Ant Colony Algorithm of Multi-objective Optimization for Dynamic Grid Scheduling

Xiaohong Kong, Junpeng Xu, Wei Zhang

School of Mechanical and Electrical Engineering,
Henan Institute of Science and Technology,
Xinxiang 453003, Henan, China

Abstract
A method for grid scheduling is proposed to optimize multiple objectives based on ant colony algorithm. Ant colony algorithm is a global optimization method and has the advantages of parallel search and positive feedback. But the algorithm is prone to stagnate or be trapped into a local optimum. This paper introduces the solution space information to improve the performance. The capacity of grid resource is exploited to produce the initial pheromone and the local pheromone and global pheromone are adjusted according to the workload in the late stage to maintain the load balance. The task cost is estimated as heuristic information when tasks are assigned to different grid resources to prevent the occurrence of premature and stagnation. The algorithm is realized in Gridsim environment and the simulation results prove that the proposed algorithm is superior to some heuristic algorithms.
Key words: GRID SCHEDULING, ANT COLONY OPTIMIZATION, PHEROMONE UPDATE, HEURISTIC INFORMATION

Introduction

With the development of computer and network technologies, Grid computing will supplement the traditional distributed computing for large-scale scientific computing projects. In general, the grid combines global resources span different organizations and share them together to provide more powerful computation and bigger memory [1]. Due to span-regional distribution of the available network resources, the grid is self-governing and had no control center. To make full use of the grid, the grid scheduling is to find a better resource for user jobs. Compared with traditional distributed system, the grid is a virtual organization and the grid resources are not dedicated, which can join in or withdraw from the grid from time to time [2]. Meanwhile, these grid resources meet the local load requirements and the grid required service at the same time, so scheduling algorithm should take into account the balance of grid tasks and the local tasks. Usually, there may be multiple jobs compete for resources and the user job submission is also dynamically changed. Besides that, communication cost among grid resources is much higher than LAN-based distributed computing environment and network delays cause considerable overhead if frequent communication between tasks. Therefore, it is assumed user jobs can be decomposed into coarse-grained meta tasks, and sub-tasks almost have no communication in order to reduce communication overhead ratio between

different computing devices. These features will challenge the scheduling algorithm when the grid executes a great quantity of computing tasks.

Considering the complexity of grid environment, the grid scheduling is a NP complete problem to coordinate the requirements of individual users and the whole optimal performance. A task scheduler is responsible for the overall allocation and resource management in traditional distributed computing system because system resources are fixed for proprietary applications. In grid environment, a centralized scheduler is not proper and several scheduling structures, decentralized, distributed, and hierarchical mode are considered. Each scheduler is responsible for a part of the scheduling or their superior's scheduler to improve overall system security and flexibility [3,4]. There are many criterions to evaluate the performance of scheduling algorithms. From the view of resources utilization, it is stressed to maximize throughput or seek the minimum workflow requirements. In terms of the user's considerations, grid should meet the various QoS (Quality of Service) requirements, such as the smallest cost in a certain period of time or reserve special devices for some work, etc[5,6].

Because of the complexity of the grid system, exhaustive search to find optimal scheduling is impossible. In order to achieve a higher performance, the existing literatures have discussed many heuristic algorithms to obtain sub-optimal solution within a certain period of time to meet the requirements[7,8,9,10,11,12,13]. But most algorithms can only meet certain scheduling constraints and have no good portability. These algorithms are classified into static and dynamic scheduling if algorithm is adjusted according to scheduling information [10,11]. Static algorithms estimate scheduling information in advance and complete all assignments at the beginning of scheduling regardless of dynamic changes in circumstances. Dynamic algorithms obtain the task executing time and resources information on-line, so the distribution of tasks is adapted to the grid autonomous features.

Dynamic scheduling is divided into immediate mode (on-line) and batch mode according to task scheduling internal [10,11]. The former schedules the task when it arrives and assigns it to resources immediately. In dynamic scheduling, ready tasks join the queue and wait to be assigned to resources based on event-driven, such as a machine released. It is clear that the latter considers the interaction among different tasks and environmental factors and can achieve a better scheduling result. In literature [10], it is investigated 11 kinds of scheduling algorithms and Min-min and Max-min show relatively good results. Min-min algorithm computes the minimum execution time for each non-scheduled task and selects the smallest time task to schedule, while the Max-min selects the largest time task from the minimum execution time series. The authors have confirmed that the algorithm is second only to genetic algorithm from the different nature of the tasks.

