Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 61 (2015) 38 – 45

Procedia Computer Science

Complex Adaptive Systems, Publication 5
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2015-San Jose, CA

Clustering Quality Improvement of k-means using a Hybrid Evolutionary Model

Jeyhun Karimov, Murat Ozbayoglu*

TOBB University of Economics and Technology, Department of Computer Engineering, Ankara, 06560, Turkey

Abstract

Choosing good candidates for the initial centroid selection process for compact clustering algorithms, such as k-means, is essential for clustering quality and performance. In this study, a novel hybrid evolutionary model for *k*-means clustering (HE-*k*means) is proposed. This model uses meta-heuristic methods to identify the "good candidates" for initial centroid selection in *k*-means clustering method. The results indicate that the clustering quality is improved by approximately 30% compared to the standard random selection of initial centroids. We also experimentally compare our method with the other heuristics proposed for initial centroid selection and the experimental results show that our method performs better in most cases.

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology

Keywords: clustering; *k*-means; cluster-centroids; PSO; Simulated Annealing; Scatter Search; hybrid model; data mining;

1. Introduction

Data clustering is the process of grouping similar multi-dimensional data vectors together into a number of partitions. The aim is to identify and classify objects into subsets that have some meaning in the context of a particular problem. A detailed discussion on clustering algorithms can be found in [1].

k-means is one of the most popular and simplest algorithms among iterative and hill climbing clustering algorithms for unsupervised clustering [2]. However, its performance is highly dependent on its initial starting points (centroids) and converges to the nearest local optima without searching the whole solution space. The algorithm selects the initial solutions (cluster centroids) randomly. Due to its greedy nature, the result of clustering can differ quite a bit based on its randomly selected initial solutions.

In this paper, we propose a model to overcome this particular issue of properly choosing the initial centroids for compact clustering problems, specifically *k*-means algorithm. The proposed solution is a hybrid model of PSO (Particle

* Corresponding author. Tel.: +90-312-292-4073; fax: +90-312-292-4180
E-mail address: mozbayoglu@etu.edu.tr

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology
doi:10.1016/j.procs.2015.09.143

Swarm Optimization) [3,4], SA (Simulated Annealing) [5], SS (Scatter Search) [6], *k*-means [2] and some other heuristics. We named it as HE-*k*means (Hybrid Evolutionary Methods and *k*-means) clustering.

This paper is organized as follows. Previous work and methodologies are covered in section 2. Our proposed model is explained in detail in section 3. In section 4 we present the experimental results comparing our method with the previous work along with the details about the specific data sets we used. We conclude in section 5.

2. Previous work

Niknam et al. have proposed a hybrid model based on PSO, ACO (ant colony optimization) and *k*-means [7]. Their model is mainly concerned with combining ACO, PSO and *k*-means to eliminate each other's drawbacks. The proposed algorithm uses the random selection procedure of the ACO algorithm to assign different global best positions to every distinct agent. In another study, a model has been developed by hybridizing PSO and *k*-means [8][9]. For that model, the authors proposed PSO to find the centroids of a user specified number of clusters. The algorithm was then extended to use *k*-means clustering to seed the initial swarm. Yet in another work the authors proposed PSO-clustering without any other combination [10]. Here PSO-clustering, *k*-means and fuzzy *c*-means [11] were compared and the best results were obtained with PSO-clustering. Arthur et al. proposed a model called *k*means++ to avoid poor clustering formed by the standard k-means algorithm [12].

Genetic algorithms influenced clustering research and some noteworthy contributions are made earlier [13][14]. Genetic *k*-means algorithm is proposed to enhance the standard *k*-means [13]. The main idea is again to avoid being stuck in local optima. Here authors define a biased mutation operator specific to clustering, called distance-based-mutation. They demonstrated that the proposed model converges to the best-known optimum corresponding to the given data in concurrence with the convergence result. Another genetic clustering algorithm is proposed by exchanging neighboring centers for *k*-means clustering and tested on large simulated data sets [14]. In [15], an SA algorithm performs the selection of initial seeds and the clustering problem is reformulated from the beginning.

