Computers and Electrical Engineering 000 (2016) 1–10

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Cooperative ant colony-genetic algorithm based on spark[☆]

Dong Gaifang, Fu Xueliang[*], Li Honghui, Xie Pengfei

Inner Mongolia Agricultural University, Hohhot 010018, China

ARTICLE INFO

Article history:
Received 14 July 2016
Revised 28 September 2016
Accepted 28 September 2016
Available online xxx

Keywords:
Spark
Mapreduce
Ant colony optimization
Genetic algorithm
TSP

ABSTRACT

By taking full advantages of both the map and reduce function for the MapReduce parallel framework and the memory computation for the Spark platform, this paper designs and implements the algorithms for solving the traveling salesman problem based on ant colony algorithm on MapReduce framework and Spark platform. Next, adds the nearest neighbor selection strategy for choosing next city for the Spark platform ant colony algorithm, and combines it with genetic algorithm by using the optimal individual between ant colony algorithm and genetic algorithm, in order to update each other's best individual at the end of each iteration. Experimental results show that with the increase of ant colony size, compared to the stand-alone ant colony algorithm, MapReduce ant colony algorithm reflects the superiority of parallel computation; compared to the MapReduce ant colony algorithm, Spark platform ant colony algorithm reflects the superiority of memory computing. Cooperated with genetic algorithm, the solution has been improved significantly in its precision.

Combinatorial optimization problem [1] is an important branch of operations research, which involves sorting, filtering and other issues. The main objective is to find the optimal solution from the feasible solution set. Although the definition of combinatorial optimization problem is very simple, solving the optimal solution is very difficult. Due to that the solving process needs huge storage space and very long running time, so traditional computer is unlikely to be achieved, that is, the so-called "combinatorial explosion" phenomenon. Therefore, for the combinatorial optimization problem, how to use heuristic algorithm to solve the problem becomes the focus of attention. Heuristic algorithm is summarized according to the nature of some excellent population collaborative behavior. The outstanding representatives of this kind of algorithm are ant colony optimization [2–4], genetic algorithm (GA) [5] and particle group optimization algorithm (PSO) [6] etc.

Ant colony algorithm is a kind of bionics algorithm which is put forward to simulate the natural ants foraging behavior. Now it has been widely used in fault recognition [7], vehicle routing problem [8,9], the system identification [10], data mining [11,12], image processing [13–15], and other fields. The basic principle of ant colony algorithm is that when ants search path, it will secrete a certain amount of pheromones. Ant colony always communicate through the path pheromones in order to achieve the goal of optimal path. After the end of each iteration, the ant colony algorithm will volatilize pheromone on the path to form a positive feedback mechanism and to ensure that the ant colony can find the optimal path results. Since it appeared, there have been a large number of scholars putting forward the improvement strategies. However, in the treatment of medium and large scale TSP, there is still problem of long searching time and easily falling into local optimal solution [16]. Many researchers focus on the improvement of the shortcomings of the ant colony algorithm, such as easy stagnation, slow convergence speed and so on. Gambardella and other scholars proposed the Ant-Q algorithm, which can be

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Dr. Z. Zheng.
* Corresponding author.
E-mail addresses: fxliang@126.com, fuxl@imau.edu.cn (F. Xueliang).

http://dx.doi.org/10.1016/j.compeleceng.2016.09.035
0045-7906/© 2016 Elsevier Ltd. All rights reserved.

a good way to maintain the balance between knowledge discovery and knowledge utilization in the process of constructing the solution [17]. WJ Gutiahr provides a graph search ant system (Ant System Graph-based, GBAS), which has a certain probability of convergence to the optimal solution of the problem [18]. Wu Bin and Shi Zhongzhi proposed a meeting algorithm, which effectively improved the quality of the ant first traversal algorithm. Zhang Xuliang, Zhang Jinbin and Zhuang Changwen introduced the cooperative mechanism in basic ant colony algorithm, and proposed the enhanced ant colony algorithm based on cooperative learning. On the convergence of the ant colony algorithm, Hou Yunhe did a lot of related research and achieved some preliminary research results; Ding Jianli and Sun Xi combined genetic algorithm and ant colony algorithm integration with the theory of Markov stochastic process for the convergence of ant colony algorithm and made a lot of fruitful research.

To get rid of the defect of long searching time, in this paper, we design and implement the basic ant colony algorithm based on MapReduce parallel framework (MRACO). Compared with basic ant colony algorithm in the single node environment, the computational time has been greatly improved. While Spark is a parallel platform which is more suitable for iterative computation in recent two years. Therefore, we design the basic ant colony algorithm based on Spark platform (Spark-ACO). Experimental results show that the Spark-ACO takes less computational time than MRACO. In view of the defects of the ant colony easily falling into the local optimal solution, Spark-ACO is improved: on the one hand, the nearest neighbor strategy is used for ants to traverse the city node; on the other hand, ant algorithm and genetic algorithm are combined to improve the defects of local optimum for ant colony. The experimental results show that the improved hybrid algorithm has a great improvement on the accuracy of the solution.

