Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2020.010128
Article

An Improved Algorithm of K-means Based on Evolutionary Computation

Yunlong Wang[1,2,3], Xiong Luo[1,2,4,*], Jing Zhang[1,2,3], Zhigang Zhao[1] and Jun Zhang[5]

[1]School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, 100083, China
[2]Beijing Key Laboratory of Knowledge Engineering for Materials Science, Beijing, 100083, China
[3]Key Laboratory of Wind Energy and Solar Energy Technology (Inner Mongolia University of Technology), Ministry of Education, Hohhot, 010051, China
[4]Shunde Graduate School, University of Science and Technology Beijing, Foshan, 528399, China
[5]Science and Technology Division, North China Institute of Science and Technology, Beijing, 101601, China
[*]Corresponding Author: Xiong Luo. Email: xluo@ustb.edu.cn

Abstract: K-means is a simple and commonly used algorithm, which is widely applied in many fields due to its fast convergence and distinctive performance. In this paper, a novel algorithm is proposed to help K-means jump out of a local optimum on the basis of several ideas from evolutionary computation, through the use of random and evolutionary processes. The experimental results show that the proposed algorithm is capable of improving the accuracy of K-means and decreasing the *SSE* of K-means, which indicates that the proposed algorithm can prevent K-means from falling into the local optimum to some extent.

Keywords: Evolutionary computation; jaya algorithm; K-means; local optimum; simulated annealing

1 Introduction

With the rapid advancement of technology, it has now entered the information age of big data. Then, big data technologies nowadays can be applied in some fields, such as cyber-physical systems [1], green applications [2], Internet of Things [3], and many others [4–7]. Meanwhile, extracting valuable information from big data plays a key role.

Clustering, as the process of dividing multiple objects into multiple classes of similar objects, is helpful in mining useful information from unsupervised data, while extracting valuable information. It looks for similar samples in the dataset and then groups them. Its goal is to make the objects within the same group similar and the objects between different groups different as much as possible. The higher the similarity in the same group is and the lower the similarities between different groups are, the better the clustering effect is. Clustering is an important technique in data mining. It can divide data into meaningful clusters, in which each can be regarded as a classification. Dividing data into different groups means marking different data with different features and highlighting the salient features of different types of data, which can help us better understand the data. Clustering techniques can be used for anomaly detection. In the data cleaning process, clustering can clean out some abnormal data with obvious problems, and it can also find abnormal users. In addition, clustering can be applied in outlier detection [8]. In search engines, the query purposes of many users are similar, but they use different keywords, and some users do not even think of important keywords. Via clustering different classes or different topics, keyword recommendation during the search process can be performed. In image segmentation, pixels of an image can be mapped into a feature space. The clustering algorithms are able to analyze the feature points of the pixels in the feature space and divide them in accordance with the aggregation results, and then map these points back to the pixel space to obtain the segmentation results.

The K-means algorithm, as one of clustering algorithms, is simple to implement with a fast convergence speed, while achieving good effect [9]. It is widely used in the industry and some other fields. However, it has many disadvantages as well, one of which is that it is sensitive to initial values and is prone to fall into local optimum. To overcome those limitations, there are many variants of K-means. K medians [10] is an algorithm that is very similar to K-means. K-means is to continuously update the centers of the cluster during constant iterations. Its selection of the centers is determined according to the mean of samples of each category while K-medians' selection is achieved according to the median. Therefore, the impact of noise on the cluster center is accordingly reduced. K-means is a hard clustering compared with fuzzy C-means clustering (FCM), a kind of soft clustering [11]. FCM can indicate the degree of membership of a sample point to any cluster. K-means++ [12] attempts to solve the initial value selection problem of K-means to some extent by selecting points farther from the current cluster center points. Kernel functions are often used to solve nonlinear problems [13]. Then, the kernel K-means [14] maps the points of the input space into a high-dimensional space through nonlinear mapping. This method of mapping data into high dimensional space can highlight the differences between sample categories, making the data approximately linearly separable. The Bisecting K-means [15] is not affected by the initial value problem, and can speed up the convergence of K-means. Its clustering result is relatively better than K-means. The Mini-Batch K-means [16] is an improvement of K-means designed for larger data sets. It obtains a subset of samples from a small amount of samples at each iteration, and uses these randomly generated subsets of samples to update the centers.

