Large-scale Agent Data Partitioning Based on DensityRepel-K_medoids

Lingjuan Wu*, Jie Liang and Bo Li

The School of Electronic and Information Engineering, Xi’an Jiaotong University, Xi’an, Shaanxi, 710049, P.R. China

*wulingjuan0910@stu.xjtu.edu.cn

Abstract. Large-scale Agent data partitioning is the premise of parallel distributed computing in the process of ABMS (Agent-based Modeling and Simulation). Based on the distance-based K_medoids clustering algorithm, this paper proposes an improved K_medoids algorithm (DensityRepel-K_medoids), which is implemented by high-performance programming language X10 and applied to large-scale Agent data partitioning simulation based on distance interaction. The DensityRepel-Kmedoids algorithm first determines the density value and the repulsion value of each Agent data in the Agent data set, and secondly pre-selects the cluster center according to the density value and the repulsion value of each Agent data, and finally uses the pre-selected cluster centers as the initial cluster centers for continuous iterative clustering until convergence. The algorithm avoids the defects of K_means clustering algorithm sensitive to outliers, and avoids the shortcomings of K_medoids clustering algorithm for large-scale data processing. By comparing and analyzing the simulation experiments of Agent data sets of different scales, the algorithm presents a better performance.

1. Introduction

ABMS (Agent-Based Modelling and Simulation) uses interactive, autonomous, and social agent to describe complex distributed intelligent systems. As the number of agents increases, the communication pressure and coordination difficulty between agents will also increase. Therefore, parallel distributed computing is important in this issue, and large-scale Agent data partitioning is the key to realize parallel distributed computing in the process of ABMS. On the current stage [1], the research on large-scale Agent data partitioning is still in the development period about the computer simulation field. At present, there are many researches on Agent data partitioning based on K_means clustering algorithm and K_medoids clustering algorithm, but they all have certain defects: K_means algorithm is more sensitive to outliers. Compared with the K_means algorithm, the K_medoids algorithm has lower sensitivity to outliers, but the calculation is much larger than Kmeans, which is generally suitable for small-scale data.

Therefore, this paper proposes an improved K_medoids algorithm (DensityRepel-K_medoids). Compared to K_means algorithm and K_medoids algorithm, the DensityRepel-K_medoids algorithm not only reduces the sensitivity to outliers, greatly reduces the amount of computation. It makes a reasonable and stable partitioning of large-scale Agent data, at the same as, it greatly improves the efficiency of large-scale Agent data partitioning.
2. K_medoids algorithm
The K_medoids algorithm is a derivative of the K_means algorithm. Compared to K_means, the K_medoids algorithm is greatly reduced by the influence of outliers, but the consequent drawback is that calculation is much larger than K_means. In the following step, K_means and K_medoids are compared [2-4], and the main differences, running speed, robustness, and data scale are analyzed, as shown in Table 1.

| Algorithm | K_means | K_medoids |
|-----------|---------|-----------|
| Algorithm step | a. K centers were randomly selected. b. Mark the clusters with centers, Calculate the distance from each agent data point to the center. c. Divide the data points into the cluster of the nearest center marked, Form K clusters. d. According to the classified cluster, the center is recalculated in each cluster: the average of all the data points in the cluster is selected as the new center. e. The iteration is repeated until the deviation is less than the set value. | a. K centers were randomly selected. (Centers must be some data point). b. Mark the clusters with centers, Calculate the distance from each agent data to the center. c. Divide the data points into the cluster of the nearest center marked, Form K clusters. d. According to the classified cluster, the center is recalculated in each cluster: the data point with the smallest sum of Manhattan distances from all the data points in the cluster is selected as the new center. e. The iteration is repeated until the deviation is less than the set value. |
| Main differences | The centroid is the average of all data points and may be a point that does not exist in the data points. | The centroid must be a data point that exists in the data points. |
| Running speed | Time complexity is $O(n)$, $n$ is the number of data points. It is more efficient and scalable for larger data. | Time complexity is $O(n^2)$, $n$ is the number of data points. Running slower because the time to update the centers is larger. |
| Robustness | Poor, affected by the outliers. | Better. |
| Data scale | Can handle large scale data. | More suitable for small sample data, the time complexity is larger. |

3. DensityRepel-K_medoids
Based on the distance-based K_medoids clustering algorithm, this paper proposes an improved K_medoids algorithm (DensityRepel-K_medoids). The DensityRepel-Kmedoids algorithm first determines the density value and the repulsion value of each Agent data in the Agent data set, and secondly pre-selects the cluster center according to the density value and the repulsion value of each Agent data, and finally uses the pre-selected cluster centers as the initial cluster centers for continuous iterative clustering until convergence.

