimization metaheuristic and the K-means algorithm. The method is based on some features developed for ACO in the traveling salesman problem and it is improved by the K-means algorithm in each iteration. The adaptation to the clustering problem takes into account the representation of clusters by barycenters, and therefore the distance between objects and barycenters is used for defining visibility and the pheromone trail.

After an extensive parameter fitting, an experimentation was implemented in order to evaluate the method. It performed very well, attaining the reference value for the inertia in each data table, in reasonable time. Furthermore, the method showed very good results when it was applied to other benchmark data sets, where the ground truth for each set was available.

Finally, the experiment revealed the parameter Q does not have a relevant role in the ACO algorithm, but the algorithm is very sensitive to the values assigned to the parameters α, β and ρ. The parameter fitting process was necessary to improve the algorithm performance and it gave the combination $\alpha = 0.25$, $\beta = 2.5$ and $\rho = 0.5$.

Acknowledgements J. Chavarría and J.J. Fallas were supported with project 5402-1440-3901 of the Research Vice-Rectory, Costa Rica Institute of Technology. J. Trejos was supported with project 821-B1-122 of the Research Vice-Rectory, University of Costa Rica.

References

1. Babu, G. P., & Murty, M. N. (1994). Clustering with evolution strategies. *Pattern Recognition*, *27*(2), 321–329. https://doi.org/10.1016/0031-3203(94)90063-9.
2. Bagirov, A. M. (2008). Modified global k-means algorithm for minimum sum-of-squares clustering problems. *Pattern Recognition*, *41*(10), 3192–3199. https://doi.org/10.1016/j.patcog.2008.04.004.
3. Barcos, L., Rodríguez, V. M., Álvarez, M. J., & Robusté, F. (2004). Routing design for less-than-truckload motor carriers using ant colony techniques. *Business Economics Series*, *14*, 4–38. https://doi.org/10.1016/j.tre.2009.11.006.
4. Bonabeau, E., Dorigo, M., & Theraluz, G. (1999). *Swarm Intelligence. From Natural to Artificial Systems*. New York: Oxford University Press. ISBN-13: 978-0195131598.

