Model Training Task Scheduling Algorithm Based on Greedy-Genetic Algorithm for Big-Data Mining

Yiqi Wang1*, Yipin Sun1, Ziwei Zhang2
1 National University of Defense Technology, Changsha, Hunan
2 31131Troop of People’s Liberation Army of China, Nanjing, Jiangsu
*Corresponding author’s e-mail: wangyiqi12@nudt.edu.cn

Abstract. With the coming of the big data age, data mining attracts more and more attention from all trades and professions. Due to the vast computation cost of data mining, the public service platform for big data mining has become the urgent needs, especially for the model training tasks. In this way, how to perform this kind of task scheduling becomes critical. This paper focuses on the assignment of tasks on multiple computing resources to optimize the total operation time. Firstly, a task scheduling algorithm based on the greedy and genetic algorithm is proposed to set the computation resource requirement for each task. Moreover, a greedy strategy is used to decide the task operation order and the assignment mapping between the tasks and the computation resources. Finally, the proposed algorithm proves to be efficient by several experiments.

1. Introduction
The rapid development of information and communication technology has led to explosive growth in global data volume. Big data mining and analysis have brought more data value, which has become a strategic consensus in various fields of national economy and people's livelihood, and the demand for data analysis shows explosive growth [1]. It is worth pointing out that the computational cost of big data mining model training is very large, which will form a potential resource threshold and restrict the application of realistic applications.

More specifically, from the perspective of computational overhead, the model training and application of data mining has significant imbalance: from the perspective of a single model, the complexity of model training (optimal solution) is usually much higher than the model application (operation). Taking deep learning as an example, the cost of model training is several times larger than the cost of model application [2]; from the whole process of model training, the selection of models and the adjustment of parameters need repeated iterations, taking random forest as an example. There are more than a dozen parameters involved in the model. In order to make the model work best, iteration is needed to determine the optimal parameter setting [3]. Therefore, for the application unit of data analysis, the computational platform for building big data mining model training is costly. At present, high-performance computing and cloud computing technologies tend to improve. Based on this, building a public support platform for big data mining model training has high research value and application prospects.

It is worth pointing out that from the perspective of computing resources, the training task of big data mining model is more complex, and higher requirements are imposed on the task scheduling of the supporting platform. The existing research is mainly for grid computing, cloud computing, the
essence of which is to split the computing task into several subtasks. Usually, the subtask is only allowed to execute on one computing resource, and each subtask is in each resource. The execution efficiency is known. The goal of task scheduling is to schedule subtasks to different computing resources to achieve global optimization goals. In comparison, the computational process of model training tasks for big data mining is highly dependent, the parallel architecture is complex, there is no universal method for sub-task segmentation, and the number of iterations in the solution process is too large. It is difficult to predict the calculation of sub-tasks in each iteration phase. Overhead. Therefore, the resource configuration is suitable for one-time configuration. Before the end of the task, resource recovery and dynamic adjustment are not performed, and the nonlinear relationship between resource allocation and operational efficiency is more significant, which makes the resource allocation uncertainty more powerful. Overall, the task scheduling for big data mining model training consists of three levels: the first is to allocate the computing resources occupied by the tasks, the second to dispatch the tasks to multiple different computing resources, the third is the scheduling of the task execution order. The task scheduling for the training of big data mining models has more uncertain factors than traditional task scheduling, and it is difficult to apply the existing results directly.

Aiming at the above problems, we propose a task scheduling algorithm based on a greedy algorithm and genetic algorithm with the task's total completion time as the global optimization goal under the condition of multi-task and multi-resources. The effectiveness of the algorithm is verified by experiments.

2. Related Work

Task scheduling has been applied in many areas, such as grid computing, cloud computing, and so on. In grid computing, task scheduling is oriented to heterogeneous platforms, usually in distributed, parallel mode, regardless of scheduling within the grid nodes, with strong scalability and adaptability [4]. In cloud computing, cloud resources vary greatly, and users' demand for cloud resources has diversity and preference. Scheduling needs to consider more aspects, such as service provider revenue, energy cost, and so on[5].

At present, most cloud computing platforms use Google's Map/Reduce programming model for parallel computing of large-scale data. The task scheduling algorithms commonly used in cloud computing include a genetic algorithm, particle swarm optimization algorithm, ant colony algorithm, Max-Min, Min-Min and so on. Literature [6] uses a three-stage-selection and a total-divide-total genetic strategy to improve the genetic algorithm. Literature [7] proposed a discrete particle swarm optimization algorithm for multiple quality of service requirements. Literature [8] proposes an improved genetic algorithm considering both time and cost constraints. Literature [9] combines cultural algorithms and genetic algorithms to perform task scheduling and achieve specific effects. Literature [10] proposes an improved genetic algorithm considering the total time, average task time and cost constraint.

