Parallel Swarm Intelligence Algorithm for Grid Scheduling Based on Multiple Objectives Optimization

Xiaohong Kong, Hao Guo, Wenqiang Yang, Baochun Wang

***Abstract*—**Grid aggregates enormous resources spanning different geographic organizations and shares them together to provide powerful computation ability and tremendous memory with low cost. Grid is a potential trend for large-scale computation and complex engineering problems. But grid resources allocation and tasks scheduling are huge challenges due to its dynamic characteristic and heterogeneous architecture. This paper proposes a parallel ant colony algorithm and ant colony is divided into subgroups. Each subgroup searches solution space respectively to enhance the global ability of algorithm. Further, subgroups interchange the information periodically to speed the algorithm convergence and avoid local optimum based on message sharing. Dynamical heuristics information and adaptive pheromone updating are implemented to optimize multi-objectives for grid scheduling. The algorithm is simulated in master-slave system with distributed memory architecture and Gridsim environment. The experiment results are compared with different algorithms and prove that the proposed algorithm can improve the solution quality as well as load balance.

***Keywords*—**Grid scheduling, parallel algorithm, ant colony, heterogeneous architecture, subgroup.

I. INTRODUCTION

With the development of technology and engineering, more and more complex problems require excess hardware working together to complete large-scale computation. On the other hand, various types of network resources aren’t fully utilized. Based on network technology, grid provides a new computing paradigm and supplements the traditional distributed computing and parallel computing [1]. Sharing global idle resources, grid computation copes with complicated problems with high performance and low cost. Resources management and tasks allocation are the key issues.

In general, grid is loose, autonomous framework and is consisted of heterogeneous resources spanning different geographical regions and organizations. To make the best of grid, tasks must be allocated to suitable resources combining cost, deadline and other criterions to satisfy user requirements. But grid resources management and tasks allocation are great challenges due to its dynamic characteristic. Furthermore, the heterogeneous architectures put forward higher requirements on resources distribution. Unlike traditional distributed system, grid resources are not dedicated and undertake workload of both grid and owners, so grid scheduling is much complex problem [1]. Many literatures have proposed effective scheduling strategies and different standards are discussed to evaluate the algorithms. Most of them are applicable to some extent and no one is universal for all occasions.

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

Xiaohong Kong is with Henan Institute of Science and Technology, Xinxiang, Henan 453003, China
Hao Guo is with Henan Institute of Science and Technology, Xinxiang, Henan 453003, China a.
Wenqiang Yang is with Henan Institute of Science and Technology, Xinxiang, Henan 453003, China a.
Baochun Wang Xinxiang is with Hongsheng vibration motor Co. Ltd., Xinxiang, Henan 453003, China

II. RELATED WORKS

Grid is virtual organizations and available resources join and withdraw from time to time. Users randomly submit jobs to grid and brokers are in charge of finding resource, receiving users’ jobs and scheduling the jobs. Therefore, single scheduler is improper, and decentralized, distributed and hierarchical scheduling architectures are discussed [1][2][3]. Authors compared existing resource allocation mechanisms and gave detail taxonomy of resource allocation to help how to choose scheduling scheme[1]. A two-level strategy was proposed to optimize the computation time and energy consumption [2]. In hierarchical structure, resources are managed by central level and low level brokers to improve the scheduling flexibility [1][3].

Due to the enormous network resources and randomly submitted user tasks, exhaustive search leads to the explosion of solution space and longer time. At present, researchers have proposed many heuristic algorithms to obtain the approximate optimal solution of the problem within reasonable time[4]-[9]. These scheduling algorithms are classified into two types, static scheduling and dynamic scheduling. In static scheduling, all resources and tasks information are known in advance and tasks are allocated only once at the beginning of running [4][5][6]. When scheduling information is not indefinite and dynamically changeable, dynamic strategy must be adopted. In dynamic scheduling, tasks are allocated based on estimated information and can be redistributed when the changeable network resources are not suitable [1][7]. The latter considers both of tasks features and resources characteristics, and can achieve better performance. Grid tasks are submitted to network randomly and scheduling information are uncertain, so grid scheduling is dynamical.