The second section of the paper describes the implementation of basic ant colony algorithm; the third section discusses the design of ant colony optimization based on MapReduce (MRACO); in the fourth section, ant colony algorithm based on Spark (Spark-ACO) and its improvement have been explained; the fifth section is the experimental results and analysis, and the sixth section is the conclusion and discussion.

1. Ant colony optimization algorithm

The traveling salesman problem (TSP) is described as follows: A traveling businessman will visit n cities. He starts from one city, and visit the cities one by one-each city can only be visited once, at last, he returns to the starting city. The shortest path is required to traverse all the cities.

The mathematical model of ant colony algorithm for solving TSP problem can be described as follows: suppose there are n nodes in different cities, the m ants (k= 1,2,..., m) are put into different cities randomly selected. Each ant does a complete traversal of all the cities, returns to the starting city, and records the total length of the path and path length; During the ant colony searching the optimal path process, each ant has a tabu list $tabu_k(k=1,2,\ldots,m)$ to record the cities which have been passed. Every step of the ants must be based on the state transition probability $P^k{}_{ij}(t)$. $P^k{}_{ij}(t)$ indicates that the probability between the city j and the city i for ant k at t time, as shown in the formula (1):

$$\mathrm{p}_{\mathrm{ij}}^{\mathrm{k}}(t) = \begin{cases} \dfrac{\tau_{ij}^{\alpha}(ij) \times \eta_{ij}^{\beta}}{\sum\limits_{s\in allowed_k} \tau_{ij}^{\alpha}(ij) \times \eta_{ij}^{\beta}} & if\ j \in allowed_k \\ 0 & otherwise \end{cases} \tag{1}$$

$allowed_k$ in the formula (1) is the collection of cities which ant k allows to access at present; $\tau_{ij}(t)$ is the residual pheromone between city i and city j at time t; $\eta_{ij}(t)$ is the degree of expectation city i moves to the city j at time t, the value of $\eta_{ij}(t)=1/d_{ij}$, d_{ij} is path length between city i and j; α represents the information heuristic factor; β represents an expected heuristic factor.

In the end of each iteration, all the paths between the cities will be residual amount of pheromone. If not promptly volatilizing the pheromone, too much pheromone will be left on the path in the next iteration process. So the randomness of the ants selecting the path will be increased. Adjust the pheromone according to the formula (2) and (3), as shown below:

$$\tau_{ij}(t+n) = (1-\rho)\tau_{ij}(t) + \rho \cdot \Delta\tau_{ij}(t) \tag{2}$$

$$\tau_{ij}(t) = \sum_{k=1}^{m} \tau_{ij}^{k}(t) \tag{3}$$

ρ is the information pheromone evaporation coefficient in the formula (2) and (3) which range is (0,1), 1 - ρ is the pheromone residual factor; $\tau_{ij}(t)$ is the pheromone between city i and city j at time t; $\Delta\tau_{ij}(t)$ is the pheromone increment between city i and city j from t to $t+n$ time; $\tau^k{}_{ij}(t)$ is the pheromone for ant k generated at t to $t+n$ time between the city i and city j. When solving the TSP problem, it is better to use the Ant-Cycle model to update the pheromone on the path, so the paper uses the Ant-Cycle model, as shown in the formula (4):

$$\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L} & <i,j>\in Path_L \\ 0 & otherwise \end{cases} \tag{4}$$

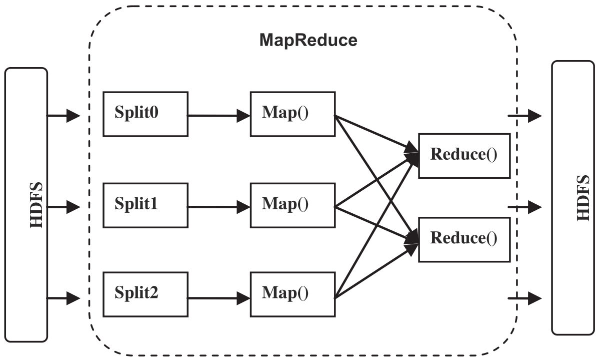

Fig. 1. The task processing figure of Hadoop.

2. Ant colony optimization based on mapreduce (MRACO)

Hadoop is one of the most commonly used platform to deal with big data business, and its core module is composed of distributed file system (Distributed File System HDFS, Hadoop) and MapReduce parallel computing model. MapReduce can let users develop distributed applications according to their demand in the cloud computing cluster so as to handle large data sets in parallel; HDFS can solve massive data distributed storage and accessing problem—-generally refers to the GB or TB level file—and store redundant data to ensure data security. Therefore, Hadoop platform is widely used in the field of data processing and research in various fields. Large data files can be processed through the Hadoop cluster to obtain the corresponding results, and the process of its tasks is as shown in Fig. 1:

The design idea of MRACO algorithm is to use the MapReduce programming model to realize the parallelism of feasible solution of ant colony algorithm based on the basic ant colony algorithm. At the beginning of MRACO algorithm, it initializes city distance matrix, pheromone matrix and other parameters; when iteration process is started, the ant colony is evenly distributed in each map. In the map stage, each node stores a feasible solution which is built by themselves in the form of ⟨key, value⟩, that is $\langle k_m, p_m\rangle$. The key represents the number of ants and the value is the traversal path sequence.