However, most algorithms do not provide solutions to solve the problem that K-means is sensitive to the initial center points. The choice of the first cluster center of K-means++ is very significant while noise points will have greater impacts on the choice of cluster centers. In the clustering process of Bisecting K-means, once one point is divided into one cluster, it can no longer be clustered with the points in other cluster. It may lead to an issue that some points which belong to the same cluster in principle are divided into different clusters, and there is no chance to be clustered together for them.

Considering that many good ideas are given for the issue of jumping out of the local optimum in the evolutionary computation field, we propose an improved algorithm of K-means in this paper to solve the problem that K-means is easy to fall into local optimum, on the basis of several evolutionary algorithms. In our algorithm, the random process is added, and there is a possibility of jumping out of the local optimum from beginning to end. In the experiments, we can find that the clustering effect has been improved while more time are spent.

The remainder of this paper is organized as follows. Section 2 reviews the related work, while Section 3 analyzes our proposed algorithm. The experimental results are summarized in Section 4, and a short conclusion is provided in Section 5.

2 Related Work

2.1 K-means Algorithm

Here, we provide a short description of K-means algorithm. Firstly, k centers are randomly selected, and then the distance between the j-th sample x_j and the i-th center c_i is calculated as:

$$dist(i,j) = \sum_{k=1}^{d}(x_{jk} - c_{ik})^2, \quad (1)$$

where d is the dimension of samples.

Then, each sample is assigned to the cluster center closest to it. In the next step, each cluster center is updated as:

$$c_i' = \frac{1}{m_i}\sum_{x_j \emptyset c_i} x_j, \quad (2)$$

where $x_j \emptyset c_i$ refers to that sample x_j belongs to the group determined by the centre c_i, and the total number of samples belonging to this group is m_i. The distance between each sample and each cluster center is recalculated. This process is repeated until the algorithm converges.

The objective function *SSE* in the entire clustering algorithm is as follows:

$$SSE = \sum_{i=1}^{k} f(c_i), \tag{3}$$

$$f(c_i) = \sum_{x_j \emptyset c_i} (c_i - x_j)^2. \tag{4}$$

The metric *SSE* is continually minimized, and when the optimization converges the clustering centers we look for are obtained.

2.2 Evolutionary Computation

Evolutionary computation is a group-oriented randomized computational model that simulates the evolution of natural organisms. Evolutionary computation algorithms can be used to solve constrained and unconstrained optimization problems.

Currently, there have been many practical problems that are difficult to be solved with traditional methods. Then, evolutionary computation methods could be used to address them. Evolutionary computation has characteristics of self-organization, self-learning and self-optimization, and its principles are often simple. Meanwhile, evolutionary algorithms usually have excellent global optimal solutions.

Evolutionary computation uses a kind of group search strategy that basically does not rely on knowledge of the search space and other ancillary information. So far, there have been many outstanding algorithms in the field of evolutionary computation.

The most representative method is the genetic algorithm (GA), which searches for the optimal solution by simulating the natural evolution process [17]. Particle swarm optimization (PSO) is a swarm intelligence algorithm designed by simulating the predation behavior of birds to search for optimal solutions and was introduced in [18]. The ant colony optimization [19] is a simulation optimization algorithm that simulates the foraging behavior of ants. The differential evolution (DE) algorithm is based on the simple mutation of the score and the one-to-one competitive selection strategy, which reduces the complexity of genetic operations [20]. Li proposed the fish-swarm algorithm in 2002 [21]. The simulated annealing (SA) algorithm [22] draws on the annealing principle of solids. This algorithm is a method based on Monte Carlo thought designed to solve the optimization problem. It accepts a solution that is worse than the current solution with a certain probability at each iteration, and thus may jump out of the current local optimum. Based on the inspiration of night sky fireworks explosion, a new swarm intelligence optimization algorithm, named fireworks algorithm, was proposed in [23]. Later in 2011, the brainstorming algorithm was proposed based on a swarm intelligence algorithm, which simulated the innovative thinking of human brainstorming [24]. It is very suitable for solving complex multi-peak and high-dimensional function problems. Jaya algorithm is a novel optimization algorithm proposed in [25], without algorithm-specific control parameters. Jaya is a widely used optimization algorithm. A GPU-accelerated parallel Jaya algorithm was presented to estimate Li-ion battery model parameters [26]. An elite opposition-based Jaya was proposed to estimate the parameters of photovoltaic cell models [27].

