Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online
CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 49-12 (2016) 243–248

BAT-CLARA: BAT-inspired algorithm for Clustering LARge Applications

Yasmine Aboubi, Habiba Drias, Nadjet Kamel

LRIA , USTHB,
USTHB, BP 32 El Alia, Bab Ezzouar Algiers, Algeria
(e-mail: yasminaboubi@gmail.com, h_drias@hotmail.fr, nadjet.kamel@gmail.com)

`www.lria.usthb.dz`

Abstract: Bat algorithm is a new nature-inspired metaheuristic optimization algorithm introduced by Yang in 2010, especially based on echolocation behavior of microbats when searching their prey. Firstly, this algorithm is used to solve various continuous optimization problems. Clustering remains one of the most difficult challenges in data mining. In this paper, an overview of literature methods is undertaken followed by the presentation of a new algorithm called BAT-CLARA for clustering large data sets. It is based on bat behavior and k-medoids partitioning. The new technique is compared to the well-know partitioning algorithms PAM, CLARA, CLARANS and CLAM, a recent algorithm found in the literature. Experimental results show that, for the same tested datasets, BAT-CLARA is more effective and more efficient than previous clustering methods.

Keywords: clustering algorithms, bat-inspired algorithm, metaheuristics, medoids

1. INTRODUCTION

Data clustering is a fundamental problem in a variety of areas of computer science and related fields, such as datamining, data compression, statistical data analysis. It aims at gathering data into groups regarding to their similarity. One important target for such datamining task is to reduce the data size and hence ensures scalability. A cluster is a collection of objects which are similar between themselves and are dissimilar to the objects belonging to the other clusters. In other words, the goal of clustering is to distribute data into clusters such that the similarities among objects within the same cluster are maximal while similarities among objects from different cluster are minimal.

Clustering algorithms are generally classified as hierarchical and partitionning algorithms. In this paper, we are interested in partitioning algorithms, as they are widely used, Often the k clusters found by a partitioning method are of higher quality than the k clusters produced by hierarchical method (Raymond T and Han, 2002). The problem of partitioning n objects over k clusters is important as it has numerous applications in diverse domains. It is NP-hard and regarding this fact, a lot of efforts have been devoted to its resolution. As evoked previously, the objective is to put together the objects having similar characteristics in one cluster. We need therefore to define a similarity measure between the objects. An effective partitioning is such that the similarity between objects of the same cluster is maximal and the similarity between objects of different clusters is minimal. In order to facilitate these measures, clusters are represented by objects, generally expressed by some statistics such as means or medians. Concretely, from the optimization viewpoint, the issue consists in determining clusters such that to minimize the sum of the dissimilarities between each object and the representative of the cluster to which it belongs. This quantity, expressed by Formula (1), represents the absolute error, also called inertia of the partitioning. I is the sum of the absolute error for all objects in the data set, p is a given object in cluster C_j and o_j is the center or representative object of C_j, k being the number of clusters.

$$I = \sum_{j=1}^{k} \sum_{p \in C_j} d(p, o_j) \tag{1}$$

The data mining community has recently deployed a lot of efforts on developing fast algorithms for partitioning very large data sets. An overview of existing methods was elaborated in order to compare our proposal to the previous techniques. The most popular clustering algorithms are k-means (MacQueen (1967)), PAM (Leonard and Peter (1990)), CLARA (Leonard and Peter (1990)) and CLARANS (Raymond T and Han (2002)). CLAM (Nguyen, Quynh H. and Rayward-Smith, Victor J. (2011)) is a recent original approach exploring metaheuristcs for clustering.

Following the last mentioned research direction, we propose a new clustering algorithm based on Bat-inspired computing for Clustering LARge Application, namely Bat-CLARA. As its name suggests, the algorithm uses the Bat optimization technique to explore the solutions space in an efficient way. It also uses some necessary concepts of PAM to build clusters. Extensive experiments were performed on real datasets to validate the effectiveness and efficiency of the proposed algorithm relatively to the existing techniques of the literature.

This paper is structured as follows. Section 2 presents

2405-8963 © 2016, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2016.07.607

related works used the most by the scientific community as well as recent ones. Section 3 describes the metaheuristic inspired by Bats behavior and based on collective intelligence. The new clustering algorithm called Bat-CLARA is introduced in section 4. whilst section 5 provides the results of the experiments that were conducted for the validation of the proposal.