MapReduce allows the user to specify a combine () function to process the output data of the map () function and then transfer them to the reduce () function so as to reduce the data transmission cost between cluster nodes. In the combine phase, each node of the cluster computes the path length of each ant's path. Then, transmits to the reduce () function in the form of $\langle k_m, v_m+p_m\rangle$.

In the reduce phase, the program will calculate the values of $\langle k_{best}, v_{best}+p_{best}\rangle$, where the key and value, respectively, indicate that the number of optimal ant and optimal path length with path sequence, and then update pheromone matrix according to the optimal path sequence. Then, MRACO algorithm goes to the next iteration and repeats until the end.

Algorithm steps are as follows:

Step 1: in the initialization phase, MapReduce programs read the coordinates of the TSP node from HDFS, calculate the city distance matrix distance[][], and then initialize pheromone matrix tao[][], pheromone volatilization coefficient ρ, the number of ants m, iterative number N_c, heuristic information factorα and expected heuristic factorβ.

Step 2: in the stage of map, in every iterative process, the map() function will distribute an ant in a random city node and calculate the transition probability from the city to all the other cities according to the formula (1). According to the roulette strategy, an ant selects the next city node. When all cities are visited, the ant number and feasible solution are transmitted to combine () function in the form of $\langle k_m, p_m\rangle$.

Step 3: in the combine stage, the feasible solution key value pair $\langle k_m, p_m\rangle$ of the ant colony is used as input, and then combine () function respectively calculates the path length of the sequence and gets key value pair $\langle k_m, v_m+p_m\rangle$. k_m is the ant number, and v_m+p_m refers to the feasible solution and the path.

Step 4: in the reduce stage, the $\langle k_m, v_m+p_m\rangle$ output by combine stage is used as input, and then reduce() function calculates and finds the $\langle k_{best}, v_{best}+p_{best}\rangle$, which are the optimal path ant and the optimal path length. Algorithm updates optimal path pheromone according to the formula (3) and (4). Finally, it updates the global pheromone matrix according to the formula (5), then writes the iterative optimal solution and path as the keys and value $\langle N_C+k_{best}, v_{best}+p_{best}\rangle$ in HDFS, thus the iteration ends.

$$\tau_{ij}(t+n) = (1-\rho)\tau_{ij}(t) + \rho\cdot\Delta\tau_{ij}(t) \tag{5}$$

3. Ant colony based on spark (Spark-ACO) and combined with genetic algorithm

3.1. Ant colony based on spark (Spark-ACO)

The design idea of Spark-ACO algorithm is as follows:

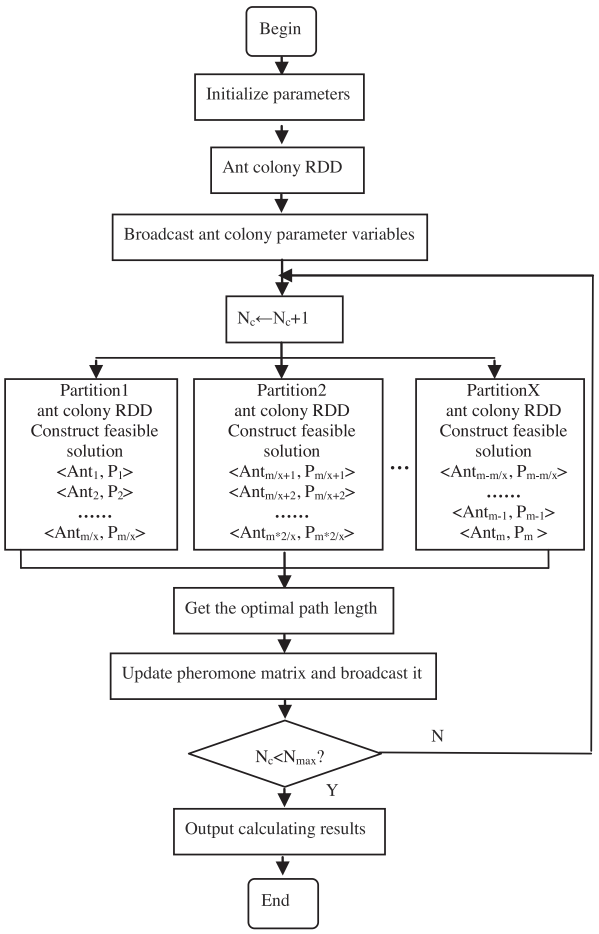
Begin
Initialize parameters
Ant colony RDD
Broadcast ant colony parameter variables
N_c←N_c+1
Partition1 ant colony RDD Construct feasible solution <Ant_1, P_1> <Ant_2, P_2> …… <Ant_m/x, P_m/x>
Partition2 ant colony RDD Construct feasible solution <Ant_m/x+1, P_m/x+1> <Ant_m/x+2, P_m/x+2> …… <Ant_m*2/x, P_m*2/x>
PartitionX ant colony RDD Construct feasible solution <Ant_m-m/x, P_m-m/x> …… <Ant_m-1, P_m-1> <Ant_m, P_m>
Get the optimal path length
Update pheromone matrix and broadcast it
N_c<N_max?
N
Y
Output calculating results
End

Fig. 2. The flow diagram of Spark-ACO.