Another work by Bradley et al. to get "better" local optima for clustering uses the refined initial starting points [16]. They choose some sample points from the data and calculate refined initial starting points. As a result, they get better results compared to standard *k*-means. Huang et al. on the other hand, proposed a modified version of standard *k*-means in order to get rid of local optima problem [17]. The method iteratively updates the weights of variables with the formula authors proposed. The weights indicate the importance of variables, which is essential when sampling and making variable selection over large scale data.

Another work using Simulated Annealing (SA) in clustering is [18]. They used experiments to discover word classes of Chinese *n*-gram model. They showed that proposed model achieves much better results than dictionary-based models. Scatter Search (SS) algorithm was also used in clustering problems [19]. SS is applied to the capacitated clustering problem. Since *k*-means is highly sensitive to initial centroids, the new modified version, global *k*-means, was developed with an incremental algorithm that dynamically adds one cluster center at a time and uses each data point as a candidate for the k^{th} cluster center [20]. Bagirov et al. modified global *k*-means such that the starting point for the k^{th} cluster center is computed by minimizing an auxiliary cluster function [21].

3. Proposed model

In this study a new hybrid clustering model, namely HE-*k*means, is proposed to improve the clustering quality. We used PSO, SA, SS and some other heuristic algorithms integrated together to implement a better performing hybrid model eliminating each other's shortcomings and promoting advantages. For example, PSO algorithm may be trapped into local optima if the global best and local best positions are equal to the particle's position over a number of iterations. Therefore, SA is combined with PSO. Moreover, there are many cases that new particles can be constructed out of existing ones with better fitness values. Therefore, SS part is combined to our model. In our algorithm, PSO, Elimination, SA, and SS are run in this order for a number of iterations (t), and then the obtained best solution ($gBest$), which is a particle consisting of the best initial centroids achieved with the algorithm, is passed to *k*-means so that it can converge faster. In the PSO step, firstly, particles are constructed randomly and after processing all data points, each particle's similarity matrix and fitness values are calculated. Then, the particles with weaker fitness values are eliminated with w_e probability (1), where w_e is probability, f_p^t is fitness value of p^{th} particle at t^{th} iteration and f_p^{t-1}

is the one at *(t-1)*th iteration. Next, the SA algorithm is applied to the particles to change their conventional directions. After this step, the SS step begins. Here the best particle elements, namely centroids, are selected from different particles with the condition that they do not have any common data points. From these centroids, new particles are constructed and the best ones are added to swarm. At the end, *gBest*, the global best particle, is decomposed into centroids and is given as the initial solution to *k*-means.

3.1. PSO Module

In PSO step the particles are constructed in a way that if k is the number of desired clusters, the random centroids are generated as $c_0, c_1 \dots c_k$ for each particle and they are combined or concatenated. After initializing all particles, for each data point the nearest centroid from all particles are calculated. Let *E[p][k][n][m]* denote the similarity array which contains shared data point count between p^{th} particle's k^{th} centroid with n^{th} particle's m^{th} centroid. This four dimensional array keeps the shared data point counts for each pair of centroids in separate particles. That is, if *E[p][k][n][m] = c*, it means that from the data points belonging to p^{th} particle's k^{th} centroid, c of them also belong to n^{th} particle's m^{th} centroid. If *E[p][k][n][m] = 0*, then p^{th} particle's k^{th} centroid and n^{th} particle's m^{th} centroids do not share any data points. In this step, array E is computed and each particle's and its centroids' fitness values are calculated using (2) and (3) respectively, where f_p is the fitness value of p^{th} particle, f_p^k is the fitness value of p^{th} particle's k^{th} centroid, x_i^j is the data point belonging to j^{th} centroid, c_p^j is j^{th} centroid of pth particle and w_f is some constant. The output of this part is the array E and the fitness values of all centroids that form the particle. The pseudo code for this part is given in Figure **1**.