Ant colony optimization (ACO), inspired by nature, attracts everyone's attention in dealing with the complex issues[14]. In this paper, a dynamic algorithm is proposed to solve the scheduling grid based on ant colony algorithm. Dynamic scheduling strategy can be adjusted according to the problem information and is able to obtain better results than static scheduling. In order to make full use of parallel search capacity of ant colony optimization, proper solution space information must be connected with algorithm parameter. In particular, the dynamic pheromone update of ant colony based on the positive feedback is similar to the dynamic performance of grid system. In this paper, the capacity of grid resource is exploited to update the local pheromone and global pheromone to improve load balance. The task cost is estimated as heuristic information when tasks are assigned to different grid resources.

Ant Colony Optimization

Ant colony algorithm was first introduced by Dorigo, et al, inspired from the behavior of natural ants foraging food [14]. When ants look for food, each ant travels different paths and deposits pheromone on their paths to remind other ants. Finally, Individuals of ants share the path information through pheromone and always find the shortest path. By simulating the swarm intelligence of a colony of ants, the ACO algorithm has successfully solved many combinatorial optimization problem [5,9,15,16]. ACO is a population-based evolutionary approach where individuals of artificial ants search for problem space step by

step. In every step, it combines the previous experience of ants and heuristic information of the problem, and leaves some probability to explore the unknown space so that the algorithm obtains better solutions free from local minima [14].

Taking Traveling Salesman Problem (TSP) as an example, the initial idea of the ACO works as follows [14]: there are m ants and n cities and every ant must traverse all the cities only once while minimizing objective function. In this case, the shortest path or the least expenditure is expected. In every generation, an ant constructs a solution independently by choosing the next city according to state transition rule step by step until a tour is finished. When Ants visit cities, they deposit pheromone on their paths which exchange information with other ants and τ_{ij} depicts the amount of pheromone in the path that connects city i and j. The pheromone is also volatile with time in real circumstance, so the algorithm modifies the amount of pheromone on the visited paths by applying the local updating rule. It is assumed the distance between cities is known in advance and let d_{ij} represent the distance between city i and city j. The value $\eta_{ij}=1/d_{ij}$ is generally used as heuristic information to guide ants finding the best neighbor city. An ant located at city i selects the next city j according to the probability based on the density of pheromone τ_{ij} and the heuristic value η_{ij}. There is some probability to select other neighbor city, so the algorithm avoids local minima. Finally, the amount of pheromone is modified by applying the global updating rule to reward the best tour so far. In the following generations, ants incorporate the pheromone deposited in paths and a specified-problem heuristics to improve the solution repeatedly. The path with high pheromone is a very desirable solution [14].

ACO state transition rule

The rth ant positioned on city r chooses the next city s from neighbor city set $J_k(r)$ according to the state transition rule given in equations (1) and (2). Here, q is a random number uniformly distributed in [0, 1] and q_0 is a fixed parameter ($0<q_0<1$). When an ant chooses a city every time , it samples a random number $q\in[0,1]$. When $q\le q_0$, the next city is the city which has the biggest value in equation (1). Otherwise, the next city is a random variable S according to the probability distribution in equation (2). β is the weight factor between pheromone and heuristic information.

$$s=\begin{cases}arg\ \max\limits_{u\in J_k(r)}\{[\tau(r,u)][\eta(r,u)]^{\beta}\} & if\ q\le q_0\\ S & otherwise\end{cases} \quad (1)$$

$$p_k(r,s)=\begin{cases}\dfrac{[\tau(r,s)][\eta(r,s)]^{\beta}}{\sum\limits_{u\in J_k(r)}[\tau(r,s)][\eta(r,s)]^{\beta}} & if\ s\in J_k(r)\\ 0 & otherwise\end{cases} \quad (2)$$

The rule resulted from equations (1) and (2) is called pseudo-random-proportional rule. Equation (1) describes the exploitation of experience and the rule is prone to transitions towards cities with short distance and high pheromone density. Equation (2) challenges the exploration of untouched field and assures that the algorithm get rid of premature. The parameter q_0 reflects the relative proportion of exploitation versus exploration.