(1) Ant colony algorithm must do lots of iterations to get the optimal solutions, and at the end of each iteration, it updates the pheromone matrix according to the best path results. Updated pheromone matrix is the key factor of ant colony traversal in the next iteration. In view of this, Spark-ACO algorithm uses the sharing mechanism of Spark platform. After the end of every iteration, it will update the pheromone matrix and transfer it to every node in the cluster as a broadcast variable, so as to be used in the next iteration. In addition, the city distance matrix in the traveling salesman problem is also transmitted to the cluster nodes by broadcasting mechanism, so as to realize the feasible solution for each node in the cluster.
(2) In ant colony algorithm in each iteration the process that ants build their own feasible solution is completely independent. In view of this, algorithm Spark-ACO will package ant colony to parallel RDD sets in each node in the cluster, of which every RDD represents an ant. Then, according to the function of Spark, a series of operations are designed to carry on the transformation and operation of RDD and to finally realize the whole process of the feasible solution of the ant colony represented by the RDD data set. The Spark program is divided into several partitions according to the number of cluster nodes, and the operation mode of RDD in each partition is completely parallel.

In addition, the process can also use some other function Spark provides to simplify programming process. For example, after each iteration, the ants in each node in the cluster building a feasible solution will converge to the master node. This program can use sortByKey () transfer function to composite the RDD data set, and then use fisrt() function to construct the iterative ant colony optimal path sequence directly. According to the above ideas, we can use the Spark programming model to achieve the parallel implementation of ant colony algorithm in the cloud computing cluster.

The specific process of Spark-ACO algorithm is shown in Fig. 2:

3.2. Nearest neighbor (NN)

In the running process of ant colony algorithm, the ants will select the next city according to the state transfer probability $P^k{}_{ij}(t)$. If the choice of the next city every time depends on the probability, it is easy to make the algorithm go into a local optimal solution. Therefore, the paper uses a nearest neighbor strategy: ants select the next city among the nearest *w* cities randomly. If all *w* cities have been selected, select one in the rest cities which have not be passed.

3.3. Combined with genetic algorithm

In order to improve the quality of the solution, genetic algorithm is integrated into the algorithm. After the genetic algorithm is integrated into the cluster, each single node also works according to the following steps:

1. After ant colony travels in each node, the traveling results of the city sequence will be treated as the initial population of genetic algorithm (GA). Individual number of the initial population is equal to the number of ants, and each individual is the tour city sequences of each ant. Each city sequence is corresponding to a cost value, and the cost value is individual fitness of genetic algorithm.
2. Run the selection, crossover and mutation operators of genetic algorithm (GA).
3. Calculate and record the smallest individual MinGeti of genetic algorithm (GA).
4. If the minimum touring cost of ant colony is less than the minimum individual fitness value of genetic algorithm (GA), i.e. *Lenth[NumShortPathing]* < *NextColony[MinGeti]*, and the smaller of the two values is less than the minimum value of last iteration, update the smallest individual chromosomal sequences of genetic algorithm (GA) with ant colony optimization path.

   Else, update ant colony optimization path with the smallest individual chromosomal sequences of genetic algorithm (GA).

5. If the number of iterations is a prime number, the main process collects the optimal solutions of other processes, calculates the process number with the minimum value, and broadcasts the minimum value and path.
6. The main process outputs the optimal sequence of the cities in this iteration.
7. Other processes update the global pheromone matrix according to the formula (2).
8. If the number of iterations is another prime number, part of the chromosome sequences of GA will be updated by part of the paths of ACO.
9. Repeat the process until the algorithm achieves the end condition.

4. Experimental results and analysis

4.1. Influence of spark cluster nodes on running time

When ant colony algorithm runs in cloud computing platform operation, communication collaboration between cluster nodes needs certain time cost. So for a particular size of ant colony, increasing cluster node number in a certain extent can reduce the running time of the algorithm. But when the cluster node quantity is greater than a critical value, the running time of the algorithm increases instead. In view of this, the paper designs the following experiments to analyze the influence of the number of cluster nodes on the running time of the ant colony algorithm.