More recently, there are some methods introduced to combine K-means and evolutionary algorithms. For example, a one-step K-means mechanism was used to enhance the performance of the original Jaya algorithm [28]. A combination of bare bones fireworks algorithm and K-means was proposed to cluster data [29]. A genetic-based algorithm was used to obtain the best parameters in two improved K-means algorithms [30]. An enhanced K-means clustering with PSO was introduced to obtain refined set of data points [31].

Different from the above methods, rather than combine several algorithms, we refer to several significant ideas from evolutionary computation to help solve the problem that K-means is easy to fall into local optimum.

3 The proposed Algorithm

As shown in Fig. 1, the new algorithm adds two new processes, i.e., random and evolutionary processes of clustering centers, into the original algorithm. In the E step, samples are divided into several parts by grouping the sample points with its nearest center point into a class. In the M step, the clustering centers are updated with new centers obtained by calculating the mean of each set of sample points. Hence, the proposed algorithm increases two process, including random process and evolution process. Here, the random process is the stochastic process of the center points and the evolution process is the evolution of central points, between the E step and the M step.

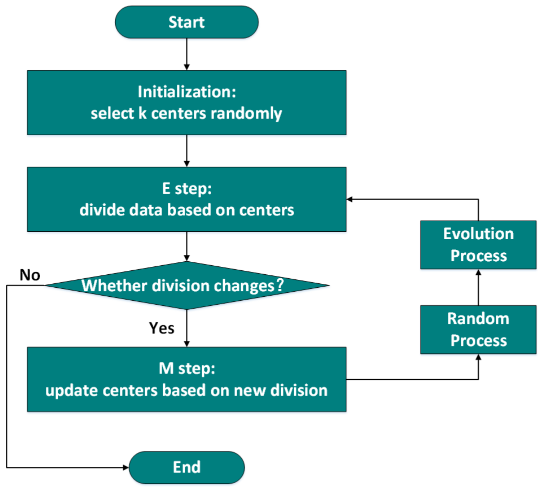
Start
Initialization: select k centers randomly
E step: divide data based on centers
Whether division changes?
No
Yes
M step: update centers based on new division
Evolution Process
Random Process
End

Figure 1: Structure of the proposed algorithm

3.1 Random Process

In random process, new random clustering centers are created randomly, which is a key module for the algorithm to jump out of local optimum. With the stochastic process in evolutionary computation, many methods can be used in this part. For example, a simple random process within a specified range can be used, or random swimming behavior and foraging behavior of the fish-swarm algorithm can be simulated. We can also refer to the intersection and variation processes of GA. For the same reason, the intersection and variation processes of DE can also be imitated.

Since the result of K-means algorithm is deterministic after the initial values of clustering centers determined, the random process of clustering centers here is particularly significant. Many ideas for generating stochastic processes can be used as references. Here, we introduce two simple random ideas from fish-swarm algorithm.

3.1.1 Swimming Random

Free swimming is a default behavior of fish-swarm algorithm. When there is no other behavior, the fish individual swims within the field of view. In order to jump out of the local optimality of K-means and look for global optimality, the horizon of fish is expanded to all searchable areas. The swimming behavior can be described as follows:

$$c_i^{sr} = c_i + step_s \,, \tag{5}$$

where c_i is the i-th center of all clustering centers and $step_s$ is the swimming step of the i-th center. It should be noted here that c_i^{sr} is random in all feasible solutions, which is a key point to help K-means to jump out of local optimum.

3.1.2 Foraging Random

Based on the current state of a fish, another state is randomly generated within its sensing range in fish-swam algorithm. Here, we use $step_f$ to control the foraging step.

$$c_i^{fr} = c_i + step_f \,, \tag{6}$$

where $\mathrm{c}_\mathrm{i}^\mathrm{fr}$ is a random candidate solution of the i-th center generated in visual field.

3.2 Evolution Process

After generating new random cluster center points, the next operation is to update centers. This updating process of the cluster centers is called evolution process, which in fact is the selection of clustering centers. Similarly, there are many selection mechanisms. Here, we introduce two methods based on Jaya optimization algorithm and SA algorithm.

3.2.1 Jaya Selection

According to the idea of Jaya optimization algorithm, the center points are supposed to be further to the less effective candidate solutions generated randomly and be closer to the better candidate solutions with a certain step size. The selection process is as follows:

$$c_i' = c_i + flag \times (c_i^r - c_i) \times p \, , \tag{7}$$

where c_i^r is c_i^{sr} or c_i^{fr} or other clustering centers generated randomly for the *i*-th center, and *flag* is 1 if the new clustering center c_i^r is better than current center c_i or -1 in contrast. Here, we use p which ranges within [0, 1] to control the updating step size, since the ranges are different in different dimensions of clustering centers. Then, c_i', i.e., the updated center of c_i after one iteration, is obtained.