Compared with the three levels of task scheduling considered in this study, the traditional work ignores the first level and the third level. Moreover, only the task scheduling on a single computing resource is discussed at the second level. This is different from the research goal of this paper. Given the application effect of genetic algorithm in dynamic programming, this paper uses a genetic algorithm to calculate resource allocation and uses greedy strategy to divide computing resources and execute sequential scheduling.

3. Task Scheduling Problem Definition

This paper discusses the allocation of big data model training tasks on multiple resources, and divides the total computing resources into different computing nodes, and makes the following assumptions:

- Different computing nodes can contain multiple computing resources, but computing resources within nodes can no longer be split.
A task can occupy a single computing resource or multiple computing resources at the same
time, but the computing resource allocation is determined by the computing node. The number
of tasks is greater than the number of resources.

- The running time of the task is determined by the number of computing resources occupied
and is independent of the computing node itself.
- The number of tasks is greater than the number of resources.
- The execution time of the task is much larger than the running time of the scheduling
algorithm, regardless of the overhead of the algorithm.

Use $N$ to indicate the number of tasks, $M$ to represent the number of computing resources, and use
$ETC_{N \times M}$ matrix to represent each task. Run time on different amounts of computing resources. $E_{ij}$
indicates the time required for the $i$-th task to run when the number of computational resources is $j$.
The task is represented by $T_i$, and the number of computing resources is represented by $P_j$, where $i \in (1, N), j \in (1, M)$.

$$ETC_{N \times M} = \begin{bmatrix}
E_{11} & \cdots & E_{1m} \\
\vdots & \ddots & \vdots \\
E_{n1} & \cdots & E_{nm}
\end{bmatrix}$$

Based on the above assumptions, the task scheduling problem can be defined as how to allocate
tasks to various computing resources reasonably and efficiently, so that the total time required for the
task to complete is shorter.

4. Greedy-Genetic Algorithm

Aiming at the allocation of multi-resources, considering parallelism and resource constraints, we
propose a greedy genetic algorithm. The main idea is to use a genetic algorithm to allocate task
resources, and encode the number of resources required for each task, through selection, crossover,
and mutation. The genetic operation finds the optimal number of resources for each task. The adaptive
function as the optimization target in the selection process is to calculate the total time required for the
task to run through the greedy algorithm.

4.1 Encoding and Decoding

The encoding of chromosomes is usually divided into direct encoding and indirect encoding. This
paper uses the form of indirect coding to encode the number of resources occupied by tasks. The
length of the chromosome is the number of tasks, and the value of each gene in the chromosome is the
number of resources required for the task at that location, as shown in Figure 1.

![Figure 1. Encoding](image)

Suppose there are five tasks, three computational resources, the chromosome length is five, and
each gene takes a random number between 1 and 3, such as randomly generating the following
chromosomal code:

```
{2,1,3,2,1}
```

Then this chromosome represents the first task running under two resources, the second task
running under one resource, and so on, the 5th task running under one resource.

By decoding, the assignment of tasks to different resources is obtained. By using the running time
of each task in the $ETC$ matrix on various resources, the total time of completion is calculated by the
greedy algorithm.
4.2 Initialization of the Population
The population size is S, the number of resources is M, the number of computing nodes is Q, and the number of tasks is N. The initialization is described as random generation of S chromosomes by the system, the length of the chromosome is N and the random number between the gene values [1, M]. The number of computing resources on the computing nodes is [1, M] and the sum is equal to the total number of resources M.

4.3 Adaptive Function Model
The genetic algorithm adaptive function can be said to be the optimization target and the individual selection. Individuals with higher fitness values have a higher probability of inheriting to the next generation; while individuals with smaller fitness values have a lower likelihood of inheriting to the next generation. This paper considers the total completion time of the task as an adaptive function, which is calculated by the greedy algorithm. The process shows in Table 1.

| Table 1. Greedy algorithm process |
|----------------------------------|
| **Input:** distribution of the decoded task, the number of occupied computing resources, and the number of computing node resources (P, T, Q) |
| **Step1:** initialize a time series. |
| **Step2:** Select the task with the largest number of resources to run, and judge whether the currently occupied resources are satisfied according to the number of resources on computing nodes. If yes, the ETC matrix is used to record the end time \( t_1 \), and the third step is performed; if not, the number of resources satisfying the condition and greater than the current number of occupied resources is selected as the new resource allocation situation, and the second step is repeated. |
| **Step3:** Calculate the number of remaining resources, select the maximum number of resources that can satisfy the total number of resources, and run the task at the end time \( t_2 \). |
| **Step4:** Determine the resource status and task status of the most recent end time according to the time series. |
| **Step5:** Repeat steps 2 and 3 until the task is finished. |
| **Output:** task completion time \( \text{complete}(T) \) |