Usually, two scheduling modes are discussed: immediate mode and batch mode[3][6]. In immediate mode, scheduler distributes tasks to resources at the moment tasks arriving. In batch mode, resource brokers collect tasks into the queue and schedulers allocate a cluster of tasks periodically. Apparently, batch mode considers the interaction between tasks in a cluster and available resources, so it is better than static one. Here, the key is how to choose a batch of tasks. Bigger tasks cluster causes extra computation cost and smaller tasks cluster leads to low efficiency. Generally, batch scheduling is driven by an event or scheduling interval.

In general, these algorithms have some limitations and different algorithms are suitable to given occasions [1][7]. Different criterions are discussed and optimization parameters are compared to evaluate algorithms performance [6]-[9]. The evolution of grid scheduling is introduced and the performance of typical algorithm is compared in detail [7][8]. The swarm intelligence and evolutionary algorithms are illustrated efficiently for grid scheduling [8]. But most algorithms pay attention on unique goal. A good algorithm should take into account the utilization of resources and the quality requirement of the user. From the perspective of resource utilization, it is desirable that the network throughput is high and the workload is balanced [4][5]. From the user point of view, it is emphasized on quality of service (QoS) and tasks are completed in short time and low cost [9].

In this paper, ant colony algorithm is used to optimize the user-related makespan and enhance the resource-related load balance. When tasks scale is larger, the algorithm time becomes a bottleneck. A Parallel ant colony version is implemented. To improve the scalability, the algorithm is tested in random network environment irrespective of computing, storage, communication abilities. Ant colony algorithm has been widely used to solve combination optimization problems and is proved to have better performance [10]-[13].

III. ANT COLONY OPTIMIZATION

Ant colony optimization (ACO), inspired by the behavior of ant colony foraging food, is demonstrated superior performance in coping with combinatory optimization problems [10]. When ant colony looks for food, the individual tries different paths and secretes pheromone on the path. The pheromone is used to exchange information among ant colony to direct other partners. With the time going, pheromone gradually disappears and pheromone concentration represents the path distance. The shorter the path is, the more pheromones are deposited and the more ants are attracted. In this process, a small number of ants encounter an obstacle and search for another path, which can be understood as an innovation in accordance with a certain probability, and this innovation may lead to better path. Ant colony makes full use of the biological information transmission based on the positive feedback and always finds the shortest path from the nest to the food.

ACO algorithm employs collaborative cooperation of swarm intelligence and is widely used to solve complex problems due to the similarity of foraging behavior and searching solution space of problems. As an example, ACO is first used in Traveling Salesman Problem (TSP) and works as follows. Given a series of cities and the distance between each pair of cities, the goal is to find the shortest route by visiting every city only once. ACO is evolutionary algorithm and constructs solution by traversing problem space step by step. In every generation, ants complete a tour according to state transition rule, choosing the adjacent city. Finally, the best city's trajectory is the optimization solution of problem.

When ants visit different cities, pheromone is accumulated in the path and the algorithm also simulates the volatile of pheromone as in real circumstance. Here, τ_{ij} represents the amount of pheromone in the path between city i and j, and it is modified by applying the local updating rule. When ant colony completes a tour, the pheromone in the best path is strengthened and is modified by applying the global updating rule. The process is repeated until a desirable solution is found.

A. State Transition Rule

When ants visit different cities, it is equivalent to searching for a discrete solution space and moving from one state to another. The pheromones record the current state and the historical state of searching solution, denoting as the previous "experience". The historical "experience" can be used by subsequent ants to make the choice of the path. As a habit, an ant chooses the next neighbor city with more pheromone. In specific issues, ants are not completely blind and also inspired by the spatial characteristics of the problem, such as shorter distance. This strategy is greedy, and is easy to lead to premature convergence and fall into the local optimum. So Pseudo-random ratio is introduced to improve the algorithm quality and permits ants to explore some unknown space by a certain probability.

The k th ant positioned on city i has an optional neighbor city set $J_k(i)$ and the next city j is chosen according to equations (1) and (2), named state transition rule. The state transition rule provides a balancing mechanism between exploitation and exploration.

$$j = \begin{cases} arg \max\limits_{j \in J_k(i)} \{ \tau_{ij}{}^{\alpha} \eta_{ij}{}^{\beta} \} & if \ q \le q_0 \\ p_k(i,j) & otherwise \end{cases} \tag{1}$$

$$p_k(i,j) = \frac{\tau_{ij}{}^{\alpha} \eta_{ij}{}^{\beta}}{\sum\limits_{j \in J_k(i)} \tau_{ij}{}^{\alpha} \eta_{ij}{}^{\beta}} \tag{2}$$

First, a random number q is generated in the range [0, 1] and compared with a fixed parameter q_0 .When $q \le q_0$, the next goal is the city with the maximum value meeting equation (1). Otherwise, the next city is a random variable in according with probability $p_k(i,j)$ in equation (2). The value η_{ij} is generally acquired from the problem as heuristic information to guide ants searching direction. The weight factor α and β measure the importance of pheromone and heuristic information.