$$w_e = \begin{cases} 1, \ \ if \ \left(f_p^t - f_p^{t-1}\right) < 0 \\ \dfrac{-\left|f_p^t - f_p^{t-1}\right|}{T}, otherwise \end{cases} \tag{1}$$

$$f_p = \frac{\sum_{j=1}^{k}\sum_{i=1}^{n}\left|x_i^j - c_p^j\right|^2}{w_f} \tag{2}$$

$$f_p^k = \frac{\sum_{i=1}^{n}\left|x_i^j - c_p^k\right|^2}{w_f} \tag{3}$$

$$v_i = v_i + w_p * r_p * \left(p_i^p - x_i\right) + w_g * r_g * (g - x_i) \tag{4}$$

$$l_i = l_i + v_i \tag{5}$$

$$v_i = -(v_i + \ w_p * r_p * \left(p_i^p - x_i\right) + w_g * r_g * (g - x_i)) \tag{6}$$

$$DP\left(k, c_p^k\right) = argmin(DP\left(k-1, n_p^k\right) + f_p^k, DP(k, n_p^k))\forall(n_p^k) \tag{7}$$

3.2. Elimination module

Within this module, similar particles are eliminated. By similarity measure, the upper threshold of the shared data points between different centroids of particles is selected. It is obvious that all particles share same points among themselves. However, different centroids of different particles share different amount of points between each other. Let, w_t denote the threshold value, where $0 < w_t < 1$. If particles are similar, then their different centroids share data points that are above the threshold. So, if particles p_1 and p_2 are similar, then there exists some one-to-one mapping between the centroids of p_1 and p_2 such that either $|c_1^0| > w_t * |c_2^0|$ & $|c_1^1| > w_t * |c_2^1| \dots \left|c_1^k\right| > w_t * \left|c_2^k\right|$ or $|c_2^0| > w_t * |c_1^0|$ & $|c_2^1| > w_t * |c_1^1| \dots \left|c_2^k\right| > w_t * \left|c_1^k\right|$, where $|c_n^k|$ denotes the number of elements that belong to that centroid. In order to achieve this, the similarity array E is used. Therefore, iterating through E array it is obvious to detect similar particles. After detecting such particles, elimination is performed with probability we. The similar

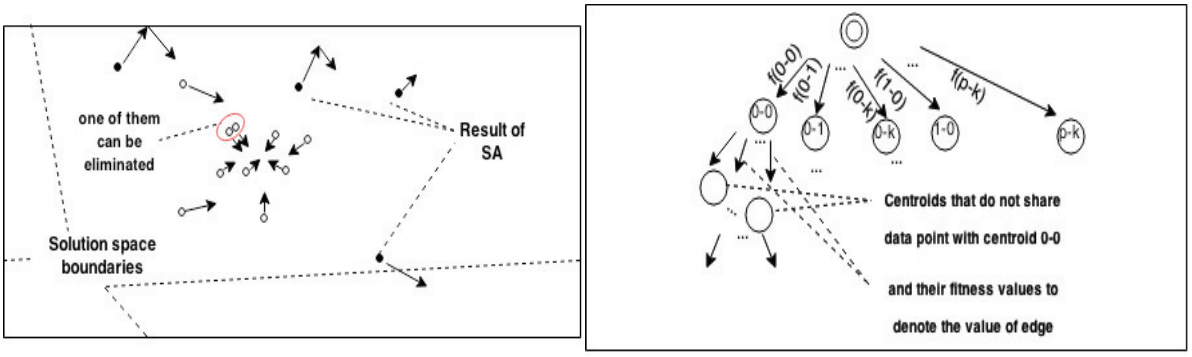
particles are not eliminated directly because in the next step these particles can have different locations due to the nature of the SA algorithm. As can be seen from Figure 2(a), there are two particles highlighted in the circle, which are similar, and therefore share some amount of data above the threshold w_t.

Input: $S=\{x_0, x_1 \dots x_n\}$ set of all data points and K value indicating number of clusters.