3.2.2 Simulated Annealing (SA) Selection

When it is searching for the best solution, the SA algorithm will accept a bad solution with a certain probability, which may be helpful in jumping out of the local optimum. This probability gradually decreases with time. The probability can be described by:

$$y = \begin{cases} 1, & f(c_i^r) - f(c_i) < 0 \\ e^{-\frac{df}{T}}, & f(c_i^r) - f(c_i) \geq 0 \end{cases} , \tag{8}$$

where *y* represents the probability described above and *T* is the temperature which will decrease gradually. Moreover, *df* represents $f(c_i^r) - f(c_i)$. If *df* < 0, then *T* is updated as:

$$T = T \times \alpha \, , \tag{9}$$

where α which ranges from 0 to 1 controls the rate of change of temperature *T*.

It is obvious that the higher the temperature is, the higher the probability is, and more likely it is to jump out of the current local solution. We also set parameters to maintain the best centers throughout the iterations of our proposed algorithm. If the centers obtained when the cluster converges are worse than the best centers, the set of best centers is the final solution.

3.3 Computational Complex Analysis

Supposing we want to find *k* center points from *n* sample points. Each sample is *d*-dimensional and the number of iteration is *t*.

We analyze the time complexity of K-means first. In each iteration, the distances between each sample and *k* centers are calculated, so the time complexity is $O(n \times k \times d)$. In each iteration, *k* cluster centers are also calculated. Considering that the time spent by addition and subtraction in the CPU is much less than multiplication and division, only the time of multiplication and division is considered here, where the time complexity is $O(k)$. Therefore, the total time complexity is about $O(t \times (n \times k \times d + k))$, which is equivalent to $O(t \times n \times k \times d)$.

In the proposed algorithm, supposing that we try the random process and evolution process for *s* times during each iteration, then the time complexity of proposed algorithm is about $O(s \times t \times n \times k \times d)$. If we consider *s*, *t*, *k*, and *d* as constants, the time complexity of our proposed algorithm is $O(n)$.

4 Experimental Results and Discussion

4.1 Experimental Environment and Metrics

The computer used here is a server with an 8-core CPU and a GeForce GTX 1080Ti graphics card. The operating system is a 64-bit Ubuntu system while the server's memory is 31.3 GB and the SSD size is 218.2 GB. In addition, Spyder, a free integrated development environment with Anaconda that is an open-source platform for python data science, is applied. In our experiments, the K-means implementation in the Scikit-learn which is a free efficient tool for machine learning in Python, is modified and called to help achieve our algorithm.

The Accuracy (*Acc*) is described as follows:

$$Acc = \frac{N_c}{N_t}, \tag{10}$$

where N_c is the number of correct predictions and N_t is the total number of predictions. It is noted here that after the data is clustered, the label with the most occurrences in each cluster is taken as the label of the samples of corresponding cluster. Specifically, some other metrics are also applied to evaluate the performance of our algorithm. The Davies-Bouldin index (*DBI*) was presented [32]. The adjusted rand index (*ARI*) was introduced [33]. The silhouette (*SIL*) coefficient was proposed in [34]. The mean *SIL* of all samples are calculated in the experiments. In addition, the adjusted mutual information (*AMI*) proposed in [35], is applied in our experiments.

In general, a total of 7 metrics which include running time on the CPU, *Acc*, *SSE*, *DBI*, *ARI*, *AMI*, and *SIL* are accordingly computed in the experiments.

4.2 Experimental Results

4.2.1 Datasets

Two datasets, i.e., Iris dataset and Glass dataset, are evaluated in this section. These datasets are both from UCI machine learning repository [36]. All samples will be used for clustering, and the evaluation for metrics will be performed on all samples.

The Iris dataset collected three types of irises: Setosa Iris, Versicolour Iris, and Virginica Iris. Each type of iris flower collected 50 sample records for a total of 150 records. This dataset consists of four attributes, the length of the flower bud, the width of the flower bud, the length of the petals, and the width of the petals. Hence, we have 150 samples, and each one is with 4 attributes and 1 label.