B. *Local Updating Rule*

When ants visit different cities, pheromone level varies with the volatility of the original pheromone and the addition of new joining individuals. Local updating rule is depicted in equation (3), updating the pheromone in every iteration.

$$(1-\rho)\tau_{ij}+\Delta\tau_{ij}\rightarrow\tau_{ij} \tag{3}$$

Parameter ρ ($0<\rho<1$) is a pheromone decay factor. $\Delta\tau_{ij}$ is the pheromone increment of the path ants passing.

C. *Global Updating Rule*

When the ant colony finishes a tour, global pheromone updating is applied to the best path according to equations (4) and (5). The global updating always emphasizes elite ants and speeds up the algorithm convergence.

$$(1-\gamma)\tau_{ij}+\Delta\tau_{ij}\rightarrow\tau_{ij} \tag{4}$$

$$\Delta\tau_{ij}=\begin{cases}Q/L & if(i,j)\in global\text{-}best\text{-}tour\\ 0 & otherwise\end{cases} \tag{5}$$

γ ($0<\gamma<1$) is the global pheromone decay parameter and L is the length of the global best tour. Generally, two kinds of increment $\Delta\tau_{ij}$ are discussed, called the global best path and the current best path, which use the length of the best tour so far and the best path in the current iteration respectively.

D. *Parallel Ant Colony Algorithm*

Due to the complexity, the ACO algorithm has longer searching time than other algorithms. At the same time, it is easy to stagnate and all the ants find the same solution. Based on fast communication and huge network resources, parallelized algorithms become a trend for large-scale optimization problem. Parallel implementation can shorten the algorithm time and improve the robustness than their sequential counterparts [11][12] [13].

There are a few parallel versions of ant colony algorithm and most implementations differ in parallel scale and the way ant colonies exchange share information [11]. The authors also introduce a new taxonomy of software-based parallel ACO. Three parallel paradigms are summarized and parallel evolutionary algorithms are proposed to solve heterogeneous grid systems [12]. Six variants of the algorithm were studied to optimize makespan and flowtime and the results demonstrated that the algorithm could efficiently tackle with large scheduling problems. Ant colony structure was modified to fit an adaptive parallel ant colony optimization for massively parallel processors [13]. The algorithm was run in different processors and each processor updates their pheromone adaptively. The results for TSP showed that the PACO algorithm has a better performance in convergence speed, speed up and efficiency.

In a word, the parallel swarm intelligence algorithms can improve the quality of the solution, shorten the algorithm time, accelerate the convergence of algorithms, etc. However, the efficiency of the parallel algorithm is also related to the structure of the architecture and the specific problems, so getting the advantages of the parallel algorithm need to consider many factors.

IV. PROPOSED ALGORITHM FOR GRID SCHEDULING

A. *Multiple Running of Ant Colony Optimization*

The same algorithm is simultaneously executed in multiple machines and ant colonies in different machines don’t exchange information. This is the simplest implementation of parallel algorithm and different parameters are selected for each algorithm process respectively. When the termination condition is met, the best solution on all machines is the final result.

B. *Multiple Subgroup Colonies with Shared Information*

In this mode, ant colony is split into small groups and each subgroup independently searches for the solution space and periodically exchanges information about their solutions. Here, the current best solution and the global best solution in every subgroup colony are selected as the exchange information to increase the diversity of solution. Inside the subgroup colony, the dynamical batch mode is adopted and scheduling of a cluster of ready tasks is driven by events such as the earliest idle machine.

The algorithm is implemented in master-slave system with distributed memory architecture and simulated in MPI (Message Passing Interface) environment [14]. The information exchange is done centrally by a master processor. The master processor collects global information and broadcasts the best solution to slave processors. Each subgroup colony works independently on a slave processor and updates the local pheromone to rebuild the solution. Coarse-grain parallelization is more potential than fine-grain parallelization because of the high communication costs associated with pheromone management. So bigger subgroup is considered and each subgroup stores the pheromone information in local memory.