ACO local updating rule

When ants visit different cities, the pheromone level of paths is updated by applying the local rule prescribed in equation (3).

$$\tau(r,s)\leftarrow(1-\alpha)\tau(r,s)+\alpha\Delta\tau(r,s) \quad (3)$$

Where $0<\alpha<1$ is a local pheromone decay parameter. $\Delta\tau(r,s)$ is the pheromone increment and has three options [14].

ACO global updating rule

Different from real ants, artificial ants of the algorithm perform global pheromone updating after all ants have completed their tours. The global updating always strengths the best tour found up to the current iteration. This choice directs ants searching in neighborhood of the best tour. The global pheromone level is updated according to equation (4).

$$\tau(r,s)\leftarrow(1-\gamma)\tau(r,s)+\gamma\Delta\tau(r,s) \quad (4)$$

Where,

$$\Delta\tau(r,s)=\begin{cases}Q/L & if\ (r,s)\in global\text{-}best\text{-}tour\\ 0 & otherwise\end{cases} \quad (5)$$

$0<\gamma<1$ is the global pheromone decay parameter and L is the length of the best tour from the beginning of the trial. Equation (5) dictates reinforcement amount $\Delta\tau(r,s)$ in

the best tour. $\Delta\tau(r,s)$ also used the length of the best tour of the current iteration, called iteration-best, as opposed to the above called global-best.

ACO Algorithm for Grid Scheduling

In the proposed algorithm, each ant takes as a scheduler and constructs a scheduling scheme independently. Each scheduler collects resource information and selects proper processors for tasks while minimizing objective function. In order to utilize ant colony algorithm to solve complex problem, it is very important how to translate the solution information into pheromone and heuristics in a specific context. As a grid scheduler, the aim is to select powerful resource for tasks. That is to say, a resource with big capacity, high computing speed, high communicating speed and low expenditure is popular. Several strategies are used to improve the algorithm performance as follows.

Pheromone based on resource capacity

In many ant colony systems, the initial pheromone is set the same amount of in the path, which maybe mislead ant colony searching all the paths and prolong the algorithm time. Due to the heterogeneity of grid resources, it is clear that the resources with more processing capability of are more likely to be selected. So capacity of resources are selected as initial pheromone [14], different from blindness of the other algorithm. That is described in equation (6).

$$\begin{aligned}\phi_0(i) &= \sum_{j=1}^{k} p_{ij} + c_i (i = 1, 2 \cdots m) \\ \tau_0(s,i) &= \phi_0(i)\end{aligned} \quad (6)$$

Here, the resource i has k processors. p_{ij} is denoted the j th processor of the resource i and c_i communication capacity.

Heuristic based on solution information

When users submit tasks to grid, it is required to select some good resources to meet objective function. For example, scheduling length is a criterion in most of grid tasks. The Min-min algorithm has good results to use the minimum completion time, so the processing time of a task in a resource is selected as heuristic information, denoted as $\eta(s,i) = 1/ECT(s,i)$. $ECT(s,i)$ means the completion time of task s in the resources i, so the resource with the earliest completing time has much probability to be chosen.

Pheromone updating based on load balance

During the scheduling process, the good resources always have large probability to be selected, which lead to much more pheromone and heavy load on the other side. In order to avoid load imbalance, a self-adaptive strategy is used for pheromone updating. Considering the load tasks, the resources pheromone formula is amended as equation (7). After each task is allocated to resource, the resource pheromone is appropriately reduced based on tasks computing cost.

$$\begin{aligned}\phi(i) &= \phi(i) - \rho\Delta\phi(i) \\ \tau(s,i) &= (1-\rho)\times\tau(s,i) + \rho\phi(i)\end{aligned} \quad (7)$$

$\Delta\phi(i)$ is proportional to the job length of assigned tasks in the resource and it is ensured that good resource have not much heavy load. After the ant colony completes one iteration, the global pheromone is updated according to Equation (8).