2. RELATED WORKS

The most known clustering techniques are k-means, k-medoids, CLARA and CLARANS. In this section, we describe their respective method as well as their benefits and disadvantages.

2.1 k-means

k-means is the most widely used clustering algorithm because of the simplicity of its implementation. It starts by drawing at random k centers then it assigns each objet to a cluster according to its distance with the cluster center. The means is calculated for each cluster once all the objects are inserted in the clusters and becomes the new center. This process is repeated until no changes occur in the centers. This algorithm is known to be not effective enough because of the representation of the center as the means of the objects residing inside the cluster. However it is efficient as it consists of one loop inside which, we dispatch the objects over the clusters and calculate the means with a simple formula.

2.2 PAM

k-medoids also called PAM (Partition Around Medoids) (Leonard and Peter (1990)) was designed to palliate to the k-means lack in effectiveness. It shares the same algorithmic structure but uses medoids as cluster representatives. The fact to substitute the means by the medoid makes the algorithm more effective because the medoid position in the cluster is central and the error value of Formula (1) decreases. Another benefit is the insensitivity to noise and outlier. However, it is less efficient.

2.3 CLARA

CLARA stands for **Cl**ustering **LAR**ge **A**pplication (Leonard and Peter (1990)). It is an improvement of PAM to handle large datasets. It runs PAM on multiple random samples, instead of the whole dataset. Experiments reported in (Leonard and Peter (1990)) indicate that five samples of size $(40+2k)$ give satisfactory results. It has been shown to produce relatively good quality solutions in a reasonable computation time for large data sets but it is less effective as it considers samples and not the entire datasets.

2.4 CLARANS

CLARANS (Ng and Han 1994) is proposed for **CL**ustering Large **A**pplications with **RAN**domized **S**earch in order to improve the effectiveness in comparison to CLARA. The algorithm uses a sampling technique to reduce the search space and the sampling is conducted dynamically for each iteration of the search procedure.
Conceptually, the clustering process can be viewed as a search through a graph $G_{n,k}$, where each vertex is a collection of k medoids $O_{m1}, ., O_{mk}$. Two nodes $S_1 = O_{m1}, , O_{mk}$ and $S_2 = O_{w1}, ..., O_{wk}$ are neighbors (that is, connected by an eadge in the graph) if their sets differ by only one object $|S1 \cap S2| = k - 1$. Each node can be assigned a cost that is defined by the total dissimilarity between every object and the medoids of its cluster. At each step, PAM examines all of the neighbors of the current node in its search for a minimum cost solution. As a consequence, the dynamic sampling used in CLARANS is more effective than the method used in CLARA in large application and more efficient than the swap phase of PAM in small application (Raymond T and Han (2002)). It outperforms then all the previous partitioning methods described previously.

2.5 CLAM

CLAM (Nguyen, Quynh H. and Rayward-Smith, Victor J. (2011)) stands for **C**lustering **L**arge **A**pplication using **M**etaheuristic). This algorithm uses a hybrid metaheuristic, combining Variable Neighborhood Search (VNS) and Tabu Search to guide the search to solve the problem of k-medoids clustering. Compared to CLARANS, experimental results show that, given the same computation times, CLAM is more effective. The only case where CLARANS outperforms CLAM is when both algorithms are set to perform a very small number of moves in the search space. In CLAM the search space is represented by a graph $G_{n,k}$, since each node has $k(n-k)$ neghboors. CLAM started with a randomly node in $G_{n,k}$; this search space is highly inter-connected, the number of shared neighbours between two neighbouring solutions is $(n2)$.
The revers VNS represents an intensification in the search strategy, the adoption of tabu list enforces the diversification of the search, CLAM employs a fixed a length of tabu liste firstin-first-out. Before a solution is considered, it is first checked against the tabu list. If the solution is not in the list, it is then visited and added to the list. If the solution is already in the list, it is discarded and the search can move on to another solution.

The first step in CLAM is to initialize a suitable value for the diameter (dim) of the datasets, defined as the largest distance between any tow data objects, so the suitable value for the radius range $[star\_radius, end\_radius]$ and the number of steps (t). For example a setting might be (50%, 40%, 30%, 20%, 10%) of dim, giving $start\_radius = dim/2$, $end\_radius = dim/10$ and $t = 5$. and $radius\_index = 1, 2, 3, 4, ...t$ as a pointer to the actual radius.