C. *Problem-specific Initial Pheromone*

In general, the initial pheromone is constant and given at the beginning of the algorithm. Ants search the solution in all directions with the same probability and it will lengthen the time. Resources with large storage capacity and fast computation power have greater potential to perform tasks, and the greater chance to be selected. So the capacity of grid resource is exploited to generate the initial pheromone to reduce the search blindness. The process is depicted in formula (6). Here, p_j denotes the process capacity of the j th resource and c_j is communication capacity.

$$\tau_{ij0}=k(p_j+c_j) \tag{6}$$

D. *Dynamical Heuristics Information*

Effective heuristic information is critical to the performance of the ant colony algorithm and guides the ant colony to find the solution quickly. In this paper, the potential computation cost of tasks on target resources is evaluated as heuristic information.

In the process of scheduling, ants dynamically select the match of tasks and grid resources. The heuristic information is extracted from the problem and guides ants to unexploited solution space, so it will prevent the premature and stagnation at some extent.

E. Adaptive Pheromone Updating

In grid scheduling, ant colony foraging is similar to find desirable resources for user tasks and better resources are distributed heavier tasks and the pheromone is strengthened. Due to positive feedback principle, too much tasks are concentrated in few resources, which leads to workload imbalance. Considering the workload in the late phase, the local pheromone and global pheromone updating are self-adaptively adjusted according to formula (7). ω is workload factor and w is the resource workload.

$$(1-\rho)\tau_{ij}+\Delta\tau_{ij}+(1-\omega)\sum w_j \to \tau_{ij} \tag{7}$$

F. Optimize the Multiple Objectives

Taking into account network utilization and user requirements, the scheduling aims at makespan and workload balance. The simulation results show that the parallel ant colony algorithm is superior to some existing algorithms.

V. EXPERIMENTS

The proposed algorithm is implemented in simulation platform: GridSim [15]. GridSim is a software package that provides a variety of network components to simulate real grid environment. The users submit its application (tasks) to the resource brokers, which adds tasks to the waiting queue, and schedulers generate distribution plan based on the proposed algorithm.

A. Experiments Benchmark

In the experiment, user tasks arrive randomly and grid resources are heterogeneous in both of computation power and storage capacity. These characteristics are determined by the following parameters in the grid. Tasks computation cost and resources speed are measured in millions of floating point operations (MFLOPs) and MFLOPs per second [16]. This project utilizes the donating resources such as computing power, storage devices of volunteers to provide potential millions of internet-attached processing platforms.

Two types of tasks with the uniform distribution and poisson distribution are studied. Computation costs are a uniform distribution of the range [1000,5000] and [4000,5000]. The tasks arriving times are in accordance with poisson distribution of the mean value [100, 500, 1000] and uniform distribution of the interval range [10,1000] respectively. Grid resources are consisted of processors of uniform distribution between [2,5] and processing speeds of processors are consistent with the uniform distribution of [10,50]. These data sets are used to test different scheduling strategies and each experiment is repeated.

B. Two Versions of Parallel ACO

Sequential ACO (SACO) and two versions of parallel ant colony algorithm are compared in the above mentioned benchmark. Multiple running of parallel ACO is denoted as PACO-I and the other is PACO-II. The second algorithm considers communication frequency among subgroup colonies, measured by cycle times T_{cycle}. Three cases are discussed (T_{cycle} =5, 10, 15). The experiment also investigates the number of subgroup colonies and the exchange information among colonies. The task costs are according with [1000,5000] uniform distribution and the task arriving times are also uniform distribution. Every algorithm is run 20 times and the best solution over 50 resources is listed in Table.1. The results of PACO-II are always better than others because subgroup colonies share solution in the searching process. In the follow experiments, the second parallel algorithm is used.