Output: E and F, similarity matrix and fitness values

1. $p_i.location \leftarrow random-location, \forall i, where\ p_i \in P$
2. $p_i.velocity \leftarrow random-velocity, \forall i, where\ p_i \in P$
3. $E[p][k][p][k] = 0\ \forall(p,k), where\ 0 \le p < |P|\ and\ 0 \le k < K$. Initialize similarity array
4. $f_i^k = \infty, \forall(i,k), where\ 0 \le i < |P|\ and\ 0 \le k < K$. Assign all particle's centroid fitness values to infinity
5. $for\ all\ x_i \in S, do$:
   - 5.1. Let c_p^k be p^{th} particle's nearest centroid to x_i
   - 5.2. Let *M[p][k]* be two dimensional array to denote all centroids that c_p^k for x_i belongs to. That is, if *M[p][k] = 1*, then x_i belongs to p^{th} particle's c^{th} centroid
   - 5.3. $M[p][k] = 0\ \forall(p,k), where\ 0 \le p < |P|, 0 \le k < K$
   - 5.4. $for\ all\ p_j \in P\ do$:
     - 5.4.1. Find c_j^k, the nearest centroid to x_i from particle p_j
     - 5.4.2. $M[j][k] = 1$. Denote previous step in array M
     - 5.4.3. $f_j^k = f_j^k + \frac{d^2}{w_f}$, where f_j^k is fitness value of k^{th} centroid of j^{th} particle, d is distance between x_i and c_j^k and w_f is some constant
   - 5.5. $for\ all\ m, n$ where $M[n][m] = 1, do$:
     - 5.5.1. $E[n][m] \leftarrow E[n][m] + M$. In the right hand side, '+' is element-wise summation
6. Output $f_j^k \in F, \forall(k,j)$ and the four dimensional similarity array E

Figure 1. Pseudo code for PSO module**.**



(a) (b)

Figure 2.(a) Simulation of SA and Elimination parts in 2 dimensions, particles with bold circles are directed 180° back as in SA, others are directed to *gBest* as in PSO and two particles highlighted in circle are similar ones. (b) Demonstration of tree needed to reconstruct new particles. Here we have two particles (0 and 1), consisting of three centroids, that is, with k=3, where f (p-k) denotes the fitness value of p^{th} particle's k^{th} cluster.

3.3. SA module

This part of the model is related with Simulated Annealing (SA) algorithm [11]. After eliminating similar particles, it is required to determine the particles' next iteration locations and velocities. In this part, for each particle, their previous fitness values and current ones are compared. If the current fitness values are lower, it is accepted because lower fitness value is better. Otherwise, it is accepted with some probability. The exact equation is described in (1). So, if $p = 1$, the current fitness value is lower than the previous, then particle's next velocity and location is calculated using (4) and (5), where v_i is the velocity of i^{th} particle, w_p is constant weight factor to personal best direction, w_g is the constant weight factor to global best direction, r_p and r_g are some random values between 0 and 1, p_i^p is the

personal best location of the particle p_i, g is the global best location and l_i is the new location of particle p_i. On the other hand, for a particular particle, if the probability is not accepted according to equation (1), then the next location of that particle is calculated using equations (6) and (5) respectively. That is, the velocity of the particle is calculated as in (4), but is multiplied with (-1) to change its direction $180°$ to go in the opposite way. Then, the new location is calculated using (5).