Therefore, the definition of the adaptive function is:

\[
\text{Fitness}(T) = \frac{1}{\text{complete}(T)} \quad (1)
\]

4.4 Genetic Operation

4.4.1 Selection
The selection operation selects individuals with strong adaptability in the population according to the principle of “survival of the fittest” to generate new populations. Simply put, the more adaptive individuals are more likely to be selected to inherit to the next generation, and vice versa. In this paper, we use the adaptive function to calculate the selection probability of each according to formula (2), and select it by “rotation selection method.”

\[
P(i) = \frac{\text{Fitness}(i)}{\sum_{j=1}^{S} \text{Fitness}(j)} \quad (2)
\]

4.4.2 Cross-Over and Mutation
Crossover operation is the primary method to generate new individuals. It determines the global search ability of genetic algorithm. The mutation operation can improve the local search ability of genetic algorithm, maintain the diversity of the population and prevent premature phenomenon. This paper uses the probabilistic approach of fixed crossover and mutation operations in the underlying genetic algorithm.
The crossover selects a random location in the chromosome, divides the chromosome into two parts, and then crosses the subsequent chromosomes. The mutation is to replace the gene value at a random position in the chromosome to achieve the purpose of variation.

5. Experimental Results and Analysis
In this paper, simulation experiments are carried out to verify the algorithm, custom population size, number of resources, number of tasks, crossover and mutation probability, ETC matrix. The experiment uses the total completion time of all tasks as the evaluation index and verifies the effectiveness of the algorithm by changing the task time. Table 2 shows the values of the parameters at the beginning of the experiment.

| Name                        | Value(s) |
|-----------------------------|----------|
| Population Size(S)          | 60       |
| Number of Computing Resources(M) | 5       |
| Number of Tasks(N)          | 50,100,200 |
| Cross-Over Probability      | 0.8      |
| Mutation Probability        | 0.01     |
| Max Iterations              | 100      |

In the experiment, the total time of the scheduled tasks obtained according to the algorithm is tested when the number of tasks is 50, 100 and 200, as shown in Figure 2.

As can be seen from the figure, as the number of iterations increases, the task completion time gradually decreases and gradually approaches a value. It shows that the algorithm can seek an optimized solution at each iteration, and find an optimal solution as the result of the algorithm after several iterations.
For the convergence of the algorithm, the change of the fitness value when the number of tasks is 200 is as shown in Figure 3. It can be seen that the fitness value increases with the increase of the number of iterations and gradually converges.

![Figure 3. The fitness value when N=200](image)

6. Conclusion

Aiming at the characteristics of big data, with the advantage of cloud computing, this paper proposes a task scheduling algorithm based on a greedy and genetic algorithm, with the task completion time as the optimization goal, using the greedy algorithm to calculate the fitness, and then through the genetic algorithm. To optimize the allocation of computing resources. Experiments show that the algorithm has good convergence and can find an optimal solution.

References

[1] Wu L, Yuan L, You J. (2015) Survey of large-scale data management systems for big data applications. Journal of Computer Science and Technology, 30(1):163-183.

[2] Yang X Y, Zhang Z, Zhang W H, et al. (2016) Research on Deep Learning Acceleration Technique. Computer Systems & Applications.

[3] Breiman L. (2001) Random Forests. Machine Learning, 45(1):5-32.

[4] Mengmeng, Tinghuai Ma, Qiaoqiao Yan, Wenjie Liu. (2011) A survey on grid task scheduling. International Journal of Computer Applications in Technology, volume 41:303-309.

[5] Asemi R. (2013) A survey for independent task scheduling approaches in cloud computing. National Conference on Advances in Computer Science and Information Retrieval Approaches.

[6] Yao H. (2017) Cloud Task Scheduling Algorithm based on Improved Genetic Algorithm. International Journal of Performability Engineering, 13(7).

[7] Zhu Zongbin, Du Zhongjun. (2013) Improved GA-based task scheduling algorithm in cloud computing. Computer Engineering and Applications, 49(5):77-80.

[8] Li JW, Qu CW. (2016) Cloud Computing Task Scheduling Based on Cultural Genetic Algorithm. MATEC Web of Conferences, 40.

[9] Wang Yue, Liu Yaqiu, Guo Jifeng. (2017) QoS Scheduling Algorithm in Cloud Computing Based on Discrete Particle Swarm Optimization. Computer Engineering, 43(6):111-117.

[10] Ge J W, Yuan Y S. (2013) Research of Cloud Computing Task Scheduling Algorithm Based on Improved Genetic Algorithm. Applied Mechanics & Materials, 347-350:2426-2429.