Table.1. The best solution of the different ant colony algorithms

Number of users tasks	SACO	PACO-I	PACO-II(Number of subgroup colonies)											
			T_{cycle} =5				T_{cycle} =10				T_{cycle} =15			
			2	4	8	10	2	4	8	10	2	4	8	10
100	4304	4107	3900	4022	4031	4044	3978	3945	3989	4015	4012	3982	3909	3893
200	7998	7956	7698	7776	7812	7990	7793	7715	7793	7768	7932	7865	7797	7600
300	12659	12345	11077	12021	12096	12176	12038	12003	12046	12067	12143	12095	11087	11056
400	16417	15399	15147	15189	15231	15378	15207	15199	15235	15287	15391	15297	15197	15165
500	19130	19080	18621	18687	18694	18743	18698	18677	18705	18709	18731	18672	18597	18563

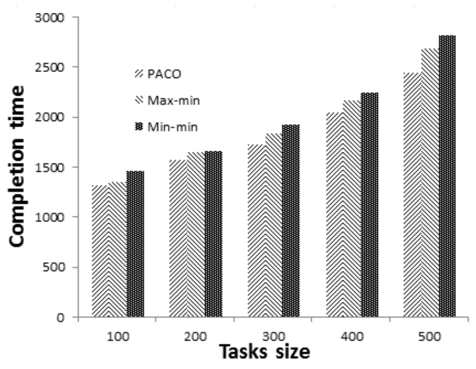
Two kinds of share information are studied: the current best solution and the global best solution so far. The dataset includes tasks of [4000,5000] uniform distribution cost with arriving time of poisson mean time 100. Table.2 gives the completion time under two circumstances. The global best information is slightly better than current best information because the global information can guide subgroup colonies to reduce blindness and enhance cooperation.

C. Compared with Other Algorithms

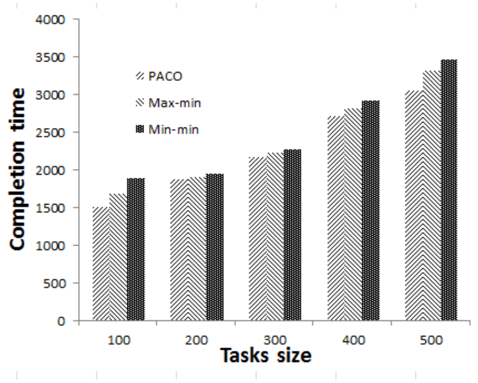
The proposed algorithm is compared with two kinds of typical algorithms, including Min-min, Max-min Algorithms [4]-[7]. When the scheduler selects resources for ready tasks, all the minimum computation costs of tasks on available resources are evaluated in advance. Min-min algorithm employs greedy strategy and the resource with the minimum value is chosen. Max-min algorithm is just the opposite and tasks are assigned to resources with the maximum value. The three algorithms are implemented in the datasets with different sizes of tasks and resources. The first case includes different tasks scales with arriving time of uniform distribution on 50 grid resources. Under the same resources, the more tasks are, the longer the completion time is. The second scenario considers 500 tasks with arrival times of poisson distribution and computation costs of uniform distribution between [1000, 5000]. Shorter time is obtained when resources increase.

Table.2. Results of different share information on 50 resources

Number of resources	SACO	PACO-II (Number of subgroup colonies)							
		Current best solution				Global best solution			
		2	4	8	10	2	4	8	10
100	4352	4136	4144	4102	4173	4078	3959	3901	3927
200	8167	7953	7967	7933	7994	7907	7826	7876	7895
300	12272	11705	11801	11743	11796	11658	10827	10939	11687
400	15721	15303	15317	15223	15405	15318	15178	15296	15379
500	19953	19260	19304	19189	19369	19245	19128	19290	19368



(1) Task cost interval [1000,5000]



(2) Task cost interval [4000,5000]

Fig.1. Completion time of different task sizes with arriving time of uniform distribution

The completion times of different task sizes are described in Fig.1 and different resource sizes in Fig.2. As can be seen from Fig.1 and Fig.2, the completion time varies with different tasks and resources and parallel ant colony algorithm performs well in both cases. Min-min algorithm and Max-min algorithm have similar results, but Max-min is slightly better.