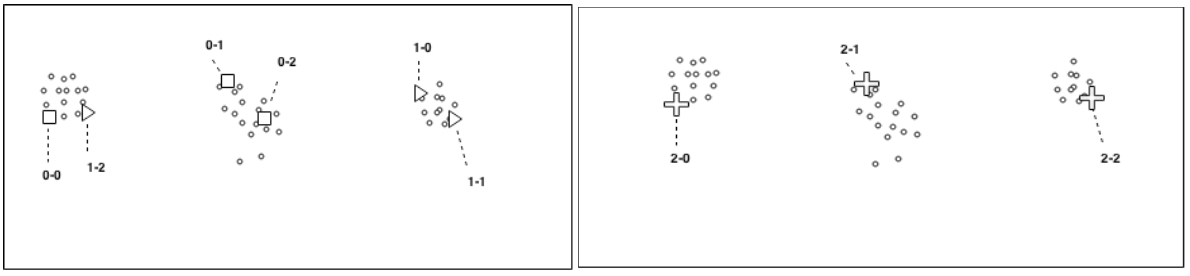
3.4. Scatter Search module

Within this module, *k* different centroids are found and joined to construct a new particle. The important point is that, k different centroids cannot share any data points, since in a particle; centroids do not have any common data points.

Figure **3** shows simulation of this part with *k=3* and *|P|=2*. That is, Figure **3**(a) shows the situation before the SS part. Here, one particle is denoted with rectangle and another with triangle. SS part computes that centroids denoted with *0-0*, *1-1* and *0-1* do not share any common data points, where *p-k* means k^{th} centroid of p^{th} particle. As a result, the new particle can be created as shown in Figure **3**(b). Indeed, it can be seen that the constructed particle have better fitness value than the existing ones.

We use the similarity array *E* to form a graph consisting of vertices denoted with centroids and edges having value of that vertex's fitness value. Since the similarity array keeps the sharing data point count between centroids, if the shared element count is zero, then the corresponding centroids can be concatenated. As can be seen from Figure 2(b), the source node is the double lined circle. It has $|P| * k$ child nodes, since that is precisely the total number of centroids within the swarm. The source node does not share any points with any of the centroids. The aim is, to find some path that consists of *k* consecutive nodes, with the minimum sum of edge values, where the incoming edge value of a particular node is its fitness value. The important point is that, if *Node-A* has a set of non-sharing clusters as *S*, in any of its child nodes, the set of their non-sharing clusters must be subset of *S*, the non-sharing cluster set of their parent node. In that way, *k* consecutive nodes cannot share any data points between themselves.

As can be seen from (7), finding the minimum length *k* united nodes is implemented recursively, where n_p^k is the child node of c_p^k. The function *DP* takes two arguments, the number of nodes needed and the current node. Finding the minimum path length consisting of *k* nodes can be done by either taking the current node into consideration, running the same function for the rest of its child nodes with *(k-1)* needed nodes and summing them, or not taking the current node into the consideration and running the function with *k* and all of current node's child nodes arguments, and giving the minimum of this results as an output. The important point here is that, the use of memoization is crucial in order to get the advantages of dynamic programming. That is, if some value of $DP(k,n_p^k)=c$, then it is stored. When it is needed, then the answer is retrieved and used in constant time. After the new particles are constructed, best of them are selected according to the sum of centroids' fitness values and is given to standard *k*-means as an input.



(a) (b)

Figure 3. (a) Particles before *SS* part (*0* and *1*), denoted with rectangle and triangle. Here, *k = 3*, and *p − k* notation stands for p^{th} particle's k^{th} centroid. (b) Location of a newly created particle by *SS* part, where new particle's centroids are denoted with *plus* sign.

The total running time of the algorithm is $O(|P| * k * n * t_1 + k * n * t_2)$, where $|P|$ is number of particles, *k* is the predefined number of clusters, *n* is number of data points, t_1 is the predefined number of iterations of the

evolutionary hybrid part and t_2 is the iteration number needed to converge in *k*-means. $O(|P| * k * n * t_1)$ is also the running time of PSO part and $O(k * n * t_2)$is the one for *k*-means. PSO and *k*-means are implemented on the data points and all other parts are implemented on the particles. Since the number of particles is much less than the number of data points, we ignore the time complexities associated with the *Elimination*, SA and *SS* parts when considering the overall complexity.

4. Experimental results

To run experiments, we used two separate data sets. For simplicity we call them DS1† and DS2‡. DS1 consists of 17 features and 22784 instances and DS2 consists of 3 features and 855367 instances. The proposed model, HE-*k*means, was compared with s-k-means (standard *k*-means) [2] and PSO+*K*-means [9]. In all experiments the results were averaged over several runs.

