ISBN 978-981-14-1455-8
Proceedings of 2019 The 11th International Conference on Future Computer and Communication
WCSE_2019_SPRING
Yangon, Myanmar, February 27-March 1, 2019, pp. 58-62

Differential Evolution for Large-Scale Clustering

Pyae Pyae Win Cho [1+], Thi Thi Soe Nyunt [2] and Thet Thet Aung [3]

[1 2 3] University of Computer Studies, Yangon

Abstract. Clustering is the task of organizing data instances into groups based on the similarity between them. It plays an essential role in knowledge discovery and data mining, ranging from preprocessing step to the final goal of the task. Evolutionary algorithms (EAs) based clustering methods have developed with the intention of enhancing the effectiveness and accurateness of clustering. The huge amount of data emerging by the progress of technology have become data clustering as a challenging task and as an attractive attention for the use of EAs based approaches. Differential Evolution (DE), an instance of EAs, has been exploited to discover the best solution for clustering problems. It has become a successful solution to produce more compact clusters than other traditional clustering techniques. This paper presents a parallel differential evolution algorithm on Spark framework to facilitate huge amount of data clustering. Experimentations were conducted on some frequently used UCI machine learning datasets. The results have presented that the proposed approach is effective and comparable to existing algorithms.

Keywords: Differential Evolution, Clustering, Apache Spark.

1. Introduction

Data clustering is a task to partition unlabelled dataset into different clusters according to the distances and similarities amongst data objects. There have been many applications for clustering such as data mining, image processing, market segmentation, information retrieval and bioinformatics. Over the years, researchers have developed a lot of algorithms for clustering [1]. Some well-known limitations of traditional clustering algorithms are sensitivity of initial condition, trapping to local optimum and the requisite of high computation effort for large problems. From a combinatorial optimization viewpoint, clustering problems can be precisely considered as a certain kind of NP-hard problem [2]. Evolutionary Algorithms (EAs) are able to offer near-optimal solutions for NP-hard problems in a satisfactory time. A large number of EAs based clustering algorithms have been developed in recent works [3]. Differential evolution (DE), one of the variants of EAs, has been shown as an effective alternative way to solve clustering problem for small to moderate size of data sets. DE has been widely used to either accomplish clustering independently or integrate with the clustering approaches. Although the DE approach is popular due to its good exploration and exploitation abilities in a search space, a lot of time is required to process the strategies of DE such as mutation, recombination and selection. Thus, the standard DE is not efficient for addressing large scale data sets. Apache Spark is a powerful in-memory cluster computing framework for processing and analysing enormous amount of data [4]. It can provide very high efficiency for the applications that utilize iterative data processing procedures, and hence, is applicable to large-scale data analysis problems. The aim of this paper is to propose a parallel differential evolution algorithm based on the Spark framework for large-scale clustering problems. The remained portions of the paper are organized as follows: related works are described in section 2, background theory is presented in section 3, the proposed approach and experimental result are shown in section 3 and 4, and finally this paper is concluded in section 6.

[+] Corresponding author. Tel.: + (095-9-975334170); fax: + (013-610-633).
E-mail address: (pyaepyaewincho@ucsy.ed.mm).

2. Related Works

Due to the unsupervised nature, clustering analysis is supposed to one of the most difficult and challenging tasks in data analysis. The idea of applying Evolutionary algorithms (EAs) for clustering problem seems to be an ordinary choice due to their capability of avoiding local minima trap. A large number of algorithms based on EAs have been recently proposed for clustering tasks. In [3], Hruschka et al. presented a broad exploration of various clustering methodologies based on EA approaches. The significant difference of these approaches is the representation of partitions encoded by individuals in population. Paterlini et al. compared to the performance of three kinds of EA, including Genetic Algorithms (GAs), Particle Swarm Optimization (PSO) and Differential Evolution (DE) on partitional clustering and concluded that DE outperforms the other two approaches [5]. Omran et al. [6] developed a clustering approach based on DE for the problems of image classification and segmentation, and stated that DE based clustering is comparable to other algorithms. Hybrid approaches combined DE with k-means also presented in a number of research works. In [7], Kwedlo et al. described a clustering method in which K-means algorithm is integrated into the procedure of DE and presented that the hybrid approach produced better cluster results than each of these two mechanisms applying independently. However, the authors also stated that the use of DE with K-means algorithm in huge amount of data clustering is restricted by unreasonable running time required to reach proper solutions. In [8], Tian et al. described a clustering approach based on DE and K-harmonic means (KHM) algorithm. Their approach applied KHM to all individuals in the population after the DE operations in order to take the merits and to escape the shortcomings of these two algorithms. Daoudi et al. exploited DE algorithm for gene expression data clustering based on MapReduce processing framework [9]. They proposed a parallel DE algorithm and stated that their proposed algorithm is comparable to classical K-means and MapReduce based PSO. Santhi et al. [10] applied Bat and Firefly algorithms to calculate the initial centroids for K-means and evaluated the performance of K-means with optimization algorithms on Spark framework.