The hardware environment of experiment is as follows: Lenovo sureserver of R680 G7, of which the server includes 4 processors and each has 8 nuclears, the frequency of CPU is 2.0GHZ, the memory capacity of server is 1024 G The software environment of experiment is as follows: Linux operating system of RHEL6.4, Hadoop-1.2.1 and JDK 1.6.0. Based on the hardware and software environments, we set up a Hadoop pseudo-distributed computing environment.

In the experiment, the parameters of ant colony algorithm based on past experience are set as follows: the number of iterations $N_C = 100$. The number of ants $m = 500$, the pheromone evaporation coefficient $\rho = 0.5$, the heuristic information factor $\alpha = 1.0$, the expected heuristic factor $\beta = 2.0$, and pheromone intensity $Q = 150$. The number of executor Spark cluster nodes are set to [1,5,10,15,20,25,30,35,40,45,50]. Experiments use TSPLIB library to provide example of the kroA100. For each executor value, we run Spark-ACO algorithm 15 times, and then compute the average time. The experimental results are shown in Fig. 3, and the time unit is in milliseconds (ms).

We can see from Fig. 3: when the cluster node executor is set to 1, that is, when the Spark cluster has only one node, the algorithm has the longest running time; when the number of cluster nodes is in executor [19] range, Spark-ACO algorithm running on the Spark cluster spends the least time; and when the executor number is greater than 30, the running time of Spark-ACO algorithm is significantly increased, which shows that when the number of cluster nodes is greater than a certain critical value, the cost of communication between nodes in cluster Spark-ACO algorithm will rise. Therefore, the increase of the number of cluster nodes can reduce the running time of the Spark-ACO algorithm. In case of ant colony size $m = 500$, Spark cluster node is set to [19] which is most time saving.

4.2. The time cost of ACO, MRACO and Spark-ACO

In order to verify the time efficiency of MRACO and Spark-ACO algorithms, the paper designs the following experiments to compare the running time of ACO, MRACO and Spark-ACO algorithms.

In this experiment, the algorithm MRACO is implemented in the Hadoop platform server, and the algorithm Spark-ACO is implemented in the Spark platform server, while ACO algorithm runs in a stand-alone mode of an arbitrary servers which

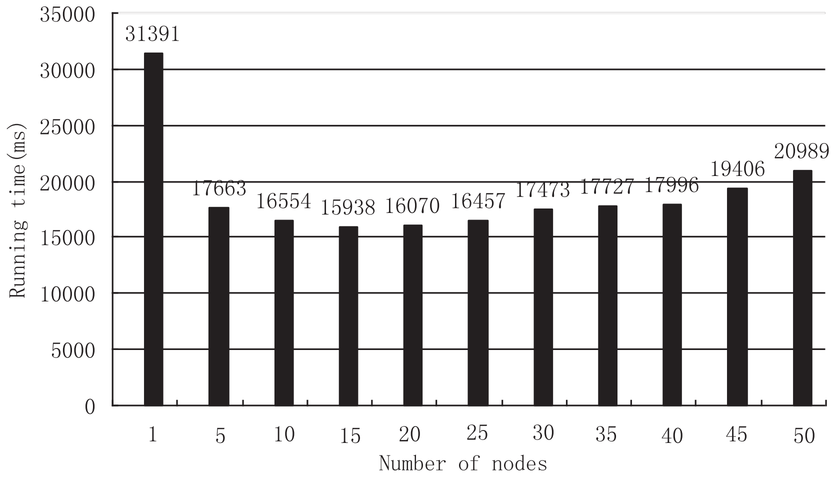

Fig. 3. The relationship between the number of cluster nodes and the running time.

Table 1
The running time of ACO, MRACO and Spark-ACO.

Ant size	ACO	MRACO	Spark-ACO
100	7397	12,012	10,857
300	21,878	15,534	13,316
500	35,773	18,922	14,905
700	50,901	23,653	16,941
900	66,819	29,404	17,369
1100	79,799	35,394	18,657
1300	94,837	40,934	19,692
1500	108,451	46,756	20,390

has the same configuration. The parameters of ant colony algorithm based on past experience are set as follows: the number of iterations $N_C = 100$, pheromone evaporation coefficient $\rho = 0.5$, the heuristic information factor $\alpha = 1.0$, the expected heuristic factor $\beta = 2.0$, and pheromone intensity $Q = 150$. The number of executors of the Spark cluster nodes and the number of Map are all set to 15. Then select the TSPLIB library provided by the kroA100 instance of the experiment. In addition, the size of the m value of ant colony is in [100, 300, 500, 700, 900, 1100, 1300, 1500]. Respectively run the three algorithms each 15 times for ant colony of each number, and average the running time (in millisecond), as shown in Table 1.