5. Dorigo, M., Birattari, M., & Stützle, T. (2006). Ant colony optimization: Artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, *1*(4), 28–39. https://doi.org/10.1109/CI-M.2006.248054.
6. Dorigo, M., Di Caro, G., & Gambardella, L. M. (1994). Ant algorithms for discrete optimization. *Artificial Life*, *5*(2), 137–172. https://doi.org/10.1162/106454699568728.
7. Dorigo, M. & Gambardella, L. C. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. on Evolutionary Computation*, *1*(1), 53–66. https://doi.org/10.1109/4235.585892.
8. Dorigo, M., Maniezzo, V. & Colorni A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, *26*(1), 1–13. https://doi.org/10.1109/3477.484436.
9. Everitt, B. S., Landau, S., Leese, M., & Stahl, D. (2011). *Cluster analysis* (5th ed.). Chichester: Wiley. https://doi.org/10.1002/9780470977811.
10. Fränti, P., Rezaei, M., & Zhao, Q. (2018). Centroid index: Cluster level similarity measure. *Pattern Recognition*, *47*(9), 3034–3045. https://doi.org/10.1016/j.patcog.2014.03.017.
11. Fränti, P., & Sieranoja, S. (2018a). K-means properties on six clustering benchmark datasets. *Applied Intelligence*, *48*(12), 4743–4759. https://doi.org/10.1007/s10489-018-1238-7.
12. Fränti, P., & Sieranoja, S. (2018b). *Machine Learning Repository*. Joensuu, Finland: University of Eastern Finland, School of Computing. http://cs.joensuu.fi/sipu/datasets/.
13. Gaertner, D., & Clark, K. (2005). On optimal parameters for ant colony optimization algorithms. In *Proceedings of the 2005 International Conference on Artificial Intelligence, 27–30 june 2005* (pp. 83–89) Las Vegas NV, USA. DBLP:conf/icai/GaertnerC05.
14. Handl, J., & Knowles, J. (2015). Nature-inspired clustering approaches. In C. Hennig (Ed.), *Handbook of Cluster Analysis* (pp. 419–439). London, UK: Chapman & Hall.
15. Handl, J., & Meyer, B. (2007). Ant-based and swarm-based clustering. *Swarm Intelligence*, *1*: 95–113. https://doi.org/10.1007/s11721-007-0008-7.
16. Jafar, O. M., & Sivakumar, R. (2010). Ant-based clustering algorithms: A brief survey. *International Journal of Computer Theory and Engineering*, *2*(5), 787–796. https://doi.org/10.7763/IJCTE.2010.V2.242.
17. Kao, Y., & Cheng, K. (2006). An ACO-based clustering algorithm. In M. Dorigo et al. (Eds.), *Ant colony optimization and swarm intelligence, Lecture Notes in Computer Science 4150* (pp. 340–347). Berlin, Germany: Springer. https://doi.org/10.1007/11839088_31.
18. Kennedy, J., & Eberhart, R. C. (2001). *Swarm intelligence*. San Francisco CA, USA: Morgan Kaufmann. ISBN 9781558605954.
19. Klein, R. W. & Dubes, R. C. (1989) Experiments in projection and clustering by simulated annealing. *Pattern Recognition* 22(2), 213–220. https://doi.org/10.1016/0031-3203(89)90067-8.
20. UCI (1998). UCI Machine Learning Repository. Irvine CA, USA: University of California, School of Information and Computer Science. Retrieved September 13, 2019, from http://archive.ics.uci.edu/ml.
21. Ng, M. K., & Wong, J. C. (2002). Clustering categorical data sets using tabu search techniques. *Pattern Recognition*, *35*(12), 2783–2790. https://doi.org/10.1016/S0031-3203(02)00021-3.
22. Runkler, T. A. (2005). Ant colony optimization of clustering models. *International Journal of Intelligent Systems*, *20*(12), 1233–1251. https://doi.org/10.1002/int.20111.
23. Sarkar, M., Yegnanarayana, B. & Khemani, D. (1997). A clustering algorithm using an evolutionary programming-based approach. *Pattern Recognition Letters 18*(10), 975–986. https://doi.org/10.1016/S0167-8655(97)00122-0.
24. Shelokar, P. S., Jayaraman, V. K., & Kulkarni, B. D. (2004). An ant colony approach for clustering. *Analytica Chimica Acta*, *509*(2), 187–195. https://doi.org/10.1016/j.aca.2003.12.032.
25. Stützle, T., López-Ibáñez, M., Pellegrini, P., Maur, M., Montes de Oca, M., Birattari, M., & Dorigo, M. (2012). Parameter Adaptation in Ant Colony Optimization. In Y. Hamadi, E.Monfroy & F. Saubion (Eds.), *Autonomous Search* (pp. 191–215). Berlin, Germany: Springer. https://doi.org/10.1007/978-3-642-21434-9_8.

26. Talbi, E. G. (2009). *Metaheuristics: From design to implementation*. Hoboken NJ, USA: Wiley.
27. Trejos, J., Murillo, A., & Piza, E. (1998). Global stochastic optimization techniques applied to partitioning. In A. Rizzi, M. Vichi & Bock, H. H. (Eds.), *Advances in data science and classification* (pp. 185–190). Berlin, Germany: Springer. https://doi.org/10.1007/978-3-642-72253-0_25.
28. Trejos, J., Murillo, A., & Piza, E. (2004). Clustering by ant colony optimization. In D. Banks, L. House, F. R. McMorris, P. Arabie & W. Gaul (Eds.), *Classification, clustering, and data mining applications* (pp. 25–32). Berlin, Germany: Springer. https://doi.org/10.1007/978-3-642-17103-1_3.
29. Wei, X. (2014). Parameters analysis for basic ant colony optimization algorithm in TSP. *International Journal of u-and e-Service, Science and Technology*, *7*(14), 159–170. https://doi.org/10.14257/ijunesst.2014.7.4.16.
30. Xavier, A. E., & Xavier, V. L. (2011). Solving the minimum sum-of-squares clustering problem by hyperbolic smoothing and partition into boundary and gravitational regions. *Pattern Recognition*, *44*, 70–77. https://doi.org/10.1016/j.patcog.2010.07.004.
31. Zhe, G., Dan, L., Baoyu, A., Yangxi, O., Xinxin, N., & Yang, X. (2011) An analysis of ant colony clustering methods: Models, algorithms and applications. *International Journal of Advancement in Computing Technology*, *3*(11), 112–121. https://doi.org/10.4156/ijact.vol3.issue11.15.