3. Background Theory

3.1. Clustering

The purpose of clustering is to maximize the similarity or intra-connectivity between data instances in the same group and minimize the dissimilarity or inter-connectivity between distinct groups. Clustering can be divided into soft clustering and hard clustering. In soft clustering, an instance may belong to all the groups according to a probability or likelihood of belonging to the group. In hard clustering, the groups are mutually disjoint and non-overlapping in nature. Any data instance may belong to one and only one group. The hard partitioning clustering algorithms only will be considered in this paper.

Given a dataset $D = \{x_1, x_2, \ldots, x_n\}$, where x_i is an n-dimensional feature vector, its partition can be defined as a set $C^* = \{C_1, C_2, \ldots, C_k\}$ of k disjoint cluster C_i such that $C_1 \cup C_2 \cup \ldots \cup C_k = D$, $\forall C_i \neq \emptyset$ and $\forall_{i \neq j}$ $C_i \cap C_j = \emptyset$. And hence, a clustering problem can be stated as the problem of examining a partition which minimizes (or maximizes) a certain clustering validity criteria. SSE, sum of squared error (or total intra-cluster distance), one of the most frequently used validity criterion functions, can be defined as follow:

$$SSE(D, C^*) = \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - m_i\|^2 \quad (1)$$

where ||.|| denotes the Euclidean distance and mi denote the centroid of the corresponding cluster C_i.

3.2. Differential Evolution

Differential evolution (DE), developed by Storn and Price in 1995 [11], is a stochastic, population-based search strategy and has become one of the most frequently used and the most powerful evolutionary algorithms (EAs) for solving the continuous global optimization problems. DE employs similar basic operations of a standard EA such as mutation, crossover and selection. Firstly, the initial population is generated by sampling with uniform distribution. Then, a temporary population is created by using reproduction operator such as mutation and crossover. The fitness of all chromosomes (individuals) in this temporary population is calculated, and finally selection is performed to generate the new population. These consecutive phases are accomplished iteratively until certain stopping criterion is met.

3.2.1 Mutation

A trial vector for each chromosome is generated by mutating a target vector with a weight differential and used in the crossover operation to produce offspring. For each individual $x_i(t)$, a target vector, $x_a(t)$, and two individuals, x_b and x_c, are randomly selected from the population such that $i \neq a \neq b \neq c$ and then the trial vector, $u_i(t)$ is produced as follows:

$$u_i(t)=x_a(t)+\beta(x_b(t) - x_c(t)) \quad (2)$$

where the scale factor, $\beta \in (0, \infty)$, controls the amplication of the differential.

3.2.2 Crossover

A offspring vector, $y_i(t)$ is generated by recombining the trial vector, $u_i(t)$, and the parent vector, $x_i(t)$, as follows:

$$y_i(t) = \begin{cases} u_{ij}(t) & \text{if rand}(j) \leq CR \\ x_{ij}(t) & \text{otherwise} \end{cases} \quad (3)$$

where $x_{ij}(t)$ refers to the j^{th} element of the vector $x_i(t)$, rand(j) $\in$ U(0,1), and CR $\in$ (0,1) is the crossover rate.

3.2.3 Selection

The selection operation is applied to determine which individuals will participate in the mutation operation to generate trail vectors and to decide which of parent and offspring will remain in the next generation. To construct the new population for the next generation, the selection is performed based on the fitness of parent and offspring. The one with better fitness will remain in the new population.