$$\begin{aligned}\phi(i) &= (1-\gamma)\phi_0(i) + \gamma \sum_{job(i)} \phi(i) \\ \tau(s,i) &= (1-\alpha)\times\tau(s,i) + \alpha\phi(i)\end{aligned} \quad (8)$$

$\sum_{job(i)} \phi(i)$ is the sum of all the job length assigned to resource in this iteration. Here, $Makespan_{best}$ is the best scheduling and $job(i) \in Makespan_{best}$. When tasks are assigned to the selected resource, the technique of first come first serve (FCFS) is employed.

Simulation Results

In grid system, the users submit their applications (tasks) to resource brokers and the broker add it into the queue (scheduling list) waiting to be scheduled. Then, schedulers produce scheduling plan according to scheduling algorithm. The proposed algorithm is simulated in GridSim [17]. GridSim is a software package to provide various network components to simulate real Grid circumstance.

In order to approximate grid environment, user tasks are submit randomly and resources capacity are random, consistent with Poisson distribution and uniform distribution. These parameters are listed as follows: task computing amount is generated according with uniform distribution of the region [1000,5000] and [1000,5000]; the

submit time of tasks is classified into two categories, Poisson distribution of the mean value [100, 500, 1000] and uniform distribution of the interval range [10,1000] respectively. Processing speeds of resources satisfy [10,50] uniform distribution and each resource includes processors of [2,5] uniform distribution. Computing cost is measured in millions of floating point operations (MFLOPs). The execution rate of each available processor is measured in MFLOPs per second, denoted as MFLOP/s [18]. To verify the performance, these dataset are used to test different scheduling strategies. Each experiment was repeated and an average makespan was calculated. Scheduling results are discussed below.

Dynamic batch mode and on-line mode strategy

Two dynamic scheduling modes are used to select resource for user tasks. In on-line mode, task is assigned to resource when it is submit. In batch mode, a set of tasks in queue are assigned at the same time. The batch mode considers the impact of the assigned task on the subsequent tasks and requirements for resources, so it has better results than on-line mode. The results of different tasks are listed in Table 1.

Table 1. Makespan under Different Dynamic Scheduling Mode with Poisson Arriving Time

Task length interval	Poisson mean time	User tasks	Batch mode	On-line mode
[4000, 5000]	100	500	19292	19294
[4000, 5000]	500	500	19048	19639
[4000, 5000]	1000	500	19520	19567

Different batch algorithms

Several typical algorithms were utilized to measure the proposed algorithm, including the Min-min, Max-min[10,11]. Min-min method uses greedy strategy, first calculating the minimum completion time of a task in all resources, and then choosing the smallest of the "task - resources". Similar to the min-min algorithms, the max-min selects the resource with the biggest time in all the minimum completion time. First, the makespan is tested under different task sizes and different resource scales.

Table 2 shows completion times of different task sizes with arriving time of passion distribution on 25 resources and the more tasks, the longer time to complete in the same resources. Table 3 depicts completion times of 500 tasks with arriving time of uniform distribution under different resources scales and the shorter time is obtained with the increase resources. From Table 2 and Table 3, it can be concluded that the results are varied with different tasks and resources, and ant colony algorithm has obvious advantages in both cases. Min-min and Max-min have close performance, but Max-min is slightly worse.

Table 2. Makespan under Different Task Sizes with Arriving Time of Poisson Distribution(25 resources)

Task length interval	Poisson mean time	User tasks	Ant colony	Max-min	Min-min
[1000,5000]	100	100	3848	3882	4046
[1000,5000]	100	300	11090	11265	11593
[1000,5000]	100	500	18721	18743	19216
[1000,5000]	500	100	3922	3947	4057
[1000,5000]	500	300	11315	11518	11653
[1000,5000]	500	500	18843	18872	19524
[1000,5000]	1000	100	4134	4225	4356
[1000,5000]	1000	300	11745	11772	11865
[1000,5000]	1000	500	19064	19260	19487

Table 3. Makespan under Different Resource Scales with Arriving Time of Uniform Distribution(500 tasks)