CLAM is described in Algorithm 1

Algorithm 1. CLAM

```
Input:
   k the number of clusters;
   D a set of n objects;
Output:
   a set of k clusters;
begin
```

```
 1. set start_radius, end_radiusproportiona
  to dim;
 2. set t the number of steps from
    start_radiusto end_radius ;
 3. set num_local to the number of required runs;
 4. set max_neighbour to the threshold
 5. cost(best_solution) := a very large number;
 6. tabu_list := empty;
 7. for i:=1 to num_local do
    7.1. set Current to a random solution;
    7.2. add Current to tabu_list;
    7.3. set radius_index to 1
         {i.e. radius := start_radius};
    7.4. while radius_index <=t do
         7.4.1. j:=1;
         7.4.2. while  j<=max_neighbour do
               1. select a neighbour of Current
                   solution, which must not be
                   in tabu_list;
               2. add neighbour to tabu_list;
               3. if cost(neighbour)<cost(Current)
                     then
                       Current:=neighbour;
                       j:=1;
                     else j:=j+1;
                  end if;
               end while;
         7.4.3. radius_index:=radius_index+1;
        end while
    7.5. if cost(best_solution)>cost(Current)
         then best_solution:=Current
        end if;
   end for;
end.
```

Swarm intelligence algorithms have been successfully applied to quite a lot of services, systems, and products frequently found in our daily life. It use an iterative search strategy to find an approximate optimal solution by using a limited computation resource, such as computing power and computation time (Tsai et al. (2014)). In (Marinakis et al. (2008)), the authors combined the greedy randomized adaptive search with Honey Bees Mating metaheuristic to solve the clustering problem. In another study (Tsutomu et al. (2010)), the authors proposed a new algorithm for clustering a large amount of dataset based on Ant Colony Optimization (ACO). The Ant Colony Clustering (ACC) algorithm uses artificial ants cooperating together to solve a problem of clustering. In another study, Tsai et al. (2013) presented an interesting review of eight kinds of recent metaheuristics that were developed for clustering during the last decade.

3. BAT ALGORITHM

Recently, a new metaheuristic was presented by Yang, Xin She. (2010) for optimizing numerical problems, called the Bat Algorithm or BA for short, which simulates the echolocation behavior of bats. Such technique has been developed to behave as a band of bats tracking prey/foods using their capability of echolocation.

3.1 Movement of Virtual Bats

In order to model this algorithm, Yang, Xin She. (2010) idealized some rules, as follows:

(1) It is assumed that all bats use echolocation to determine distances, and all of them are able to distinguish food, prey, and background barriers in some magical way.
(2) A bat b_i searches for a prey with a position x_i and a velocity v_i with a fixed frequency f_{min}, varying wavelength λ (or frequency f) and loudness A_0 to search for prey. Wave length k and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission r_i in the range of [0, 1], depending on the proximity of their target.
(3) Although the loudness can vary in many ways, Yang Yang, Xin She (2011) assume that the loudness varies from a large (positive) A0 to a minimum constant value A_{min}.

Algorithm 2 illustrates the pseudo-code for bat algorithm. First, the bat population, position x_i, velocity v_i and frequency f_i are initialized. At iteration t, the movement of virtual bats is updated through their velocity and position using Equations (2), (3) and (4).

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{2}$$

$$v_{ij}^t = v_{ij}^{t-1} + [x_{ij}^{t-1} - x^*]f_i \tag{3}$$

$$x_{ij} = x_{ij}^{t-1} + v_{ij}^t \tag{4}$$

$$\text{for all } i = 1..p \text{ and } j = 1..k$$

Eq (2) is used to control the pace and range of bats movements, where $\beta \in [0,1]$ denotes a randomly generated vector from uniform distribution, Eq (3) defines the velocity of decision held by bat i in time t and x^* represents the current global best location (solution), which is located after comparing all the solutions among all the n bats. Finally Eq (4) defines the position of decision held by bat i in time t.