3.3. Apache spark

Spark, in-memory distributed data processing engine intended for large-scale data analysis [4], is easy to use, general-purpose, fast, fault tolerant and scalable. The primary goal of designing Spark is to support a wider range of applications that use iterative algorithms, interactive data analysis and stream processing. The main concept of Spark is the distributed memory abstraction called Resilient Distributed Datasets (RDDs). An RDD is an irreversible subdivided collection of data objects distributed over the nodes of a Spark cluster. RDDs can be created from data on a storage space or other existing RDDs through transformation and action operations. Transformations are used to create new RDDs from existing dataset and actions yield the final result of a computation on RDDs. Apache Spark takes the master/slave architecture, and a spark cluster consists of driver program, cluster manager and worker nodes. The overview of the master/slave architecture on Spark is described in Fig. 1.

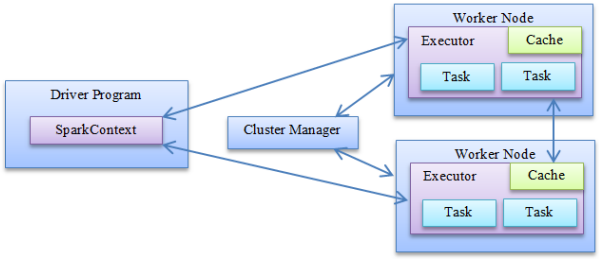

Fig. 1: The overview of the master/slave architecture on Spark [4]

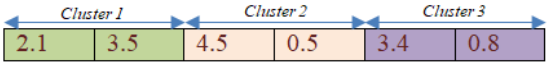

Fig. 2: Example of centroid-based representation

3.4. Parallel differential evolution for clustering (PDEC)

With the purpose of solving hard partitioning clustering problem based on the differential evolution algorithm, each cluster solution is encoded as a chromosome of the population by using a real-valued vector. In centroid based representation, individuals are characterized by the coordinates of centroids. The size of an individual in the population is the multiplication of cluster number and total number of features. If the i^{th} individual u_i encodes k cluster centroids in a d-dimensional space R^d, the first d points represent the first cluster solution; the following d points represent the second cluster solution and so forth. Fig. 2 shows an illustration of centroid based representation that encodes the cluster solution of 3 cluster centroids with 2

features. In order to evaluate the fitness of individuals, various cluster validity index can be used. The sum of squared error, SSE given in the equation (1) is used in this work.

The proposed PDEC algorithm is based on the master/salve architecture of Spark and exploits the power of RDD abstraction. To implement DE for clustering based on Spark, the dataset to be clustered is loaded into RDDs. The initial population is created by sampling data instances randomly from RDDs. As the first parallel stage, the fitness of all chromosomes in the initial population are evaluated in worker nodes and then, sent back the fitness results to the driver. In order to perform DE operations in parallel, the driver choose target vector and other two individuals for each member of the current population and then, distribute all members with selected vectors to the workers that accomplish reproduction and selection operation to generate new population. After all workers completed the process, the driver collects the populations from them to create a new population for the next generation. The process will be performed until the maximum iteration is reached and finally the best chromosome will be the cluster solution for the given dataset. The pseudocode of the proposed algorithm is presented in Fig. 3.

```
Algorithm: PDEC
Input: Dataset D, Number of clusters, Number of population, Maximum number of iterations (maxIt),
Scaling Factor, Crossover rate
Output: Cluster solution
1.  Create RDDs
2.  Initialize population P by sampling data instances randomly from RDDs
3.  Evaluate the fitness of each individual p from population P in parallel way
4.  for i=1 to maxIt
5.      for each individual p in population P do in parallel
6.          trialVector ← mutation(p)  // execute mutation operation according to eq (2)
7.          offspringVector←crossover(p) // execute crossover operation  according to eq (3)
8.          newVector ← selection(p)
9.      End for
10.     P ← newVector.collect()
11. End for
```

Fig. 3: Pseudocode of PDEC.

4. Experimental Result

To evaluate the effectiveness of the proposed approach, the experimental test was conducted on the commonly used UCI machine learning datasets that are shown in Table 1. For the testing of PDEC, the best chromosome from the current population is selected as the target vector for mutation operation, the scaling factor is set to 0.24, the crossover rate is 0.4, the number of population is 100 and the number of maximum iteration is 200. In Table 2, the total intra-cluster distance of the proposed approach is compared to the results of DE/best/1 and four well-known population-based algorithms in [12]; the last column is obtained from this experiment. According to Table 2, the proposed approach obtained better results than other methods on iris, glass, and breast cancer dataset, while PSOAG is the best one for wine dataset. In Table 3, the result of PDEC is compared to the results of other clustering algorithms based on Spark [13]. From Table 3, PDEC is comparable to other approaches.