Figure 4(a) shows the results of experiments that was carried on DS1 with different number of particles for HE-*k*means and PSO+*k*-means [9] with $k = 2$. s-k-means is not dependent with any particles, so it is shown in the figure just to indicate the difference between the results. The experiments were carried with $t = 20$, iteration number of *20*. The graph indicates that as the number of particles increases PSO+*k*-means and HE-*k*means tend to find better results. This is expected since more particles mean more search can be performed on the solution space and the probability of finding a better result is increased. As can be seen from the figure, the proposed model gives better results than s-k-means and PSO+*K*-means. One of the reasons for this is due to the fact that, HE-*k*means has *k*-means and PSO integrated together in such a way that the algorithm utilizes all advantages of the corresponding algorithms while removing the drawbacks. It is probably safe to state that the local minimums of this data set was in close proximity, because the results between s-k-means which is based on a greedy algorithm and other two algorithms did not vary much. This was observed because the data set had one major minimum and most points were able to converge to the nearest local minimum with greedy manner.

Figure 4(b) shows the experimental results for $k = 2$ with different iteration values. The overall picture stays almost the same, however, one important noteworthy point is that as the iteration number increase, the probability of finding a better result is increased more in HE-*k*means. The reason is that in SS part of the proposed model, there are more options to form a better particle and as we will see below, considerable amount of *gBest* solutions indeed come from this part of the algorithm.

Another experiment that was carried is the frequency of updating the *gBest* value in PSO+*K*-means and HE-*k*means in *10* iterations. The results show that PSO+*K*-means updates the *gBest* value *1.2* times on average in *10* iterations. On the other hand, HE-*k*means performs the same in 1.8 times on average. As a result, HE-*k*means tends to find better solutions more often. The reason for this is as follows: HE-*k*means encapsulates almost everything that PSO+*K*-means has; moreover, it has more functions to search and find new solutions. However, increasing the number of iterations, does not guarantee finding global best solution, because it is NP-hard problem.

Yet another experiment was done on HE-*k*means's parts. To be more specific, the experiment was carried *100* times with random parameters like $k = 2$ *or 3*, data set = DS1 or DS2, iteration number = a random number between *10* and *20*, particle number = a random number between *10* and *30*. We noted the part of the algorithm in which *gBest* values gets updated. Since during the SS part of the algorithm, no computation of *gBest* was implemented, and only new particles were constructed, in the next iteration, if one of these particles were chosen as *gBest*, we consider that *gBest* came from the SS part. The results indicate that from *100 gBest* values, *11* of them came from the SA part, *47* of them came from the PSO part and *42* of them came from the SS part.

Figure 4(c) shows the comparison between proposed model and *s*-k-means for various initial tries. It is clear that, repeating s-k-means many times does not bring better results significantly. The reason is that, s-k-means does not search for solution space and starts randomly. We gave same running time for s-k-means as HE-*k*means. That is, we run HE-*k*means and give s-k-means several tries for the same running time. The results were similar to Figure 4(c) because there is a correlation between running time and number of initial tries. Moreover, the probability of s-k-

† http://sci2s.ugr.es/keel/dataset.php?cod=158

‡ http://sci2s.ugr.es/keel/dataset.php?cod=1261

mean's finding better solution by just randomly selecting initial centroids than HE-*k*means is very low and it is getting even lower when data set is dense and have large solution space.