Algorithm 2. Bat Algorithm

```
Input: empirical parameters: MaxIter, p;
Output: best solution;
begin
  1. Objective function
     f(x_{i})= f(x_{i1},..., x_{ik});
  2. Initialize the bat population
     xi and vi for (i = 1, 2,..., p );
  3. Define pulse frequency f_{i} at x_{i};
  4. Initialize pulse rates r_{i}
     and the loudness A_{i}
  5. while (t <MaxIter) do
   5.1. Generate new solutions by adjusting
      frequency, and updating velocities and
      locations/solutions using
      equations (2)to (4);
   5.2. if (rand > ri) then
     5.2.1. select a solution among the
            best ones;
     5.2.2. Generate a local solution around
     the selected best solution;
        end if;
    5.3. Generate a new solution
        by flying randomly;
    5.4. if (rand < Ai and f(xi) < f(x*)) then
       5.4.1. Accept the new solutions;
       5.4.2. Increase ri and reduce Ai;
       end if;
    5.5. Rank the bats and find the current
        best x*;
      end while
  6. Postprocess results and visualization;
end.
```

3.2 Loudness and Pulse Emission

Yang, Xin She. (2010) justified the performance of the BA compared to other metaheuristics. He proved that BA is able to outperform in terms of both improved local optima avoidance and convergence speed (Mirjalili, Seyedali and Seyed, Mohammad Mirjalili and Yang, Xin-She (2014)). In order to improve the variability of the possible solutions, a random walk procedure has been used to generate a new solution for each bat as follows:

if $(rand > r_i)$ then $X_{new} = X_{old} + \varepsilon A_t$
where $rand$ is a random number $\in [0, 1]$ and ε another random number $\in [-1, 1]$, while $A_t =< A_t >$ is the average loudness of all the bats at current generation. For each iteration of the algorithm, the loudness A_i and the emision r_i are updated if and only if the new solutions are improved, which means that these bats are moving towards the optimal solution. A solution is accepted if a number drawn at random is less then the loudness A_i and $f(xi) < f(x*)$, A_i and r_i are updated using rspectively Equation (5) and Equation (6).

$$A_i(t + 1) = \alpha A_i(t) \tag{5}$$

$$r_i(t + 1) = r_i(0)[1 - \exp(-\gamma t)] \tag{6}$$

Where α and γ are constants, α in $[0, 1]$ and $\gamma > 0$, α is similair to the cooling factor of a cooling schedule in the simulated annealing.

$$A_i^t \to 0, r_i^t \to r_i^0, ast \to \infty \tag{7}$$

Generally, the initial loudness A_i can typically be in $[1, 2]$, assuming $A_{min} = 1$ means that a bat has just found the prey and temporarily stop emittign any sound, while the initial emission rate $r_i(0)$ can be around zero, or any value $r_i \in [0, 1]$ if using. The choice of parameters equires some experimenting (Yang, Xin She. (2010)).

4. BAT ALGORITHM FOR CLUSTERING LARGE APPLICATION

In this section, we present Bat-CLARA, our proposed algorithm based on Bat algorithm for clustering LARge Application. CLARANS finds good clusters in randomized search. With Bat algorithm we can optimize the solution for better efficiency, so for finding the best set of k-medoids in BAT-CLARA we use the swarm intelligence of bat to guid the search. To adapt BA for clustering problem, we have to define the following components: the articial world where the virtual bats live, the neighborhood search, the bats position, the fitness function and the distance between two neighbors.

4.1 Artificial world where the virtual bats live

The search space is represented by the set of possible solutions, which are vectors of k elements. Its size is then equal to:

$n * (n - 1)... * (n - k + 1)$

The position x_i of each virtual bat defines a medoid (object) of a cluster C_i. To find the best set of k medoids $O_1, O_2, , O_k$ with the minimum cost, each bat adjusts its velocity by randomly selecting frequency f_i of sonic wave and each bat uses the loudness A_i and pulse emission rate r_i to control the intensive local search, which is processed to generate a new solution. In this study, the solution of k medoids found by the i^{th} bat is the global best solution $X^* = x_1^*, x_2^*, , x_k^*$.

4.2 The neighbor search

The neighbor search is represented by a subgraph denoted $G_{n,k}$ where n is the number of objects and k is the number of bats, which encapsulate solutions representing clusterings. Each node includes a set of k objects (a collection of k-medoids or positions of a bat). The subgraph contains $k(n - k)$ nodes highly inter-connected and two nodes are neighbors if and only if the cardinality of their intersection (respective positions of their bats) is equal to $(k - 1)$ that is, if they differ by only one position.