Table 1. Four commonly used datasets

Dataset	No. of attributes	No. of instances	No. of Class
Iris	4	150	3
Wine	13	178	3
Glass	9	214	6
Breast Cancer	9	699	2

Table 2. Total intra-cluster distance

Dataset	DE/best/1	ACO	ABC	PSO	PSOAG	PDEC
Iris	96.70	100.67	101.00	104.45	96.97	**95.34**
Wine	16302.01	16300.71	16506.75	16303.16	16296.30	16297.25
Glass	266.53	226.41	297.10	324.30	244.99	**218.03**
Breast Cancer	2974.10	3376.20	3102.63	4024.79	2984.24	**2964.63**

Table 3. Comparison between PDEC and other clustering algorithms on Spark

Dataset	K-means	GA-Kmeans	PSO	PDEC
Iris	102.15	97.08	101.46	**95.34**
Wine	16930.21	16440.13	16315.54	**16297.25**

5. Conclusion

Clustering is an essential tool for data analysis. Differential Evolution (DE) can be exploited as an alternative solution for clustering problems since it is able to find more compact clusters than other traditional clustering algorithm. Apache Spark is a powerful data processing framework for large-scale analysis that uses iterative algorithms. This paper proposed a parallel DE algorithm based on Spark framework to solve the massive data clustering problem. The experimental results showed that the proposed approach is effective and comparable to other existing clustering approaches on Spark. As future work, an initialization method will be adapted to generate high-quality chromosomes for initial population, and the impact of different DE variants and parameter settings on clustering results will be analysed. As well as, the speed of convergence and the robustness will be considered for the performance of the proposed approach.

6. References

[1] R. Xu, D. Wunsch, Survey of clustering algorithms. IEEE Trans Neural Netw. 2005, 16(3):645–678

[2] E. Falkenauer, Genetic Algorithms and Grouping Problems. John Wiley & Sons, 1998

[3] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. P. L. F. de Carvalho, A Survey of Evolutionary Algorithms for Clustering, IEEE Transactions on Systems, Man, and Cybernetics. 2009, 39(2):133 – 155

[4] https://spark.apache.org

[5] S. Paterlini, T. Krink, Differential evolution and particle swarm optimisation in partitional clustering. Computational Statistics & Data Analysis. 2006, 50(5):1220–1247

[6] M.G.H. Omran, A. P. Engelbrecht, A. Salman, Differential evolution methods for unsupervised image classification. IEEE Congress on Evolutionary Computation. (2005)

[7] W. Kwedlo, A clustering method combining differential evolution with the K-means algorithm. Pattern Recognition Letters. 2011, 32(12): 1613-1621

[8] Y. Tian, D. Liu, H. Qi, K-harmonic means data clustering with Differential Evolution. International Conference on Future BioMedical Information Engineering (FBIE). (2009)

[9] M. Daoudi, S. Hamena, Z. Benmounah, M. Batouche, Parallel Differential evolution clustering algorithm based on MapReduce. International Conference of Soft Computing and Pattern Recognition (SoCPaR). (2014)

[10] V. Santhi, R. Jose, Performance Analysis of Parallel K-Means with Optimization Algorithms for Clustering on Spark. International Conference on Distributed Computing and Internet Technology, ICDCIT. 2018, pp. 158-162

[11] A. P. Engelbrecht, Computational Intelligence-An Introduction, Second Edition, John Wiley & Sons Ltd, England, 2007.

[12] X. Wan-li, Z. Ning, M. Shou-feng, M. Xue-lei, A. Mei-qing, A dynamic shuffled differential evolution algorithm for data clustering. Journal Neurocomputing. 2015, 158(C): 144-154

[13] T. Chun-Wei, L. Shi-Jui, W. Yi-Chung, A parallel metaheuristic data clustering framework for cloud. Journal of Parallel and Distributed Computing. 2018, 116: 39-49