A Novel Grid Workflow Scheduling Using Heuristics Based Approach

Lathigara Amit Maheshbhai[1], Piyush V. Gohel[2] , Jagruti D. Goswami[3] and Nirav V. Bhatti[4]

[1]*Research Scholar, Department of CSE,*
Adhiyamaan College of Engineering, Hosur (Tamilnadu)

[2]*P.G.Scholar, Department of CSE,*
NITTTR, Chandigarh

[3]*P.G. Scholar, Department of CSE,*
RCW College, Jaipur, Rajasthan

[4]*Asst.Prof.,Department of MCA*
RK University, Rajkot,Gujarat

***Abstract* - The advent of Grid environments made feasible the solution of computational intensive problems in a reliable and cost-effective way. As workflow systems carry out more complex and mission-critical applications, Quality of Service (QoS) analysis serves to ensure that each application meets user requirements. In that frame, we present a novel Ant colony algorithm with a quick convergence of ant to move from source to destination which allows the mapping of workflow processes to Grid provided services assuring at the same time end-to-end provision of QoS (cost, makespan and reliability) based on user-defined parameters and preferences. We also demonstrate the operation of the implemented algorithm using seven different heuristics and evaluate its effectiveness using a Grid scenario.**

***Keywords* - Ant colony optimization (ACO), grid computing, workflow scheduling.**

I. Introduction

Grid computing [1] is increasingly considered as an infrastructure able to provide distributed and heterogeneous resources in order to deliver computational power to resource demanding applications in a transparent way [2]. Built on pervasive internet standards, Grids allow organizations to share computing and information resources across department and organizational boundaries in a secure and highly efficient manner. Grids support the sharing, interconnection and use of diverse resources, integrated in the framework of a dynamic computing system.

When processing a computing application in grids, we often regard the application as a workflow. Workflow is a wide concept in technology. In grid environments, we can define workflow as a collection of atomic tasks that are processed in a specific order to accomplish a complicated goal [3]. The workflow model based on loosely coupled coordination of atomic tasks has become one of the most attractive paradigms for grid computing applications [4].

One of the most challenging problems in grid computing is to schedule the workflow to achieve high performance. Usually, a workflow is given by a directed acyclic graph (DAG) [5] in which the nodes represent individual application tasks and the directed arcs stand for precedence relations between the tasks. The scheduler has to assign the tasks to heterogeneously distributed computing sites to process with the objective to achieve the customers' quality of service (QoS) requirements as well as to optimize the performance. As scheduling in a DAG is NPcomplete [6] in general, the workflow scheduling problem is complicated and highly critical to the performance of a computational grid.

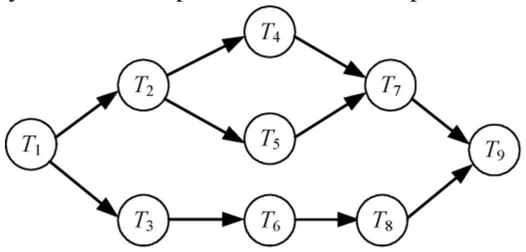

Fig. 1 Workflow representation with DAG

In the past few years, researchers have proposed scheduling algorithms for parallel system [7 - 8]. However, the problem of grid scheduling is still more complex than the proposed solutions. Therefore, this issue attracts the interests of the large number of researchers [9].

Current systems [10] of grid resource management was surveyed and analyzed based on classification of scheduler organization, system status, scheduling and rescheduling policies. However, the characteristics and various techniques of the existing grid scheduling algorithms are still complex particularly with extra added components.

At the present time, job scheduling on grid computing is not only aims to find an optimal resource to improve the overall system performance but also to utilize the existing resources more efficiently. Recently, many researchers have been studied several works on job scheduling on grid environment. Some of those are the popular heuristic algorithms, which have been developed, are min-min [11], the fast greedy [11], tabu search [11] and an Ant System [12].

The heuristic algorithms proposed for job scheduling in [11] and [12] rely on static environment and the expected value of execution times. H. Casanova et al. [13] and R. Baraglia et al. [14] proposed the heuristic algorithms to solve the scheduling problem based on the different static data, for example, the execution time and system load. Unfortunately, all of information such as execution time and workload cannot be determined in advance of dynamic grid environments.