The Glass dataset is not related to agriculture field, and it is used to verify the effectiveness of the proposed algorithm. This dataset is a Glass identification dataset. It is made up of ten attributes, including id, refractive index and some others. The number of instances is 214.

Firstly, the methods are applied in Iris dataset which has 150 samples with features of 4 dimensions and corresponding labels. The methods that we choose to compare are listed in Tab. 1.

Specifically, we randomly select initial centers for K-means and record the results of the K-means algorithm. Furthermore, these methods are run 100 times and the average results are taken to evaluate the performance.

4.2.2 Comparison

Tab. 2 shows the performance of proposed algorithm on Iris dataset. In the KSJ algorithm, we perform random perturbations in the full range, and then advance or retreat in a certain step learning from Jaya Algorithm. After the random perturbation process in the KSAJ algorithm, we first use the mechanism of SA algorithm to accept or abandon the new candidate solution of the current sample, and then make the sample approach or stay away from the solution in the Jaya mechanism. In the KFJ and KFAJ algorithms, the processing is similar to the previous two algorithms. The difference is that in the random process we take a fixed step size to achieve the random process within a certain range.

Table 1: Methods and its abbreviation

Abbreviation	**Methods**
K	K-means
KSJ	K-means + Swimming + Jaya
KSAJ	K-means + Swimming + Annealing + Jaya
KFJ	K-means + Foraging
KFAJ	K-means + Foraging + Annealing + Jaya
BK	BisectingKmeans
K++	K-means++
KSJ++	Swimming + Jaya + K-means++
KSAJ++	Swimming + Annealing + Jaya + Kmeans++
KFJ++	Foraging + K-means++
KFAJ++	Foraging + Annealing + Jaya + Kmeans++

Table 2: Performance comparison on Iris dataset

Methods	Time/s	*Acc*	*SSE*	*DBI*	*ARI*	*AMI*	*SIL*
K	0.830	0.667	142.754	0.993	0.429	0.528	0.519
BK	12.093	**0.865**	**85.868**	**0.667**	0.672	0.677	0.538
KSJ	5.198	0.810	101.902	0.779	0.617	0.664	0.538
KSAJ	6.673	**0.862**	**86.451**	**0.701**	**0.686**	**0.713**	**0.546**
KFJ	4.852	0.750	121.576	0.872	0.538	0.608	0.528
KFAJ	10.292	**0.885**	**80.161**	**0.670**	**0.717**	**0.736**	**0.551**

The values in time column show that the new methods take about 10 times the time of the original algorithm generally, which corresponds to the stochastic process that generates random centers for 10 times in the random step. The new methods have improved the accuracy of K-means. Among these methods, the KSAJ method is relatively better in the Iris dataset whose accuracy is about 0.2 higher than 0.667 of K, while the KFAJ method performs best in the Iris dataset whose accuracy reaches 0.885, about 22 percent higher than K. The *SSE* column shows that different methods can reduce *SSE* to different extents and obtain better results. The best of these methods is KFAJ, which reduces the original *SSE* value from 142.754 to 80.161. In the *DBI* item, the lower the *DBI* is the better the result is. The best one is the KFAJ method which reduces the *DBI* from 0.993 to 0.670, which is a good result relatively.

Table 3: Performance of improved K++ on Iris dataset

Methods	Time/s	*Acc*	*MSE*	*DBI*	*ARI*	*AMI*	*SIL*
K++	1.124	0.883	80.798	0.656	0.717	0.734	0.556
KSJ++	6.718	0.889	78.854	0.665	0.721	0.739	0.552
KSAJ++	53.222	**0.889**	**78.854**	0.664	**0.722**	**0.739**	0.552
KFJ++	6.181	0.887	79.493	0.662	0.721	0.738	0.553
KFAJ++	204.971	**0.893**	**78.852**	0.662	**0.730**	**0.748**	0.553

AMI evaluates the agreement of two assignments, while the *ARI* measures the similarity of two assignments. *SIL* combines both cohesion and resolution and can be used to evaluate different algorithms based on the same raw data. For these metrics, the higher the values are, the better the result is. From Tab. 2, we can easily find that the proposed algorithm is able to improve the performance of initial K-means on these three metrics. And BK and KSAJ perform similarly on the *Acc* metric. On the whole, the KSAJ and KFAJ algorithms outperform the BK algorithm.

To verify the versatility of the random process and evolution process, these two processes are added to the K++ algorithm. The experimental results are in Tab. 3.