Task length interval	Uniform interval range	Resource scale	Ant colony	Max-min	Min-min

[1000,5000]	[10,1000]	10	18701	19140	19368
[1000,5000]	[10,1000]	15	18996	19074	19148
[1000,5000]	[10,1000]	20	18629	19730	19076
[1000,5000]	[10,1000]	25	18621	18880	19142
[4000,5000]	[10,1000]	10	19523	19641	19110
[4000,5000]	[10,1000]	15	19229	19466	19350
[4000,5000]	[10,1000]	20	19143	19276	19432
[4000,5000]	[10,1000]	25	19012	19036	30320

Experiments show that ACO algorithm can take advantage of known pheromone feedback and the heuristic information to explore the unknown space simultaneously to solve this kind of problem. ACO is a good parallel algorithm and it combines exploitation and exploration strategy to obtain better results. When the task cost has bigger disparities, ant colony algorithm has strong search capability and emphasizes the correspondence of the task to the optimum resources. After better information to be strengthened, local and global pheromone take a different upgrade strategy to ensure improving the algorithm performance further. Min-Min has better results in the static scheduling [11], but such a greedy selection method may extend the execution time when tasks have bigger discrepancy. Compared with Min-min algorithm, Max-min algorithm gives a priority to a larger workload task, which takes precedence over better resources.

Load balance

In general, different indicators may be used to represent load ratio, such as tasks number on current resources, CPU utilization, and resource execution time etc. In this case, execution time is considered as a standard to examine each resource load and algorithms are tested in an environment with 10 resources and 500 tasks. As shown in Table 4, 10 resources have different task lengths because of the different processing capacity. The Table 5 shows the sum processing time of each resource under three algorithms. It demonstrates that each resource is basically the same time in ACO algorithm while several resource have longer processing times in the other two methods. The two algorithms always select resource greedily and some tasks occupy resources and block subsequent tasks for special resources, which led to the longer completion time of the entire job.

Table 4. Tasks Length on Different Resources (MFLOPs)

	NO.1	NO.2	NO.3	NO.4	NO.5	NO.6	NO.7	NO.8	NO.9	NO.10	Sum
Ant colony	161476	131767	225348	70536	72468	105196	120677	263876	157322	227618	1536284
Max-min	248233	166484	198780	56327	75336	102840	107056	224355	231891	204106	1536284
Min-min	219466	187795	204039	165730	137949	136118	127108	123571	113988	120520	1536284

Table 5. Processing Time on Different Resources (s)

	NO.1	NO.2	NO.3	NO.4	NO.5	NO.6	NO.7	NO.8	NO.9	NO.10	Average
Ant colony	6006	5513	6293	7075	5197	5280	5505	6033	5642	5866	5741
Max-min	9228	6964	5552	5643	5395	5161	4886	5135	5309	5268	5854
Min-min	8166	7860	5699	16601	9876	6829	5798	2832	4094	3108	7086

In order to indicate the contribution of every resource to the whole task, load ratio is defined as the proportion of processing time to average time to measure the load balance. Load ration is shown in Figure1 and it is clear that the ACO algorithm has better load

balance than other algorithms. In the proposed algorithm, a flexible pheromone updating method is used to ensure the load distribution according to resource capacity.

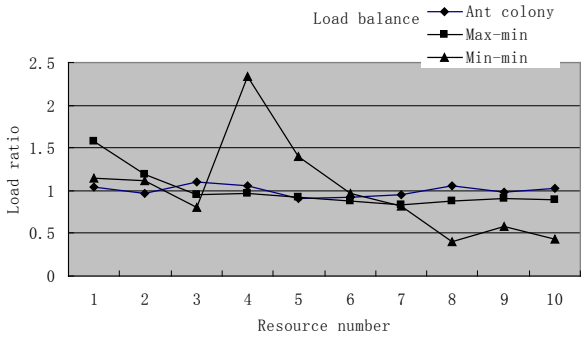

Figure 1. Load balance

Conclusions

Considering the dynamic characteristic of grid resources, this paper proposed a multi-objective dynamic algorithm for grid task scheduling based on ant colony optimization. The proposed algorithm had superior performance due to the self-adaptive pheromone update strategy and real-time adjustment of dynamic parameters. Simulation results demonstrated that the algorithm ensured the load balance and desirable scheduling length. Further, the technique can be extended to the data grid, distributed storage grid and other combinatory problem.