In 1999, the Ant Colony Optimization (ACO) metaheuristic was proposed by Dorigo, Di Caro and Gambardella, which has been successfully used to solve many NP-problem, such as TSP, job shop scheduling, etc. In the past few years, several researchers proposed solutions to solve grid scheduling problem by using ACO.

Several studies have been trying to apply ACO for solving grid scheduling problem. Z. Xu et al. [15] proposed a simple ACO within grid simulation architecture environment and used evaluation index in response time and resource average utilization. E. Lu et al. [16] and H. Yan et al. [17] also proposed an improved Ant Colony algorithm, which could improve the performance such as job finishing ratio. However, they have never used the various evaluation indices to evaluate their algorithm.

This paper focuses on large-scale workflow scheduling problems in a global grid environment with various QoS parameters. In our model, three of the basic QoS parameters are addressed: reliability, time, and cost. These parameters are important for a grid application and their characteristics are quite different.

II. PROBLEM STATEMENT

We focus on the scheduling module in this paper. The scheduling problem involves a workflow application the services of which can be implemented by different GSPs. For the same service, GSPs charge higher price for short-makespan implementation and lower price for long-makespan implementation. The scheduling problem is to allocate each service to a GSP so that the workflow can be done within the users' deadline requirements and the cost is minimized. The objective of the scheduling algorithm is to find a schedule that optimizes the user-preferred QoS parameter and satisfies all QoS restrictions.

Generally, workflow applications can be modeled as a directed acyclic graph (DAG) $G = (V, A)$. Let n be the number of services in the workflow. The set of nodes $V = \{S_1, S_2, \dots, S_n\}$ corresponds to the services of the workflow. The set of arcs A represents precedence relations. An arc is in the form of (S_i,S_j), where S_i is called the parent service of S_j, and S_j is the child service of S_i. Typically in a workflow, a child service cannot be executed until all of its parent services have been completed.

Each Task T_i $(1 \le i \le n)$ has an implementation domain Si = $\{S_i^1, S_i^2, \dots, S_i^{mi}\}$, where S_i^j $(1 \le j \le m_i)$ represents a service instance provided by a GSP and m_i is the total number of available service instances for T_i. The properties of a service instance can be denoted as a group of four variables (S_i^j.g, S_i^j.r, S_i^j.t, S_i^j.c). S_i^j.g means that the GSP of S_i^j. S_i^j.r, S_i^j.t and S_i^j.c stand for reliability, execution time, and cost of S_i^j respectively.

III. ACO ALGORITHM FOR THE SCHEDULING PROBLEM

In this paper, we apply the ant colony system (ACS) algorithm, which is one of the best ACO algorithms by now, to tackle the workflow scheduling problem in grid applications. Informally, the algorithm can be viewed as the interplay of the following procedures:

1) Initialization of algorithm: All pheromone values and parameters are initialized at the beginning of the algorithm.

2) Initialization of ants: A group of *M* artificial ants are used in the algorithm. In each iteration, each ant randomly selects a constructive direction and builds a sequence of tasks.

3) Solution construction: M ants set out to build *M* solutions to the problem based on pheromone and heuristic values using the selection rule of the ACS algorithm.

4) Local pheromone updating: Soon after an ant maps a service instance S_i^j to task T_i, the corresponding pheromone value is updated by a local pheromone updating rule.

5) Global pheromone updating: After all ants have completed their solutions at the end of each iteration, pheromone values corresponding to the best-so-far solution are updated by a global pheromone updating rule.

6) Terminal test: If the test is passed, the algorithm will be ended. Otherwise, go to step 2) to begin a new iteration.

A. Pheromone and Heuristic Information

Pheromone and heuristic information are the most important factors of an ACO algorithm. In general, pheromone is used to record the historical searching experiences and bias the ants' searching behavior in future. On the other hand, heuristic information is some problem-based values to guide the search direction of ants. As the scheduling problem is mainly to map all tasks in the abstract workflow to service instances to form a concrete workflow, we denote the pheromone value of mapping service instance S_i^j to task T_i as τ_{ij}, and the heuristic information value of mapping S_i^j to task T_i as η_{ij}.