The results in Tab. 3 show that the time of clustering increases after the K-means++ algorithm is improved. The KFAJ++ algorithm takes the most time, which is about 205 seconds and it also achieves the most improvement in accuracy. Specifically, KFAJ++ increases the accuracy from 0.883 to 0.893.

In addition, the *MSE* metric has been reduced to varying degrees, while the results on *ARI* and *AMI* are also improved. The KFAJ++ algorithm achieves the most obvious improvement, increasing *ARI* by 1.3 percent and increasing *AMI* by 1.5 percentage points. There is basically no change in *DBI* and *SIL*.

On Glass dataset, we take a similar configuration in the previous experiments on Iris dataset. The results are in Tab. 4.

Table 4: Performance comparison on Glass dataset

Methods	Time/s	*Acc*	*MSE*	*DBI*	*ARI*	*AMI*	*SIL*
K	1.016	0.449	704.298	1.247	0.224	0.222	0.568
KSJ	4.801	0.455	691.844	1.205	0.224	0.226	0.568
KSAJ	4.890	0.456	693.273	1.196	0.223	0.225	0.562
KFJ	5.399	0.471	671.698	1.105	0.223	0.231	0.542
KFAJ	6.238	**0.481**	**660.695**	**1.051**	0.219	**0.232**	0.527

From the experimental results in Tab. 4, it is evident the new methods still spend more time than K, but the time taken is less than 10 times the time of K. Besides, new methods are able to improve the accuracy rate by a few percentage points compared with K. The algorithm with the least improvement is KSJ which is about 1 percent better than K, while the highest is KFAJ which is about 3 percent.

From the experimental results in Tab. 2, we can conclude that the proposed algorithm improves the accuracy of clustering on the Iris agricultural dataset and performs well on several other metrics. It improves the original K-means algorithm and has an advantage over another improved algorithm BK. The results in Tab. 4 can also verify the effectiveness of the proposed algorithm. From the experimental results in Tab. 3, we can find that after applying the improved K-means++ algorithm to the Iris agricultural dataset, good clustering results can be achieved.

In practical applications, agricultural data is generally very complex. The clustering results of agricultural data are improved, which means that we can more easily extract the hidden information in the data and mine the rules that are difficult to find. In so doing, the agricultural data can be better analyzed.

4.2.3 Impact of Parameters

In order to explore the effect of different parameters on the results, different values of two important parameters, i.e., T and α appeared in (8) and (9) of KSAJ, are set in experiments on the Iris dataset.

First, T is set to 100, then the value of α is changed and the results are in Fig. 2. The results show that the larger α is, the more time is consumed, and the higher the accuracy is in general. The explanation for this phenomenon is that if α is larger, T will decrease more slowly, and the search range will be expanded while the time will increase correspondingly.

Then, α is set to 0.95, and then the value of T is changed. The results are in Fig. 3, which indicate that the larger T is, the longer the consumption time is, and the better the corresponding accuracy is. The explanation for this phenomenon is that if T is larger, the search range will be expanded, and the time will increase correspondingly.

In general, the newly proposed algorithm has achieved satisfactory performance, but it also has its shortcomings as well. For those data with global optimal solution and local optimal solutions far away, we recommend the KSAJ method and for those cases where the global optimal solution and most of the local optimal solutions are relatively close, we recommend the KFAJ method.

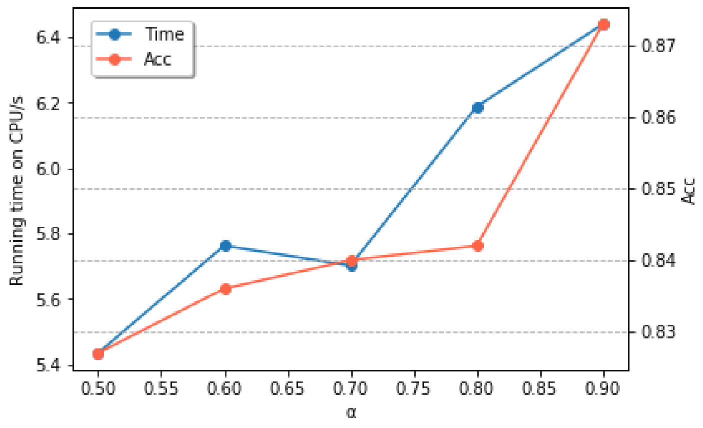

Figure 2: Experimental results of running time and *Acc* of KSAJ with different *α*