Table 1 shows: when the ant colony size is below 100, the running time of ACO is shorter than that of MRACO and Spark-ACO; when the colony size is equal to 500, the running time of Spark-ACO can be saved by 21.22% compared to the time cost of MRACO, and MRACO algorithm can save 47.1% than ACO; with the growth of colony size, the range that Spark-ACO saves time increasingly exceeds that of algorithm MRACO; when the colony size is equal to 1500, Spark-ACO can save 56.39% of the time compared to the algorithm MRACO, while MRACO saves 56.88% of the time compared to ACO. The running time of above three algorithms is drawn into a broken line, as shown in Fig. 4:

In summary, when the colony size is small, the basic ant colony algorithm ACO in stand-alone mode can save more time than MRACO and Spark-ACO algorithm. Because when the ant colony algorithm runs in a cluster, the starting cluster nodes and the consumption communication between nodes account for a large proportion of the overall operation time. With increased colony size, a cluster of parallel computation will gradually show the advantages over stand-alone mode. However, due to the complexity of computing tasks, the differences between MRACO and Spark-ACO has been exposed. The main reason is as follows. The MRACO algorithm is calculated based on the MapReduce framework, and the algorithm will construct a feasible solution for key value pairs stored and transmitted in the cluster nodes between the ants. Each iteration in the process of building a feasible solution needs a Map stage and a Reduce stage, in which Map stage records each ant optimization process, and Reduce phase calculates each ant searching results between Map and Reduce. These two stages inevitably need to transmit data and thus take a lot of time. On contrary, the Spark-ACO algorithm is calculated based on the Spark framework and packages the ant colony in flexible distributed data RDD Set. The ant colony RDD is evenly distributed in each node in the cluster, and makes good use of the characteristics of computational memory of spark platform. In each iterative process, ant colony constructs the optimal solutions through a series of RDD conversion operations. Therefore, Spark shows its advantages in computing based on memory and preventing data from landing. With the expansion of the colony size, the algorithm Spark-ACO will save more time than the algorithm MRACO.

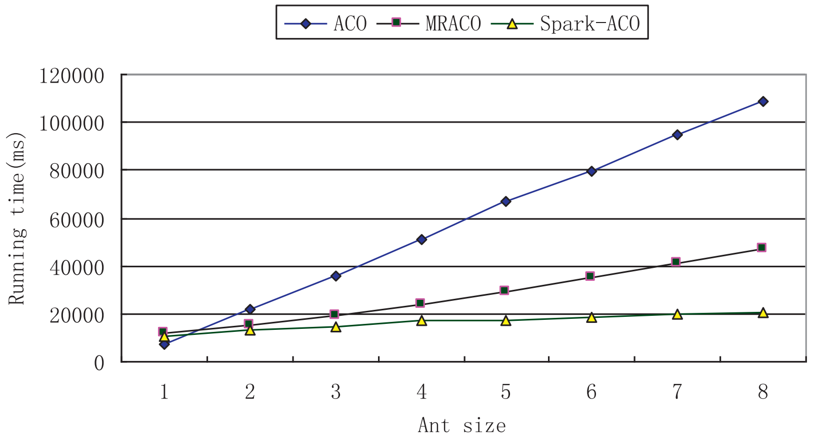

Fig. 4. The running time figure of ACO, MRACO and Spark-ACO.

Table 2
The operating results of ACO, MRACO and Spark-ACO.

TSP instances	Spark-ACO	Spark-ACO + NN	Spark-ACO + NN + GA	TSPLIB
berlin52	7592	7542	7542	7542
kroA100	21,939	21,292	21,282	21,282
ch130	6210	6132	6110	6110
kroB150	27,650	26,563	26,130	26,130
d198	17,033	16,032	15,780	15,780
rd400	18,004	16,010	15,493	15,281
rat783	10,832	9500	9498	8806

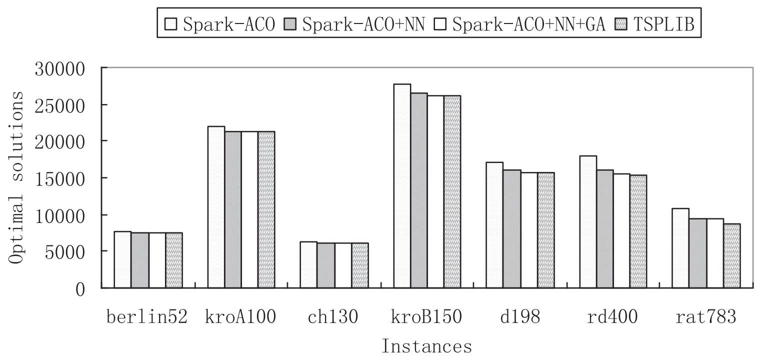

Fig. 5. The optimal result compare of Spark-ACO, Spark-ACO +NN, and Spark-ACO +NN +GA.

4.3. The accuracy of Spark-ACO, Spark-ACO+NN and Spark-ACO+NN+GA

In this paper, the method of the nearest neighbor strategy and hybriding genetic algorithm is proposed to optimize the solution of the basic ant colony algorithm. In order to verify the improvement of the cooperative ant colony optimization genetic algorithm compared to the basic ant colony algorithm for solving the quality of solution, an experiment is designed as follows: ACO, MRACO and Spark-ACO are run to compare the results.