At the beginning of the algorithm, we set all pheromone
values to an initial value τ_0 , i.e.,

$\tau_{ij} = \tau_0,\quad 1 \le i \le n,\ 1 \le j \le m_i$

Moreover, as there are multiple QoS parameters with different characteristics in the considered model, we defined seven heuristics for the algorithm as follows.

1) Heuristic A: Reliability Greedy (RG): The RG heuristic biases the artificial ants to select the service instances with higher reliabilities. Suppose that an ant's heuristic type is the RG heuristic, then the heuristic value of mapping S_i^j to T_i (denoted as RG_{ij}) is set to

$$\eta_{ij} = RG_{ij} \frac{S_i^j.r - \min\_reliability_i + 1}{\max\_reliability_i - \min\_reliability_i + 1}$$

where $\text{min\_}reliability_i = min_{1 \le j \le m_i}\{S_i^j.r\}$ and $\text{max\_}reliability_i = max_{1 \le j \le m_i}\{S_i^j.r\}$ According to above equation, a service instance with higher reliability will be associated with a higher heuristic value. It also certifies that the value of η_{ij} is normalized $\in (0, 1]$. The reason of adding 1 in both numerator and denominator is to prevent the case of zero divided.

2) Heuristic B: Time Greedy (TG): The TG heuristic biases the artificial ants to select the service instances with shorter execution time. Suppose that an ant's heuristic type is the TG heuristic, then the heuristic value of mapping S_i^j to T_i (denoted as TGij) is set to

$$\eta_{ij} = TG_{ij} = \frac{max\_time_i - S_i^j.t + 1}{max\_time_i - min\_time_i + 1}$$

where $\text{min\_}time_i = min_{1 \le j \le m_i}\{S_i^j.t\}$ and $\text{max\_}time_i = max_{1 \le j \le m_i}\{S_i^j.t\}$ According to above equation, a service instance with shorter execution time will be associated with a higher heuristic value and $\eta_{ij} \in (0, 1]$.

3) Heuristic C: Cost Greedy (CG): The CG heuristic biases the artificial ants to select the service instances with lower cost. Suppose that an ant's heuristic type is the CG heuristic, then the heuristic value of mapping S_i^j to T_i (denoted as CGij) is set to

$$\eta_{ij} = CG_{ij} = \frac{max\_cost_i - S_i^j.c + 1}{max\_cost_i - min\_cost_i + 1}$$

where $\text{min\_}cost_i = min_{1 \le j \le m_i}\{S_i^j.c\}$ and $\text{max\_}cost_i = max_{1 \le j \le m_i}\{S_i^j.c\}$ According to above equation, a service instance with lower cost will be associated with a higher heuristic value and $\eta_{ij} \in (0, 1]$.

*4) Heuristic D: Suggested Deadline (SD)**:*** There are always tradeoffs between QoS parameters. For example, a service instance with shorter duration may have higher cost or lower reliability. With the consideration of such tradeoffs and the restriction of deadline, the SD heuristic biases the artificial ants to select the just-in-time service instances. To achieve this objective, we assign suggested deadlines to every task in the abstract workflow based on the user defined deadline of the application. Suppose that an ant's heuristic type is the SD heuristic, then the heuristic value of mapping S_i^j to T_i (denoted as SD_{ij}) is set to

$$\eta_{ij} = SD_{ij} = \frac{\max\{|\max\_time_i - SD_i|, |SD_i - \min\_time_i|\} - |s_j^i.t - SD_i| + 1}{\max\{|\max\_time_i - SD_i|, |SD_i - \min\_time_i|\} + 1}$$

According to it, a service instance, the execution time of which is closer to SD_i, will be associated with a higher heuristic value and $\eta_{ij} \in (0, 1]$.

5) Heuristic E: Suggested Budget (SB): Similar to the SD heuristic, the SB heuristic biases the artificial ants to select the service instances with just-within-budget cost. To achieve this objective, for each task T_i, we assign a suggested budget SB_i based on the user-defined budget of the application. Suppose that an ant's heuristic type is the SB heuristic, then the heuristic value of mapping S_i^j to T_i (denoted as SB_{ij}) is set to

$$\eta_{ij} = SB_{ij} = \frac{\max\{|\max\_cost_i - SB_i|, |SB_i - \min\_cost_i|\} - |s_j^i.t - SB_i| + 1}{\max\{|\max\_cost_i - SB_i|, |SB_i - \min\_cost_i|\} + 1}$$

According to it, a service instance, the cost of which is closer to SB_i, will be associated with a higher heuristic value and $\eta_{ij} \in (0, 1]$.