First, determine the density value $D_i$ and the repel value $R_i$ of agent data $i$ in the Agent data set. The density value $D_i$ of the data point $i$ is [5-6]:

$$D_i = \frac{\sum_{j \in N(i)} d(j,i)}{|N(i)|}$$

$$R_i = \sum_{j \in N(i)} d(j,i)$$

where $N(i)$ is the neighborhood of data point $i$. The repulsion value $R_i$ is the sum of distances from data point $i$ to its neighbors.
\[
D_i = \sum_j \chi(d_{ij} - d_c)
\]

agent data \( j \) and agent data \( i \) are not equal, and both belong to the data set; \( \chi(x) \) is a function that represents \( x < 0 \), \( \chi(x) = 1; x \geq 0 \), \( \chi(x) = 0 \); \( d_{ij} \) represents the distance from agent data \( j \) to agent data \( i \); \( d_c \) represents the distance threshold; That is, \( D \) represents the number of agent datas whose distance from the agent data \( i \) is less than \( d_c \).

Appropriate distance threshold \( d_c \):\( d_c \) is a distance threshold and play an important role in the implementation of the algorithm. This parameter is used to calculate the density value \( D \) for each agent data. During the process of large-scale Agent data partitioning, We need to determine the number of partitions based on the number of processors available. To ensure load balancing across processors and reduce communication interactions, This paper uses a method to select the appropriate \( d_c \). Assume that the number of available machines is \( k \), data set range is \( N \times M \). The \( d_c \) is:

\[
d_c = \min(N, M)/k
\]

\( \min(N, M) \) represents the minimum of \( N \) and \( M \); \( k \) represents the number of processors available.

The repel value \( R \) of the data point \( i \) is [5-6]:

\[
R_i = \begin{cases} 
\min(d_{ij}), & \exists j: D_j > D_i \\
\max(d_{ij}), & \forall j: D_j < D_i
\end{cases}
\]

agent data \( j \) and agent data \( i \) are not equal, and both belong to the data set; \( \exists \) agent data \( j \), agent data \( j \) has a density value \( D_j \) greater than the density value \( D_i \) of agent data \( i \), \( R_i \) represents the closest distance between agent data \( i \) and the agent data with a density value greater than agent data \( i \)'s density value \( D_i \); \( \forall \) agent data \( j \), its density value \( D_j \) is smaller than the density value \( D_i \) of agent data \( i \), and \( R_i \) represents the largest distance between the agent data \( i \) and the agent data from the data set.

Secondly, pre-select the cluster center according to the density value \( D \) and the repulsion value \( R \) of each agent data. Constructor \( DR_i = \rho * D_i + \delta * R_i \), Select agent datas with larger density value and larger repel value, that's \( DR_i \) is larger. The value of \( \rho + \delta \) is 1, \( \rho \) and \( \delta \) are based on a large number of experiments, and the experimental results are shown in Figure 1.

![Figure 1. Effect of \( \rho \) and \( \delta \) values on the number of iterations.](image-url)
Finally, the pre-selected cluster centers are used as the initial cluster centers for continuous iterative clustering until convergence.

Algorithm steps:

1) Data initialization
   • Setting the number of clusters according to the number of available processors;
   • Calculating the density value \( D_i \) and the repel value \( R_i \) of each agent data \( i \) from the data set;
   • Construct \( DR_i = \rho * D_i + \delta * R_i \), where \( \rho \) and \( \delta \) are the influence factors, and select agent datas that \( D_i \) and \( R_i \) are both larger (\( DR_i \) is larger) served as pre-selected cluster centers from data set;
   • The pre-selected clusters centers are taken as the initial cluster centers.

2) Forming the initial cluster
   • The non-centrographic agent datas are divided into cluster of the center marked that is the most closest to them, and the loop operation is performed to complete the first division of all non-centrographic agent datas.