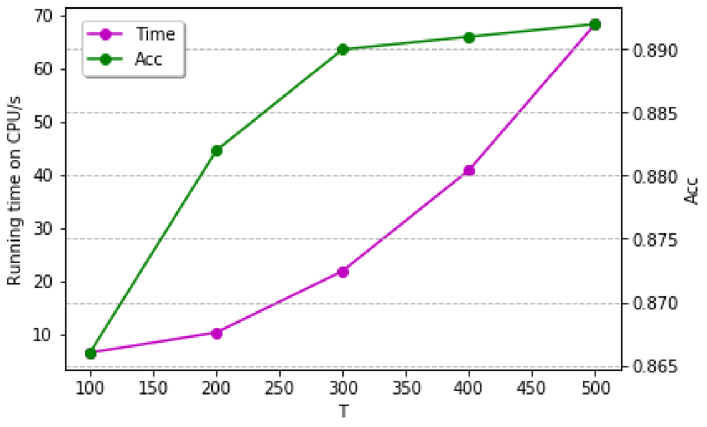

Figure 3: Experimental results of running time and *Acc* of KSAJ with different *T*

5 Conclusion

In this paper, motivated by evolutionary computation paradigms, a new algorithm used to improve traditional K-means is proposed. Stochastic process and evolutionary process are incorporated into initial iterative process. The experimental performance on the Iris and the Glass datasets show that our new algorithm is able to improve the result of K-means. Meanwhile, the metrics *MSE* and *DBI* can be lowered, and several other metrics can be improved to varying degrees.

In the future work, we will try more combinations in the proposed algorithm framework, such as trying to combine GA which is very representative in evolutionary computation with Jaya algorithm, with the purpose of further improving the computational performance.

Funding Statement: This work was supported in part by the National Key R&D Program of China under Grant 2016YFC0600510, in part by the National Natural Science Foundation of China under Grant U1836106, in part by the Beijing Natural Science Foundation under Grant 19L2029, in part by the Beijing Intelligent Logistics System Collaborative Innovation Center under Grant BILSCIC-2019KF-08, in part by the Scientific and Technological Innovation Foundation of Shunde Graduate School, USTB, under Grant

BK19BF006, in part by the Fundamental Research Funds for the University of Science and Technology Beijing under Grant FRF-BD-19-012A, and in part by the Key Laboratory of Geological Information Technology, Ministry of Natural Resources of the People's Republic of China under Grant 2017320.

Conflicts of Interest: No potential conflict of interest was reported by the authors.

References

[1] R. Atat, L. Liu, J. Wu, G. Li, C. Ye and Y. Yang, "Big data meet cyber-physical systems: A panoramic survey," *IEEE Access*, vol. 6, pp. 73603–73636, 2018.

[2] J. Wu, S. Guo, J. Li and D. Zeng, "Big data meet green challenges: Big data toward green applications," *IEEE Systems Journal*, vol. 10, no. 3, pp. 888–900, 2016.

[3] H. Teng, Y. Liu, A. Liu, N. N. Xiong, Z. Cai *et al.,* "A novel code data dissemination scheme for internet of things through mobile vehicle of smart cities," *Future Generation Computer Systems*, vol. 94, pp. 351–367, 2019.

[4] M. Chen, Y. Li, X. Luo, W. Wang, L. Wang *et al.,* "A novel human activity recognition scheme for smart health using multilayer extreme learning machine," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1410–1418, 2019.

[5] X. Luo, J. Sun, L. Wang, W. Wang, W. Zhao *et al.,* "Short-term wind speed forecasting via stacked extreme learning machine with generalized correntropy," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4963–4971, 2018.

[6] X. Luo, Y. Xu, W. Wang, M. Yuan, X. Ban *et al.,* "Towards enhancing stacked extreme learning machine with sparse autoencoder by correntropy," *Journal of The Franklin Institute*, vol. 355, no. 4, pp. 1945–1966, 2018.

[7] X. Luo, C. Jiang, W. Wang, Y. Xu, J. H. Wang *et al.,* "User behavior prediction in social networks using weighted extreme learning machine with distribution optimization," *Future Generation Computer Systems*, vol. 93, pp. 1023–1035, 2019.

[8] H. Du, S. Zhao, D. Zhang and J. Wu, "Novel clustering-based approach for local outlier detection," in *Proc. of the IEEE Conf. on Computer Communications Workshops*, Piscataway, NJ, USA, 2016, pp. 802–811.