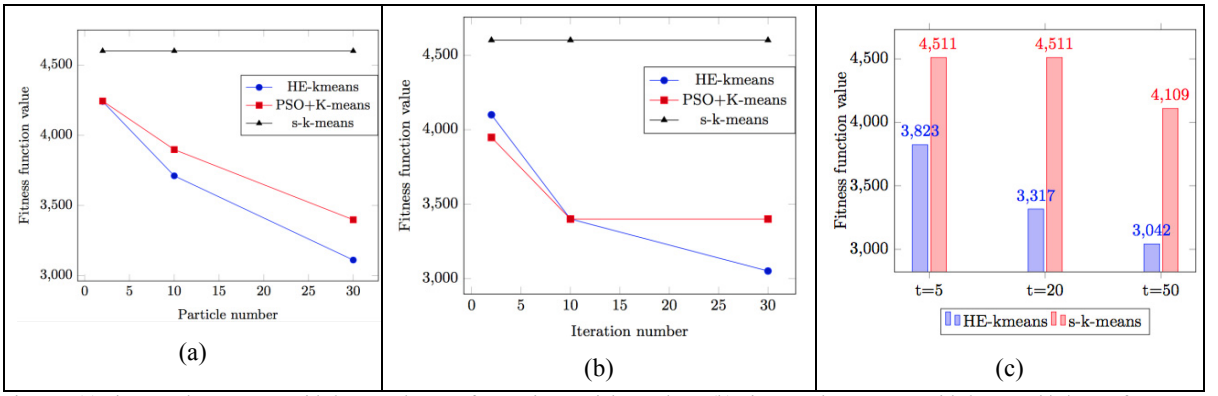


Figure 4.(a) Fitness values on DS1 with $k = 2$ and $t = 20$ for varying particle numbers. (b) Fitness values on DS1 with $k = 2$ and $|P| = 20$ for varying iterations. (c) Comparison of fitness values on DS1 with $k = 5$ between HE-kmeans and s-k-means.

The results of the same experiments performed with DS2 are shown in Table **2**, where t is the number of iterations and $|P|$ is the number of particles. The overall picture is almost the same as DS1 case. Here we encountered several cases where PSO+*K*-means got slightly better results than HE-*k*means. This can be seen in 4th line of table. This can happen when the number of particles is small. Since both PSO+*K*-means and HE-*k*means begin with randomly distributed particles, some particles can fall into the location, which has better fitness value than others. However, as the number of particles increase, PSO+*K*-Means' search mechanism becomes insufficient when compared to HE-*k*means. Again, we included s-k-means to the table to show the results with $k = 3$ and $k = 2$ and compare them accordingly, since s-k-means does not depend on the parameters t and $|P|$. Table **1** shows the information about the running time of HE-*k*means in seconds. It is obvious that PSO+*K*-means would run faster than HE- *k*means and worse than s-k-means, therefore, we did not put an extra column for PSO+*K*-Means. From the table we see that, as $|P|$ increases, the running time of HE-*k*means increases approximately linearly. We say approximately because, in *k*-means part of HE-*k*means, the iteration number needed for convergence cannot be predicted beforehand. The same can be applied to the increase in the iteration number i. As the iteration number increases, the running time of HE-*k*means increases.

Table 1. Running times on DS1 for different configurations.

Line	Configuration	HE-*k*means	s-k-means	Line	Configuration	HE-*k*means	s-k-means
1	k=2,i=10,|P |=2	50	35	4	k=2,|P |=10,i=10	62	35
2	k=2,i=10,|P |=10	62	35	5	k=2,|P |=10,i=30	91	35
3	k=2,i=10,|P |=30	76	35	6	k=2,|P |=10,i=50	133	35

Table 2. Fitness values on DS2 for different configurations

Line	Configuration	HE-kmeans	PSO+K-Means	s-k-means	Line	Configuration	HE-kmeans	PSO+K-Means	s-k-means
1	t=10,k=2,|P |=2	7072	7101	9488	7	|P |=10,k=2,t=10	6023	6203	9488
2	t=10,k=2,|P |=10	6023	6203	9488	8	|P |=10,k=2,t=30	5311	5377	9488
3	t=10,k=2,|P |=30	5688	6203	9488	9	|P |=10,k=2,t=50	4611	5066	9488
4	t=10,k=3,|P |=2	4111	4101	4712	10	|P |=10,k=3,t=10	2712	3811	4712
5	t=10,k=3,|P |=10	2533	3211	4712	11	|P |=10,k=3,t=30	2712	3033	4712
6	t=10,k=3,|P |=30	2533	3091	4712	12	|P |=10,k=3,t=50	2712	3033	4712