3) Update the cluster centers
   • Recalculate the new center in each of the formed clusters. And within the cluster, choose the agent data that smallest sum of Manhattan distances between it and agent data of all agent datas in the cluster as the new center.
   • Divide all non-centrographic agent datas into cluster of the center marked that is the most closest to them.
   • Loop operation until all the centers that have updated are the same as all the centers of the previous one.

4. Experiment

The cellular automaton model is a commonly used model in simulation. It is essentially defined in a cell space composed of discrete, finite state cells, and evolves in discrete time dimensions according to certain behavioral rules. It is widely used in the field of ecology, for example, wolf-sheep-grass ecosystem; in the field of physics, for example, the formation of dendrites, simulated snowflakes; in the field of transportation science, for example, the study of urban traffic flow; in the field of computer science and informatics, for example, the proliferation of network information.

The experiment uses the cellular automaton model as the research object. Compare \( K_{medoids} \) and \( \text{DensityRepel-K}_\text{medoids} \). The experimental environment is Inter® CoreTM i5-3230M CPU@2.60 GHz, 4GB RAM, 400GB hard drive, 24 core, X10-dt development environment. The initial cell (agent data) is generated using a random number. The algorithm clustering results were evaluated by the number of iterations and simulation time, as shown in Table 2.

| Table 2. Comparison of two clustering algorithms \( K_{medoids} \) and \( \text{DensityRepel-K}_{medoids} \). |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| Data set range  | Valid dataset   | Number of iterations | Running time/ms |
|                 |                 | \( K_{medoids} \) | \( \text{DensityRepel-K}_{medoids} \) | \( K_{medoids} \) | \( \text{DensityRepel-K}_{medoids} \) |
| 100*100         | 6000            | 9                | 7               | 87              | 79              |
| 200*200         | 32000           | 7                | 5               | 762             | 432             |
| 300*300         | 54000           | 7                | 3               | 2017            | 1455            |
| 400*400         | 96000           | 6                | 4               | 4205            | 3761            |
| 500*500         | 150000          | 8                | 5               | 8437            | 5645            |
It can be seen from Table 2 that compared to K_medoids, the DensityRepel-K_medoids algorithm presents a better performance, which reduces the number of iterations and reduces the running time.

5. Conclusions and prospects

The DensityRepel-K_medoids algorithm, that proposed in this paper, compared to the K_means algorithm, it reduces the sensitivity of outliers; compared with the K_medoids algorithm, it reduces the number of iterations, which reduces the running time and weakens the shortcomings of K_medoids' slow processing of large-scale data. The algorithm is applied to the partitioning of large-scale agent data, which improves the efficiency of partitioning and explains the feasibility of the DensityRepel-K_medoids algorithm.

The current partitioning algorithm is just a master machine single-thread to complete the division of all agent data. From the perspective of algorithm optimization, when the number of agents gets larger and larger, the time and space overhead of the clustering-based partitioning process will become longer and longer, and the partitioning process can be parallelized in subsequent work.

Acknowledgments

In the process of completing the paper, I got technical guidance of my teacher Bo Li’s and senior school mate Jie Liang’s. As the same as, my classmates around me also provide great help. Thank you very much for the guidance of the teacher, the help of the senior school mate and the help of my classmates around me.

References

[1] Wang Y, Lees M and Cai W. 2009 Cluster based partitioning for agent-based crowd simulations[C]. Winter Simulation Conf, 1047-1058.
[2] Madhulatha T S . Comparison between K-Means and K-Medoids Clustering Algorithms[J]. 2011.
[3] Meng Y, Luo K, Liu J H. An Improved K-Medoids Clustering Algorithm[J]. Advanced Materials Research, 2012, 562-564:2106-2110.
[4] Arora P , Deepali D , Varshney S . Analysis of K-Means and K-Medoids Algorithm For Big Data☆[J]. Procedia Computer Science, 2016, 78:507-512.
[5] Alex R and Alessandro L 2014 Clustering by fast search and find of density peaks[J]. Science, 344(1492): 1492-1496
[6] Zheng J , Sun H , Zhang M , et al. Research on Optimization of Clustering by Fast Search and Find of Density Peaks[C]. Web Information Systems & Applications Conf. IEEE, 2017.