6) Heuristic F: Time/Cost (TC): The TC heuristic considers the effectiveness of both time and cost of a service instance. It integrates the TG heuristic with the CG heuristic. Suppose that an ant's heuristic type is the TC heuristic, then the heuristic value of mapping S_i^j to T_i (denoted as TC_{ij}) is set to

$$\eta_{ij} = TC_{ij} = \frac{1}{2}(TG_{ij} + CG_{ij})$$

According to it, a service instance with shorter execution time and lower cost will be associated with a higher heuristic value and η_{ij} to the interval (0, 1].

7) Heuristic G: Overall Performance (OP): The effectiveness of all QoS parameters (including reliability, time, and cost) is considered in the OP heuristic. It unites the TG, CG, and RG heuristics together. Suppose that an ant's heuristic type is the OP heuristic, then the heuristic value of mapping S_i^j to T_i (denoted as OP_{ij}) is set to

$$\eta_{ij} = OP_{ij} = \frac{1}{3}(TG_{ij} + CG_{ij} + RG_{ij})$$

According to it, a service instance with shorter execution time, lower cost, and higher reliability will be associated with a higher heuristic value and $\eta_{ij} \in (0, 1]$.

Based on different user-defined QoS preferences, the algorithm uses different heuristics. 1) If the objective of the algorithm is to optimize reliability, the algorithm will use all of these seven heuristics. Therein, the RG and OP heuristic are used to find service instances with high reliability, and the other heuristics are applied to ensure that the cost and makespan of the schedule meet the QoS constraints. 2) If the objective is to optimize makespan, only the TG, CG and TC heuristic will be used. The TG and TC heuristic are used to find the service instances with shorter execution time, and the CG, and TC heuristic are used to search for the service instances that satisfy the budget constraints. In this case, reliability constraints can be achieved by simply not choosing the service instances the reliability of which is lower than the reliability constraint, so the RG and OP heuristic is not needed. 3) If the objective is to optimize the cost, only the TG, CG, and TC heuristic will be used.

B. Construction of Solution Schedules

In every iteration of the algorithm, a group of *M* ants sets out to build solutions. The procedure of solution construction can be divided into two steps.

1) Initialization of Ants: At the beginning of each iteration, every ant randomly selects a constructive direction (forward or backward). A forward ant will normally traverse the workflow network in terms of the given precedence relations. Oppositely, a backward ant will begin the searching from the ending node of the DAG and reverse the directions of all arcs, just as what we do when calculating the value of backward earliest start time. The strategies of scheduling from both directions enable the algorithm to explore more different solutions for DAG-based scheduling problems, which have been proven in [18].

2) Solution Construction: In this step,*M* artificial ants build *M* solutions to the problem. Each ant maintains a building process and all ants construct their solutions in parallel. In each iteration, every ant iterates this selection scheme for *N* times so that *N* tasks are mapped to *N* corresponding service instances and a complete solution schedule is consequently built. In some case, if the quality of a partial solution built by an ant is already worse than the best-so-far ant, this partial solution will be deserted. Each ant monitors and records the QoS (Cost, Makespan and Reliability) parameters during the construction of solution.

C. Pheromone Management

1) Pheromone Initialization: In the ACS algorithm, all pheromone values are initially set to τo.

$$\tau o = \begin{cases} \frac{\min\_Reliability}{\max\_Reliability}, \text{Reliability Optimization} \\ \frac{\min\_Makespan}{\max\_Makespan}, \text{Makespan Optimization} \\ \frac{\min\_Cost}{\max\_Cost}, \text{Cost Optimization} \end{cases}$$

where, *min_Reliability* is the minimum reliability of all service instances and *max_Reliability* = 100. *min_Makespan* is the estimated minimum makespan of the workflow application, and *max_Makespan* is the estimated maximum makespan.