D. Load Balance

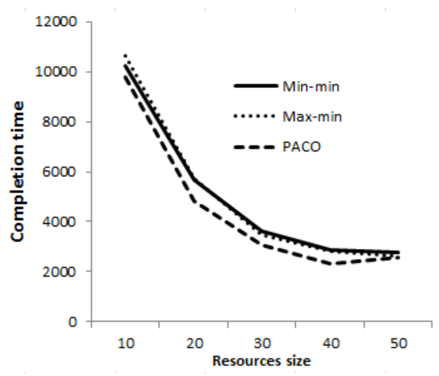
ACO algorithm uses pheromone to enhance the search direction, but also uses heuristic information to explore the unknown solution space and avoid the algorithm premature. Combining with the parallel advantage, it can quickly get better results. When costs of tasks are quite different, ant colony algorithm emphasizes the matchup between the task and the optimal resource. So the algorithm has stronger ability in solving problem. After a certain search period, local pheromones and global pheromones take different upgrade strategies to ensure further improving algorithm performance. Min-Min uses a greedy selection strategy and can extend execution time when costs of tasks are obvious differences. Unlike the Min-min algorithm, the Max-min algorithm prioritizes larger workload and has slightly good performance.

Due to the positive feedback of the ant colony algorithm, the better resource is selected more probability, resulting in load imbalance of different resources. In this paper, the algorithm takes the existing load on the resource as part of the pheromone update. The more load on the resource is, the less pheromone increases, or even negative growth. When tasks are assigned to grid resource, the tasks cost is estimated to update the sum workload. Adapting to the changing grid systems, the dynamical adjustment improves load balance.

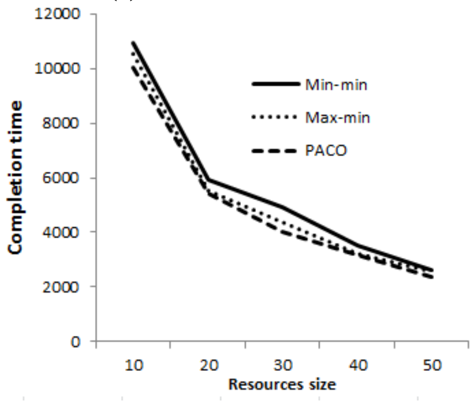
Simulation experiments are implemented on 10 resources and 500 tasks with costs of uniform distribution [4000,5000].

The load rate is defined as the ratio of the actual load of each resource to the average load of the whole task in all resources. The results of three algorithms are shown in Fig.3 and the proposed algorithm has better load balance than Min-min and Max-min. The success of the parallel ant colony algorithm lies in multi-agent collaborative mechanism. The algorithm obtains the real-time dynamic load of the resource and adapts the scheduling in the subsequent resource selection. It avoids the difficulties that the superior resources have too heavy load and inferior resources are not fully exploited, so it can solve load balance of the network resource successfully.

The other two algorithms are always greedy to choose resources for tasks, resulting too many tasks congestion on superior resources and inadequate use of inferior resources, so it triggering longer completion time and load imbalance.



(1) Poisson mean time 100



(2) Poisson mean time 500

Fig. 2. Completion time of 500 tasks on different resource scales

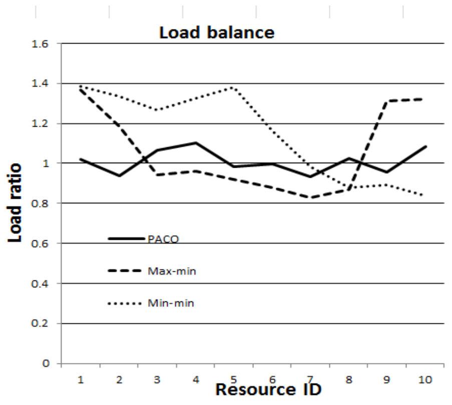


Fig.3. Load balance in different resources

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, a parallel ant colony algorithm is proposed for grid scheduling problem. The initial pheromone is extracted from the problem and guides the ant colony to quickly and effectively search the solution space. Based on the grid resource specificity, the ant colony pheromone is adaptively updated and heuristic information is dynamically adjusted. In contrast to other counterparts, parallel ant colony algorithm solves grid scheduling effectively and efficiently, and experiment results testify it a promising strategy for task completion time and workload balance.

In future, how the adaptive strategy and parallel structure affect the performance of the algorithm is also a key issue. The impact of subgroup scale and internal parallelization are also studied further. In this paper, few strategies to exchange information among multiple ant colonies are investigated and the frequency of information exchange and how to choose the share information aren't known: exchanging the local best solution only with the neighbor colonies not too often or exchanging the global best solution very often among all colonies.

REFERENCES

[1] M. B. Qureshi, M. M. Dehnavi, N. Min-Allah, M. S. Qureshi, H. Hussain, I. Rentifis, et al, "Survey on grid resource allocation mechanisms", *Journal of Grid Computing*, vol.12, no.2, pp. 399-441, 2014.