4.3 updating the movement equations

the bats movement is calculated by the three previous equation (2), (3) and (4). Each artificial bat uses equation (2) to select a frequency f_i in the range of frequency $[f_{min}, f_{max}]$, where f_{min} and f_{max} are two integers in the range $[1, n]$. It also uses equation (3) to update its velocity. The resulting value is applied in the third equation (4) to calculate the next position.

4.4 The fitness function

The fitness function is defined to be the total dissimilarity between every object and the medoids of its cluster. It is defined in (1).

4.5 The distance between two neighbors

The cost differential between two neighbors S_1 and S_2 is given by TC_{ij} in equation(8), where O_i, O_j are the difference between S_1 and S_2 (i.e. O_i and O_j not in $S_1 \cap S_2$, but $O_i \in S_1$ and $O_j \in S_2$)(Raymond T and Han (1994)).

$$TC_{i,j} = \sum_i C_{hji} \tag{8}$$

C_{hji} is defined by one of the equation below, where O_h denotes other non-medoid objects that may or may not need to be moved and O_{h2} denotes a current medoid that is nearest to O_h.

(1) **Case 1:** O_h currently belongs to representative object O_i and $d(O_h, O_{h2}) < d(O_h, O_j)$:
$$C_{hij} = d(O_h, O_h 2) - d(O_h j, O_i) \tag{9}$$
(2) **Case 2:** O_h currently belongs to representative object O_i and $d(O_h, O_j) < d(O_h, O_{h2})$:
$$C_{hij} = d(O_h, O_j - d(O_h, O_i) \tag{10}$$
(3) **Case 3:** O_h currently belongs to representative object $O_h 2$ and $d(O_h, O_{h2}) < d(Oh, Oj)$:
$$C_{hij} = 0 \tag{11}$$
(4) **Case 4:** O_h currently belongs to representative object O_{h2} and $d(O_h, O_j) < d(O_h, O_{h2})$:
$$C_{hij} = d(O_h, O_j) - d(O_h, O_{h2}) \tag{12}$$

Bat-CLARA is outlined in Algorithm3.

Algorithm 3. Bat-CLARA

```
Input:
 k the number of clusters;
 D a set of n objects;
 Empirical parameters: MaxIter, p;
Output:
 a set of k clusters;
begin
 1. Initialize the size of bat population to p;
 2. Generate a random starting position
    xi=(xi1, x2i, ..., xki}) for each bat i;
 3. Initial the loudness Ai in [1.0,2.0]
    and the emission rates ri in[0.0,1.0]
 4. Define pulse frequency fmin and fmax
    (fmin<fmax) as integer in [1,n]
 5. While (t<MaxIter) do
    5.1. Generate a new solution by adjusting
         frequency and updating velocity and
         position using equations (2),(3)and(4);
    5.2. If (rand<ri)
         5.2.1. Neighbor search (xt);
         5.2.2. Select best solution ;
         End if;
    5.3. Evaluate the new solution
         (fitness=f(xt));
    5.4. Generate a new solution by
         fluing randomly
    5.5. If(rand<Ai and fitness<f(x*)) then
         5.5.1. Accept the new solution;
         5.6.2. Increase ri and reduce Ai
               according to equations (6)and(7);
         End if;
    5.6. Rank the bats and find
         the current best X*;
        End while
 6. Return x*
   end.
```

5. EXPERIMENTAL RESULTS

Bat-CLARA was implemented in java NetBeans 7.1, on a personal computer with an Intel R Core(TM) i7-4500U CPU (2.40 GHz) with 8GB RAM. The numerical tests were performed on two benchmarks available from the UCI machine learning repository

5.1 Description of the Benchmarks

The first dataset called CONCRETE (Yeh (2007)) includes 1030 instances, each with 10 attributes, among them eight input attributes, which are cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, fine aggregate and age. We run bat-CLARA algorithm to cluster the dataset instances into 10 clusters.
The second dataset is the Wisconsin Breast Cancer database (WIlliam H (1992)). The benchmark contains 699 instances, each object characterized by nine attributes, which are clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelialcell size, bare nuclei, bland chromatin, normal nucleoli and mitoses. The instances belong to one of the two possible classes: benign or malignant.