[9] E. W. Forgy, "Cluster analysis of multivariate data: efficiency versus interpretability of classifications," *Biometrics*, vol. 21, pp. 768–769, 1965.

[10] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ, USA: Prentice Hall, 1988.

[11] J. C. Dunn, "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters," *Journal of Cybernetics*, vol. 3, no. 3, pp. 32–57, 1973.

[12] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proc. of the Eighteenth Annual ACM-SIAM Sym. on Discrete Algorithms*, New York City, NY, USA, 2007, pp. 1027–1035.

[13] X. Luo, J. Deng, J. Liu, W. Wang, X. Ban *et al.,* "A quantized kernel least mean square scheme with entropy-guided learning for intelligent data analysis," *China Communications*, vol. 14, no. 7, pp. 1-10, 2017.

[14] B. Schölkopf, A. Smola and K. R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[15] M. Steinbach, G. Karypis and V. Kumar, "A comparison of document clustering techniques," in *Proc. of the ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining Workshop on Text Mining*, New York City, NY, USA, 2000.

[16] D. Sculley, "Web-scale k-means clustering," in *Proc. of the 19th Int. Conf. on World Wide Web*, New York City, NY, USA, pp. 1177–1178, 2010.

[17] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.

[18] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning Encyclopedia of Machine Learning*. Boston, MA, USA: Springer, pp. 760–766, 2010.

[19] M. Dorigo and M. Birattari, *Ant Colony Optimization*. Boston, MA, USA: Springer, 2010.

[20] R. Storn and K. Price, "Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[21] X. L. Li, “An optimizing method based on autonomous animals: Fish-swarm algorithm,” *Systems Engineering-Theory & Practice*, vol. 22, no. 11, pp. 32–38, 2002.

[22] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[23] Y. Tan and Y. Zhu, “Fireworks algorithm for optimization,” in *Proc. of the Int. Conf. in Swarm Intelligence*, Berlin, Germany, pp. 355–364, 2010.

[24] Y. Shi, “Brain storm optimization algorithm,” in *Proc. of the Int. Conf. in Swarm Intelligence*, Berlin, Germany, pp. 303–309, 2011.

[25] R. Rao, “Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems,” *International Journal of Industrial Engineering Computations*, vol. 7, no. 1, pp. 19–34, 2016.

[26] L. Wang, Z. Zhang, C. Huang and K. L. Tsui, “A GPU-accelerated parallel Jaya algorithm for efficiently estimating Li-ion battery model parameters,” *Applied Soft Computing*, vol. 65, pp. 12–20, 2018.

[27] L. Wang and C. Huang, “A novel Elite Opposition-based Jaya algorithm for parameter estimation of photovoltaic cell models,” *Optik*, vol. 155, pp. 351–356, 2018.

[28] F. Chen, Z. Ding, Z. Lu and X. Zeng, “Parameters identification for chaotic systems based on a modified Jaya algorithm,” *Nonlinear Dynamics*, vol. 94, no. 4, pp. 2307–2326, 2018.

[29] E. Tuba, R. Jovanovic, R. C. Hrosik, A. Alihodzic and M. Tuba, “Web intelligence data clustering by bare bone fireworks algorithm combined with k-means,” in *Proc. of the 8th Int. Conf. on Web Intelligence, Mining and Semantics*, New York City, NY, USA, 2018.

[30] S. S. Yu, S. W. Chu, C. M. Wang, Y. K. Chan and T. C. Chang, “Two improved k-means algorithms,” *Applied Soft Computing*, vol. 68, pp. 747–755, 2018.

[31] N. Kant and M. Mahajan, “Time-series outlier detection using enhanced k-means in combination with PSO algorithm,” *Lecture Notes in Electrical Engineering*, vol. 478, pp. 363–373, 2019.

[32] D. L. Davies and D. W. Bouldin, “A cluster separation measure,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[33] L. Hubert and P. Arabie, “Comparing partitions,” *Journal of Classification*, vol. 2, no. 1, pp. 193–218, 1985.

[34] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[35] N. X. Vinh, J. Epps and J. Bailey, “Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance,” *Journal of Machine Learning Research*, vol. 11, pp. 2837–2854, 2010.

[36] M. Lichman, *UCI Machine Learning Repository*. 2020. [Online]. Available: http://archive.ics.uci.edu/ml.