In the experiment, the parameters of ant colony algorithm based on settings in past experience are as follows: the number of iterations $N_C = 300$. The number of ants $m = 200$, the pheromone evaporation coefficient $\rho = 0.5$, the heuristic information factor $\alpha = 1.0$, the expected heuristic factor $\beta = 2.0$, and the pheromone intensity $Q = 150$. In addition, the paper selects examples of berlin52, kroA100, ch130, kroB150, d198, rd400 and rat783 for experiments. For each TSP instance, the three algorithms are run 15 times, and the optimal solution and the TSPLIB provided by the three algorithms are compared. The experimental results are shown in Table 2:

Based on the solutions we draw a column chart, as shown in Fig. 5:

Fig. 5 shows: Spark-ACO+NN and Spark-ACO+NN +GA get the same optimal solution 7542 in berlin52. While in the calculation of five instances from berlin52 to d198, Spark-ACO+NN+GA has reached the optimal solution provided in TSPLIB [20]. The paper draws the optimal solution path graph of Spark-ACO+NN +GA, as shown in Figs. 6 and 7.

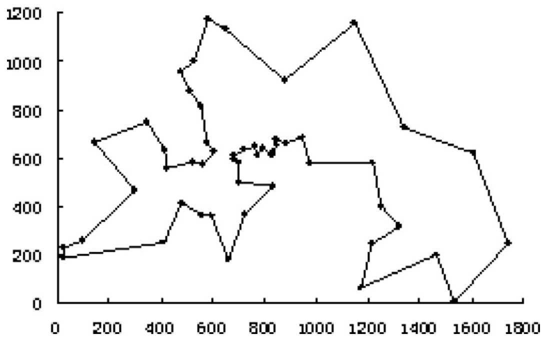

Fig. 6. The optimal solution of Spark-ACO+NN+GA for berlin52.

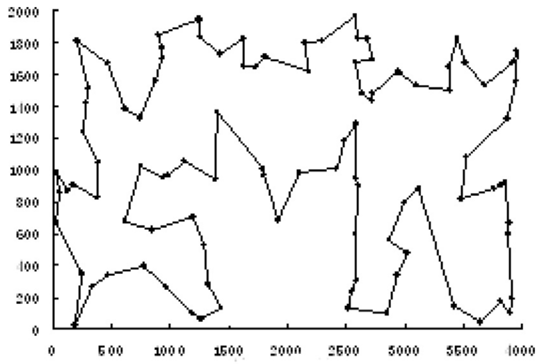

Fig. 7. The optimal solution of Spark-ACO+NN+GA for kroA100.

In this chapter, we introduce the design of the experiments. Firstly, it is proved that the increase of the number of cluster nodes does not reduce the running time of the ant colony algorithm. Then, by contrasting running time and path search results using different instances of the TSPLIB of Spark-ACO, Spark-ACO+NN, and Spark-ACO+NN+GA, the experiments show that, compared to the ACO algorithm, MRACO greatly shortens the operation time, and Spark-ACO algorithm obtains a better running speed than MRACO. In addition, compared to the path searching of Spark-ACO algorithm, Spark-ACO+NN+GA algorithm has greatly improved the quality of solutions.

5. Conclusion and discussion

According to the fact that a single machine cannot bear the calculation time of a large scaled traveling salesman problem, the basic ant colony algorithm is designed and implemented under MapReduce and Spark. Compared with the stand-alone basic ant colony system, the time of MapReduce basic ant colony algorithm has been reduced; compared with MRACO, the time of Spark platform basic ant colony has also been greatly reduced. In order to improve the accuracy of the solution, the NN strategy and GA are designed to be added in the Spark-ACO, and the accuracy of the solution is greatly improved.

The paper focuses on combining improved ant colony algorithm with parallel computing framework Spark, which improves the ant colony algorithm in cloud computing cluster at the running speed. However, the improved ant colony algorithm for the exact solution is not enough, because some of the examples haven't got the optimal solutions yet. Therefore, there are still many deficiencies in the accuracy of the ant colony algorithm. Next, we need to make efforts in the following aspects.

(1) Research on the other intelligent algorithms, and use their advantages to make up the shortcomings of the ant colony algorithm and improve the precision of the ant colony algorithm, so as to get a better path optimization results.
(2) Further study the operation principle of Spark parallel computing model and design the parallel ant colony algorithm more skillfully, so that it can run better and faster in the cloud computing cluster.
(3) Apply ant colony algorithm based on Spark platform to the practical field, such as biological information computing, job scheduling, network routing and other fields, in order to solve practical problems more efficiently.

Acknowledgement

This research was financially supported by Chinese Natural Science Foundations (61363016, 61063004), Key Project of Inner Mongolia Advanced Science Research (NJZZ14100), Inner Mongolia Colleges and Universities Education Department

Science Research (NJZC059), Natural Science Foundation of Inner Mongolia Autonomous Region of China (NO. 2015MS0605, NO. 2015MS0626 and NO. 2015MS0627) and Ministry of Education Scientific research foundation for Study abroad personel [2014] 1685.