5.2 Experimentations and Results

In order to evaluate Bat-CLARA in practice, we compare its performance with that of different k-medoids clustering techniques, using the dataset mentioned previously that is, CONCRETE and Wisconsin Breast Cancer. Five algorithms were implemented for these experiments: Bat-CLARA, CLAM, PAM, CLARA and CLARANS. Bat-CLARA, CLAM and CLARANS are tested with different number of neighbors:5%, 10%, 50% and 100% of k(n-k). Experimental results are presented in Figures 1, 2 and 3. We remark that Bat-CLARA outperforms the four other techniques. For Wisconsin dataset, Bat-CLARA outperforms all techniques with 10% of neighbors. In CONCRETE dataset, it reaches tha same paerformance as CLAM and CLARANS with only 20% of neighbors whereas tha latter did with 100% neighbors.

Table 1. Setting parameters for CONCRETE Dataset

Parameter	Value in CONCRETE
Population size : k	2
Emission rate r_i	r_i in $[0.0, 1.0]$
Loudness A_i	A_i in $[1.0, 2.0]$
Minimal frequency f_{min}	1, $(1 \leq f_{min} < n)n = 1030$
Minimal frequency f_{max}	1030

Table 2. Setting parameters for Wisconsin Breast Cancer

Parameter	Value in W.B.C
Population size : k	10
Emission rate r_i	r_i in $[0.0, 1.0]$
Loudness A_i	A_i in $[1.0, 2.0]$
Minimal frequency f_{min}	1, $(1 \leq f_{min} < n)n = 699$
Minimal frequency f_{max}	699

The second experiment result, figure 3 depicts the optimal runtime for the best techniques. It is clear that BAT-CLARA is the most efficient for the Wisconsin dataset. In the third series of experiments use the data set Wisconsin. We fix the number of MaxIter at 5, then we vary the number of neighbors at 5%, 10%, 20%, 50% and 100% of $k(n - k)$. The average cost depends on the number of neighbors. Figure 1 illustrate the performance of BAT-CLARA with respect to these parameters. We notice that

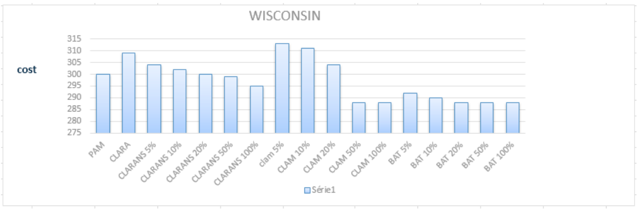

Fig. 1. Comparing different k-medoids clustering algorithms using Wisconsin dataset

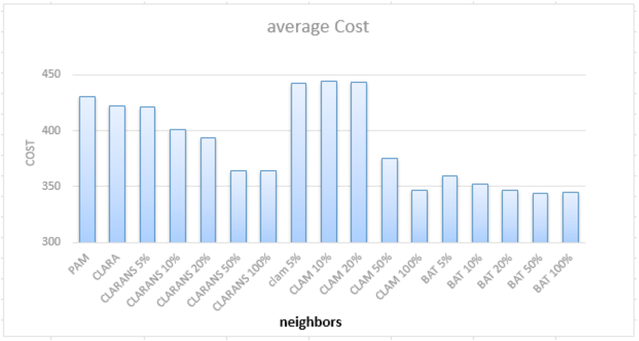

Fig. 2. Comparing different k-medoids clustering algorithms using CONCRETE dataset

the fitness decreases with the increase number of bees. The same remark is observed for the number of neighbors.