[2] B. Dorronsoro, S. Nesmachnow, J. Taheri, A. Zomaya, E.G. Talbi, P. Bouvry, "A hierarchical approach for energy-efficient scheduling of large workloads in multicore distributed systems", *Sustainable Computing Informatics & Systems*, vol. 4, no.47, pp.252-261, 2014.

[3] J. Kolodziej, S. U. Khan, "Multi-level hierarchic genetic-based scheduling of independent jobs in dynamic heterogeneous grid environment", Information Science, vol.214, no.23, pp. 1–19, 2012.

[4] T. K. Ghosh, R. Goswami, S. Bera, S. Barman, " Load balanced static grid scheduling using Max-Min heuristic", *Proc. of IEEE International Conference on Parallel Distributed and Grid Computing*, pp.419-423, 2012.

[5] T. Kokilavani, D. I. George Amalarethinam, "Load balanced min-min algorithm for static meta-task scheduling in grid computing", *International Journal of Computer Applications*, vol.20, no.2, pp. 42-48, 2011.

[6] M. Shahid, Z. Raza, "Performance evaluation of Static Level based Batch Scheduling Strategy (SLBBS) for computational grid", *Proc. of International Conference on Parallel, Distributed and Grid Computing*, pp.169-175, 2015.

[7] Y. Hao, G. Liu, "Evaluation of nine heuristic algorithms with data-intensive jobs and computing-intensive jobs in a dynamic environment", *IET Software*, vol. 9, no.1, 7-16, 2015.

[8] A .Yousif , S.M. Nor , A. H .Abdualla , M.B. Bashi, "Job scheduling algorithms on grid computing: State-of- the Art ", *International Journal of Grid Distribution Computing*, vol. 8, no.6, pp.125-140, 2015.

[9] A. Vassiliki, M. Konstantinos, T. Konstantinos, "Dynamic QoS-aware data replication in grid environments based on data "importance" ", *Future Generation Computer Systems*, vol.28, no.3, pp.544-553, 2012.

[10] T. Liao, M. Dorigo, D. Thomas, "Improved ant colony optimization algorithms for continuous and mixed discrete-continuous optimization problems", *Proc. of 2011 Proceedings of the 34th International Convention*, pp.553-558, 2011.

[11] M. Pedemonte, S. Nesmachnow, H. Cancela, " A survey on parallel ant colony optimization", *Applied Soft Computing*, vol.11, no.8, pp.5181-5197, 2011.

[12] S. Nesmachnow, "Parallel multiobjective evolutionary algorithms for batch scheduling in heterogeneous computing and grid systems", *Computational Optimization and Applications*, vol.55, no.2, pp. 515-544, 2013.

[13] C. Ling, S. Hai-Ying, W. Shu, "A parallel ant colony algorithm on massively parallel processors and its convergence analysis for the travelling salesman problem", *Information Sciences*, vol.199, pp.31–42, 2012.

[14] J. A. Zounmevo, A. Afsahi, "An efficient MPI message queue mechanism for large-scale Jobs", *Proc. of International Conference on Parallel and Distributed Systems*, pp. 464-471, 2012.

[15] A. Sulistio, U. Cibej, S. Venugopal, B. Robic, R. Buyya, "A toolkit for modelling and simulating data grids: an extension to gridsim", *Concurrency & Computation Practice & Experience*, vol. 20, no.13, pp.1591-1609,2008.

[16] E. J. Korpela, "SETI@home, BOINC, and volunteer distributed computing", *Annual Review of Earth & Planetary Sciences*, vol. 40, No.40, pp.69-87, 2012.

Xiaohong Kong was born on December 14, 1972**.** She received the PhD degree in Light Industry Information Technology and Engineering from Jiangnan University. Currently, she is a researcher (assistant professor) at Henan Institute of Science and Technology, China. Her major research interests include automatic control and intelligent algorithm.

Hao Guo was born on Oct. 28, 1980**.** He received the PhD degree in System Science from Kobe University of Japan. Currently, he is a researcher (assistant professor) at Henan Institute of Science and Technology, China. .

Wenqiang Yang was born on Oct. 19, 1984**.** He received the PhD degree in Control Theory and Control Engineering from Shanghai University of China. Currently, he is a researcher (assistant professor) at Henan Institute of Science and Technology, China.