5. Conclusions

In this study we proposed a hybrid model for compact data clustering problems, which integrates PSO, SS and SA to find "good" initial centroids for *k*-means. The hybrid model was constructed in such a way that the overall model improved the clustering quality achieved by the standard *k*-means algorithm without sacrificing much from the execution time performance of *k*-means. The results were compared with the original *k*-means algorithm and a PSO+*K*-means implementation by using two separate datasets with different clustering and model parameters and our model outperformed the original and PSO-based *k*-means in clustering quality by approximately *30%*.

References

1. A. K. Jain and R. C. Dubes, Algorithms for clustering data, Prentice-Hall, Inc. (1988).
2. J. MacQueen, Some methods for classification and analysis of multivariate observations, Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, California, USA (1967) vol. 1, pp. 281–297.
3. J. Kennedy, Particle swarm optimization, Encyclopedia of Machine Learning, Springer (2010) pp. 760–766.
4. J. Kennedy, J. F. Kennedy and R. C. Eberhart, Swarm intelligence, Morgan Kaufmann (2001).
5. P. J. Van Laarhoven and E. H. Aarts, Simulated annealing, Springer (1987).
6. F. Glover, M. Laguna and R. Mart, Fundamentals of scatter search and path relinking, Control and cybernetics (2000) 29, 653–684.
7. T. Niknam and B. Amiri, An efficient hybrid approach based on pso, aco and k-means for cluster analysis, Applied Soft Computing (2010) 10(1) 183–197.
8. D. Van der Merwe and A. P. Engelbrecht, Data clustering using particle swarm optimization, The 2003 Congress on Evolutionary Computation, IEEE (2003) vol. 1, pp. 215–220.
9. A.Ahmadyfard, H. Modares, Combining pso and k-means to enhance data clustering, International Symposium on Telecommunications, IEEE (2008) 688–691.
10. C. Y. Chen and F. Ye, Particle swarm optimization algorithm and its application to clustering analysis, IEEE International Conference on Networking, Sensing and Control, IEEE (2004) vol. 2, pp. 789–794.
11. J. C. Bezdek, Pattern recognition with fuzzy objective function algorithms, Kluwer Academic Publishers (1981).
12. D. Arthur and S. Vassilvitskii, k-means++ The advantages of careful seeding, Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics (2007) 1027–1035.
13. K. Krishna and M. N. Murty, Genetic k-means algorithm, Systems, Man, and Cybernetics, Part B, IEEE Transactions on Cybernetics (1999) 29(3), 433–439.
14. M. Laszlo and S. Mukherjee, A genetic algorithm that exchanges neighboring centers for k-means clustering, Pattern Recognition Letters (2007) 28(16), 2359–2366.
15. G. P. Babu and M. N. Murty, Simulated annealing for selecting optimal initial seeds in the k-means algorithm, Indian Journal of Pure and Applied Mathematics (1994) 25(1-2), 85–94.
16. P. S. Bradley and U. M. Fayyad, Refining initial points for k-means clustering, ICML, Citeseer (1998) vol. 98, pp. 91–99.
17. J. Z. Huang, M. K. Ng, H. Rong and Z. Li, Automated variable weighting in k-means type clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence (2005) 27(5), 657–668.
18. C. H. Chang, Simulated annealing clustering of chinese words for contextual text recognition, Pattern Recognition Letters (1996) 17(1) 57–66.
19. S. Scheuerer and R. Wendolsky, A scatter search heuristic for the capacitated clustering problem, European Journal of Operational Research (2006) 169(2) 533–547.
20. A. Likas, N. Vlassis and J. J. Verbeek, The global k-means clustering algorithm, Pattern recognition (2003) 36(2), 451–461.
21. A. M. Bagirov, Modified global k-means algorithm for minimum sum-of-squares clustering problem, Pattern Recognition 41.10 (2008) 3192-3199.