2) Local Pheromone Updating: In the ACS algorithm, immediately after an ant has mapped a task T_i to a service instance S_i^j , the local pheromone updating rule is applied to reduce the attraction of S_i^j for the later ants. The local pheromone updating rule is given by the following equation:

$$\tau ij = \frac{1}{(1-\rho)} \tau ij + \rho\tau_0$$

where, ρ *within* (0, 1) is a parameter. As $\tau 0$ is the minimum value of all pheromone values, the function of the local updating rule is to decrease the value of τij to enhance diversity of the algorithm.

3) Global Pheromone Updating: Global updating takes place after all ants have built their solutions.

IV. EXPERIMENTAL ANALYSIS

We first test the parameters of the ACS algorithm. There are mainly three parameters in the algorithm: β and q_0 in the pseudorandom proportion selection rule and ρ in the pheromone updating rule. In our experiments, we set $\rho = 0.1$, which is the same as the suggestion given by the traditional ACS algorithm for traveling salesman problem (TSP). We find that this configuration still performs well in the workflow scheduling problem.

We simulated the result and during simulation, resources with advanced reservation support have been created as shown in Table-1. We proposed seven different heuristics in the algorithm, including RG, TG, CG, SD, SB, TC and OP. We simulated the algorithm and it was found to be working efficiently and effectively. Experimental test carried out for a 60 task as an input set to ascertain the efficiency of the algorithm. We executed experiments to compare the performance of these heuristic schemes and the results are shown in Figs. 2 to 6 respectively.

Resource Name in Simulation	**Simulated Resource Characteristics Vendor, Resource Type, Node OS, No of PEs**	**Equivalent Resource in World Wide Grid (Hostname, Location)**	**A PE SPEC/ MIPS Rating**	**Resource Manager Type**	**Price (G$/ PE time unit)**
Resource_0	Compaq, Alpha Server, CPU, OSF1, 4	grendel.vpac.org, VPAC, Melbourne, Australia	515	ADVANCE RESERVATION	8
Resource_1	Sun, Ultra, Solaris, 4	hpc420.hpcc.jp, AIST, Tokyo, Japan	377	ADVANCE RESERVATION	4
Resource_2	Sun, Ultra, Solaris, 4	hpc420-1.hpcc.jp, AIST, Tokyo, Japan	377	ADVANCE RESERVATION	3
Resource_3	Sun, Ultra, Solaris, 2	hpc420-2.hpcc.jp, AIST, Tokyo, Japan	377	ADVANCE RESERVATION	3
Resource_4	Intel,	barbera.cnuce.cnr.it,	380	ADVANCE	2

	Pentium/VC820, Linux, 2	CNR, Pisa, Italy		RESERVATION	
Resource_5	SGI, Origin 3200, IRIX, 6	onyx1.zib.de, ZIB, Berlin, Germany	410	ADVANCE RESERVATION	5
Resource_6	SGI, Origin 3200, IRIX, 16	Onyx3.zib.de, ZIB, Berlin, Germany	410	ADVANCE RESERVATION	5
Resource_7	SGI, Origin 3200, IRIX, 16	mat.ruk.cuni.cz, Charles U., Prague, Czech Republic	410	ADVANCE RESERVATION	4
Resource_8	Intel, Pentium/VC820, Linux, 2	marge.csm.port.ac.uk, Portsmouth, UK	380	ADVANCE RESERVATION	1
Resource_9	SGI, Origin 3200, IRIX, 4	green.cfs.ac.uk, Manchester, UK	410	ADVANCE RESERVATION	6
Resource_10	Sun, Ultra, Solaris, 8	pitcairn.mcs.anl.gov, ANL, Chicago, USA	377	ADVANCE RESERVATION	3

Table 1 WWG testbed resources used in simulation.

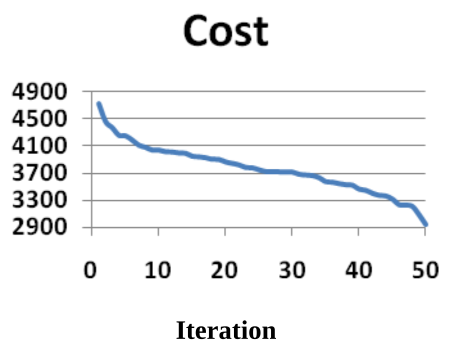

Fig. 2a Performance of Cost in the case of Cost Greedy Heuristic.