References

[1] Schrijver A. On the history of combinatorial optimization (Till 1960)[*J*]. In: Handbooks in operations research & management science, 12; 2005. p. 1–68.
[2] Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies[*C*]. In: Proceedings of the 1st european conference on artificial LIFE. Elsevier Publishing; 1991. p. 134–42.
[3] Dorigo M, Gambardella M. A cooperative learning approach to the traveling salesman problem[*J*]. IEEE Trans Evol Comput 1997;1(1):53–66.
[4] Dorigo M. Ant colonies for the traveling salesman problem[*J*]. Biosystems 1997;43(2):73–81.
[5] Holland JH. Adaptation in natural artificial systems [M]. MIT Press; 1975. p. 1–17.
[6] Kennedy J, Eberhart R. Particle swarm optimization[*C*]. In: IEEE international conference on neural networks, 1995. Proceedings, vol. 4; 1995. p. 1942–8.
[7] Li X, Zheng A, Zhang X, Li C, Li Z. Rolling element bearing fault detection using support vector machine with improved ant colony optimization[*J*]. Measurement 2013;46(8):2726–34.
[8] Pan D., Yang C. An improved ant colony optimization for vehicle routing problem with time windows, logistics sci-tech, 2014.
[9] Dhawan C, Kumar Nassa V. Review on vehicle routing problem using ant colony optimization[*J*]. Int J Adv Res Comput Sci 2014.
[10] Eftekhari M, Zeinalkhani M. Extracting interpretable fuzzy models for nonlinear systems using gradient-based continuous ant colony optimization (GCACO)[*J*]. Fuzzy Inf Eng 2013;5(3):255–77.
[11] Chen YJ, Wong ML, Li H. Applying ant colony optimization to configuring stacking ensembles for data mining[*J*]. Expert Syst Appl 2014;41(6):2688–702.
[12] Arora V, Ravi V. Data mining using advanced ant colony optimization algorithm and application to bankruptcy prediction[*J*]. Int J Inf Syst Soc Change 2013;4(3):33–56.
[13] Johnson TV, Abbasi A, Kleris RS, Ehrlich SS, Barthwaite E. Segmentation and edge detection based on modified ant colony optimization for iris image processing. J Artif Intell Soft Comput Res 2013;3(2):133–41.
[14] Feng D, Zhang F, Zeng Q, Wang Q, Li C. Medical image processing and management based on ant colony optimization and support vector machine[*J*]. Int J Adv Comput Technol 2012;4(18):608–15.
[15] Thukaram P, Saritha SJ. Image edge detection using improved ant colony optimization algorithm[*J*]. Ijrcct 2013;2(11).
[16] Lee SG, Jung TU, Chung TC. An effective dynamic weighted rule for ant colony system optimization[*C*]. In: Evolutionary computation, 2001. Proceedings of the 2001 congress on, vol. 2; 2001. p. 1393–7.
[17] Gambardella LM, Dorigo M. Q: a reinforcement learning approach to the traveling salesman problem – machine learning proceedings 1995 - Ant[*J*]. Mach Learn Proc 2000;170(3):252–60.
[18] Gutjahr WJ. A graph-based ant system and its convergence[*J*]. Fut Gener Comput Syst 2000;16(8):873–88.
[19] Aharia M, Das T, Li H. Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters[*C*]. In: Usenix conference on hot topics in cloud ccomputing; 2012. p. 10-10.
[20] TSPLIB[*EB*/*OL*].[2016-03-31]. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp.

Dong Gaifang received her B.*Sc*. and M.*Sc*. degrees both in computer science from Inner Mongolia Normal University and GuiZhou University in 2002 and 2005. She is currently a Ph.D. student and an Associate Professor at the College of Computer and Information Engineering in the Inner Mongolia Agricultural University. Her research interests include bioinformatics computing, computational intelligence and parallel computing.

Fu Xueliang received his M.*Sc*. and Ph.D. degrees in software engineering from DALIAN University of Technology in 2005 and 2008 in China. Currently, he is a professor and the vice dean in the College of Computer and Information Engineering at Inner Mongolia Agricultural University in China. His research interests include Intelligent Computation, Data Mining and Big Data Processing.

Li Honghui received the B.*Sc*., M.*Sc*. and Ph.D. degrees all in computer science from Electronic Technology University, China, Inner Mongolia University, China and Concordia University, Canada in 1992, 2002, and 2012. She is currently a professor in Computer science & technology at Inner Mongolia Agricultural University, China. Her research interests include network optimization algorithms, routing algorithms and network planning.

Xie Pengfei graduated from the Zhengzhou University in 2013 and received his B.*Sc*. degree in software engineering. In July 2016, he received his M.*Sc*. degree in Computer Application Technology from the Inner Mongolia Agricultural University. Currently, he is a software engineer in Beijing. His research interests include biological information computing, parallel computing and big data processing.