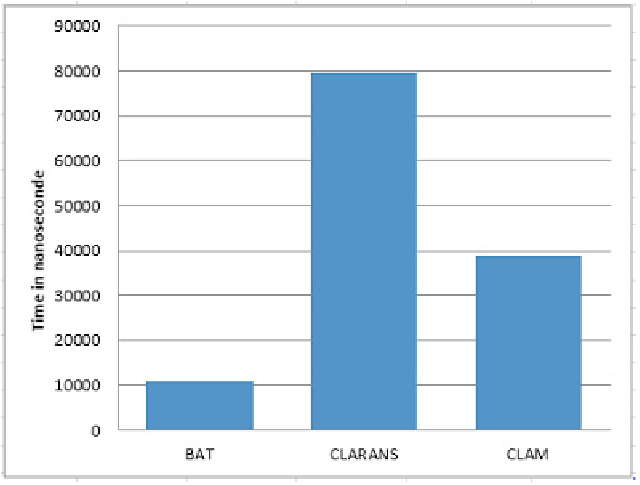

Fig. 3. Comparing the runtime for the best k-medoids algorithms using The Wisconsin dataset

6. CONCLUSION

In this work, a Bat-inspired Algorithm, which a robust optimization technique is design for clustering large datasets, namely Bat-CLARA. For the optimization aspect, it is based on the behavior of real Bats especially for their more attractive collective intelligence. It uses also k-medoids clustering for building the clusters. Its performance was compared with those of PAM, CLARA, CLARANS, the most known classic methods and CLAM a recent approach using metaheuristic. The computational results revealed that BAT-CLARA outperforms all compared algorithms to solve the clustering problems.

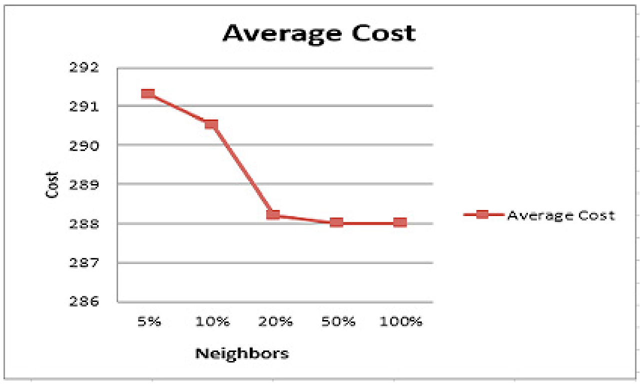

Fig. 4. neighbors vs. Cost

REFERENCES

Leonard, K. and Peter, J. (1990). Finding groups in data: an introduction to cluster analysis.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1, 921–926. 14.

Marinakis, Y., Magdalene, M., and Nikolaos, M. (2008). A hybrid clustering algorithm based on honey bees mating optimization and greedy randomized adaptive search procedure. *Learning and Intelligent Optimization. Springer Berlin Heidelberg*, 138–152.

Mirjalili, Seyedali and Seyed, Mohammad Mirjalili and Yang, Xin-She (2014). Binary bat algorithm. *Neural Computing and Applications*, 25, 663–681. 3-4.

Nguyen, Quynh H. and Rayward-Smith, Victor J. (2011). CLAM: Clustering Large Applications Using Metaheuristics. *J. Math. Model. Algorithms*, 10, 57–78.

Raymond T, N. and Han, J. (1994). Efficient and Effective Clustering Methods for Spatial Data Mining. *Very Large Data Bases (VLDB94)*, 144–155.

Raymond T, N. and Han, J. (2002). Clarans: A method for clustering objects for spatial data mining. *Knowledge and Data Engineering, IEEE Transactions*, 14, 1003–1016. 5.

Tsai, C.W., Huang, W.C., and Chiang, M.C. (2013). In J.J. Park, H. Adeli, N. Park, and I. Woungang (eds.), *MUSIC*, volume 274 of *Lecture Notes in Electrical Engineering*, 629–636. Springer.

Tsai, C.W., Huang, W.C., and Chiang, M.C. (2014). Recent development of metaheuristics for clustering. In *Mobile, Ubiquitous, and Intelligent Computing*, 629–636.

Tsutomu, S., Fumihiko, Y., and Yoshiaki, T. (2010). A new algorithm based on metaheuristics for data clustering. *Zhejiang University SCIENCE A ISSN*, 12, 921–926.

WIlliam H, W. (1992). UCI Repository of Machine Learning Databases. *University of California.*

Yang, Xin She. (2010). A new metaheuristic bat-inspired algorithm. *Nature inspired cooperative strategies for optimization (NICSO 2010), Springer Berlin Heidelberg,*, 65–74.

Yang, Xin She (2011). Bat algorithm for multi-objective optimisation. *International Journal of Bio-Inspired Computation*, 3, 267–274. 5.

Yeh, I.C. (2007). UCI Repository of Machine Learning Databases. *University of California.*