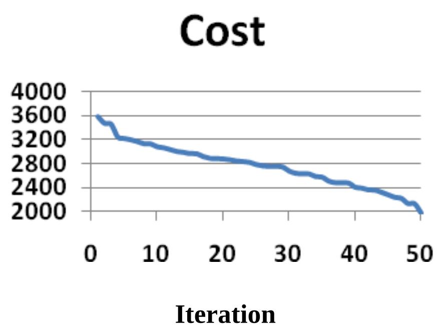

Fig. 3a Performance of Cost in the case of Time Greedy Heuristic.

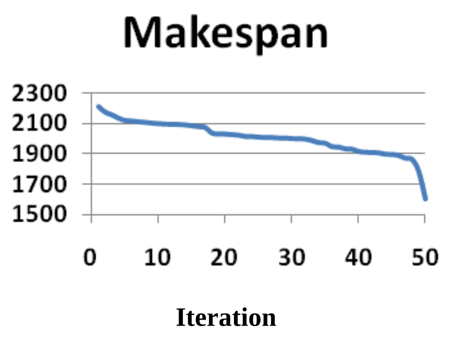

Fig. 2b Performance of Makespan in the case of Cost Greedy Heuristic.

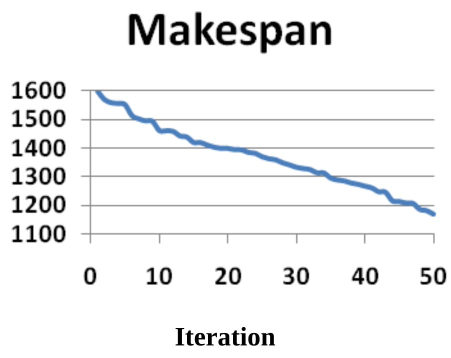

Fig. 3b Performance of Makespan in the case of Time Greedy Heuristic.

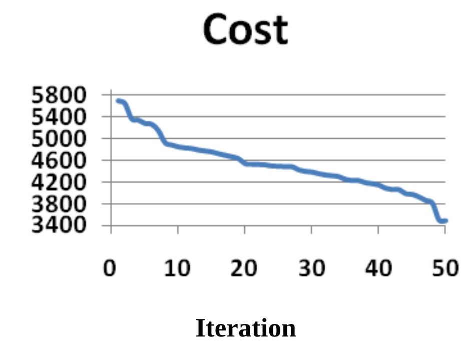

Fig. 4a Performance of Cost in the case of Reliability Greedy Heuristic.

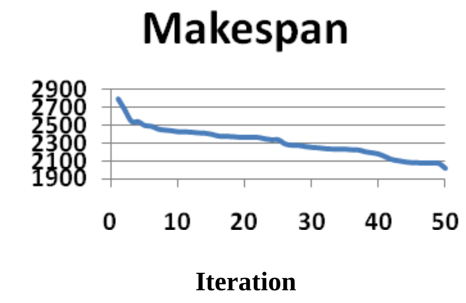

Fig. 4b Performance of Makespan in the case of Reliability Greedy Heuristic.

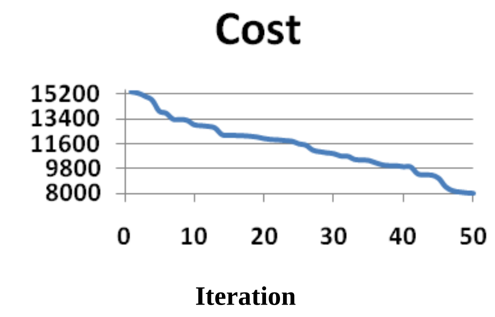

Fig. 5a Performance of Cost in the case of Time/Cost Heuristic.

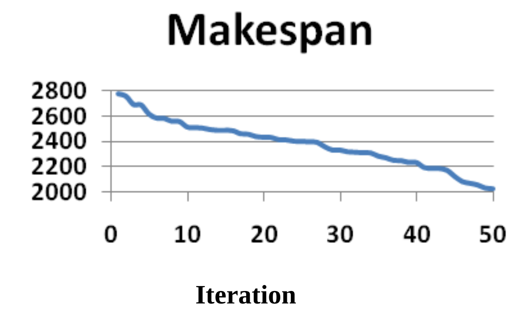

Fig. 5b Performance of Makespan in the case of Time/Cost Heuristic.