Acknowledgements

This work was supported by Scientific and Technological project of Henan Province(No.102102210190 and No.142102210112)

References

1. Foster I, Kesselman C. (2004) *The Grid: Blueprint for a New Computing Infrastructure Second Edition*, Morgan Kaufman Publication, USA.
2. Kokilavani T, George Amalarethinam D I. (2010)Applying Non-Traditional Optimization Techniques to Task Scheduling in Grid Computing. *International Journal of Research and Reviews in Computer Science*, 1(4), p.p. 34–38.
3. FIBICH Pavel, MATYSKA Luděk, RUDOVÁ, Hana. (2005)Model of Grid Scheduling Problem. Exploring Planning and Scheduling for Web Services, Grid and Autonomic Computing. *Technical Report, Menlo Park, California, USA: AAAI Press*, p.p. 17-24.
4. Jeyarani R, Ram R.V, Nagaveni N. (2010) Design and Implementation of an Efficient Two-Level Scheduler for Cloud Computing Environment. *Proc. 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)*, 585, p.p17-20.
5. Chen W.-N, Zhang J. (2009) An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. *IEEE Trans. Syst. Man Cybern. Part C Appl*, 39(1), p.p.29-43.
6. Nidhi J, Inderveer C. (2014) Energy Efficient and High Performance Scheduling Algorithm for Grid Computing. *International Journal of Cloud Computing and Services Science*, 3(3), p.p.179-188
7. Kokilavani T, George Amalarethinam D.I. (2011) Load Balanced min–min Algorithm for Static Meta-Task Scheduling in Grid Computing. *International Journal of Computer Applications*, 20(2), p.p.43-49.
8. Zhang L, Chen Y, Sun R, Jin S, Yang B. (2008)A Task Scheduling Algorithm Based on PSO for Grid Computing. *International Journal of Computational Intelligence Research*, 4(1), p.p.37–43.
9. Kousalya K, Balasubramanie P. (2008) An enhanced ant algorithm for grid scheduling problem. *International Journal of Computer Science and Network Security*, 8(4), p.p.262–271.
10. Braun T.D, Siegel H.J, Beck N. (2001) A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. *Journal of Parallel and Distributed Computation*, 61(6).p.p.810 - 837.
11. Siegel H. J, Ali S. (2000) Techniques for Mapping Tasks to Machines in Heterogeneous Computing Systems. *Journal of Systems Architecture, Special Issue on Heterogeneous Distributed and Parallel Architectures: Hardware, Software and Design Tools, 46*(8), p.p.627-639.
12. Zheng W, Sakellariou R. (2012) Budget-deadline constrained workflow planning for admission control in market-oriented environments. *Economics of Grids, Clouds, Systems, and Services*, p.p.105–119.
13. Zahra P, Mohammad S, Bahman Javadi. (2012) Independent Task Scheduling in Grid Computing Based on Queen-Bee Algorithm. *International Journal of Artificial Intelligence*, 1(4), p.p.171-181.
14. Dorigo M, Caro G. D, Gambardella L. M. (1999) Ant Algorithms for Discrete Optimization. *Artificial Life*, 5(3), p.p.137-172
15. Xin L, Zilong G. (2011) A load-adaptive cloud resource scheduling model based on

ant colony algorithm. *2011 IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, 296(300), p.p.15-17.
16. Xu Z. H, Hou X. D, Sun J. Z. (2003)Ant Algorithm-based Task Scheduling in Grid Computing. *Canadian Conf. on Electrical and Computer Engineering*, 2, p.p. 1107-1110.
17. Buyya R. (2002) *Economic-based Distributed Resource Management and Scheduling for Grid Computing.* Melbourne, Australia: Monash University.
18. Page A. J, Naughton T. J. (2005) Dynamic Task Scheduling using Genetic Algorithms for Heterogeneous Distributed Computing. *Proc. of 19th IEEE International Parallel and Distributed Processing Symposium*, p.p.189–197.