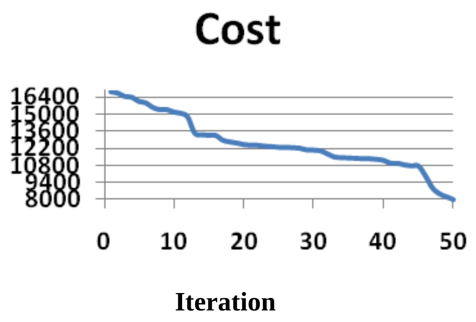

Fig. 6a Performance of Cost in the case of Overall Performance Heuristic.

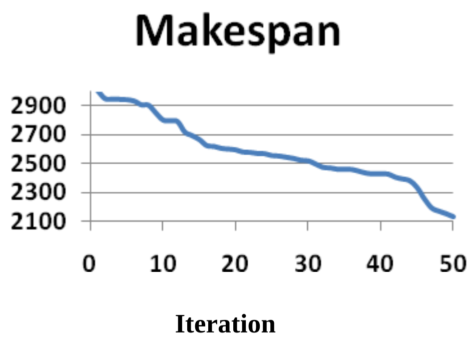

Fig. 6b Performance of Makespan in the case of Overall Performance Heuristic.

V. CONCLUSION

A novel ant colony optimization with modified local pheromone updating rule for a large-scale workflow scheduling problem in computational grids has been proposed. The scheduling algorithm is designed for workflow applications in market driven or economy-driven grids under the service-oriented architecture. In the algorithm, different QoS parameters are considered, including reliability, time, and cost. Users are allowed to define QoS constraints to guarantee the quality of the schedule. Moreover, the optimizing objective of the algorithm is based on the user-defined QoS preferences. We proposed seven new heuristics for this problem. Experimental results demonstrate the effectiveness of the proposed algorithm.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their constructive suggestions and insightful comments that have helped enrich the content and improve the presentation of this paper.

REFERENCES

[1] I. Foster, C. Kesselman, The Grid: Blueprint for a Future Computing Infrastructure, Morgan Kaufmann Publishers, USA, 1999.
[2] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the Grid: Enabling scalable virtual organizations, International Journal Supercomputer Applications 15 (3) (2001).

[3] D. Kyriazis *et al.*, “An innovative workflow mapping mechanism for grids in the frame of quality of service,” *Future Gen. Comput. Syst.*, to be published.

[4] R. Prodan and T. Fahringer, “Overhead analysis of scientific workflows in grid environments,” *IEEE Trans. Parallel Distrib. Syst.*, to be published.

[5] Z. Shi and J. J. Dongarra, “Scheduling workflow applications on processors with different capabilities,” *Future Gen. Comput. Syst.*, vol. 22, pp. 665–675, 2006.

[6] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.

[7] D.G. Feitelson, L. Rudolph, U. Schwiegelshohn, K.C. Sevcik, and P. Wong. “Theory and practice in parallel job scheduling”, In *3rd Workshop on Job Scheduling Strategies for Parallel Processing*, volume LNCS 1291, pages 1–34, 1997.

[8] J. Krallmann, U. Schwiegelshohn, and R. Yahyapour. “On the design and evaluation of job scheduling algorithms”, In *5th Workshop on Job Scheduling Strategies for Parallel Processing*, volume LNCS 1659, pages 17–42, 1999.

[9] K. Li, “Job scheduling and processor allocation for grid computing on Metacomputers ”, *Journal of Parallel and Distributed Computing, Elsevier,* 2005

[10] K. Krauter, R. Buyya and M. Maheswaran, “A taxonomy and survey of Grid resource management systems for distributed computing”, SoftwarePract. Exp. 2 (2002) 135–164.

[11] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen and R. F. Freund (2001), “A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems”, *Journal of Parallel and Distributed Computing*. Vol.61(6): Pages 810-837.

[12] G. Ritchie and J. Levine, “A fast, effective local search for scheduling independent jobs in heterogeneous computing environments”.

[13] H. Casanova, A. Legrand, D. Zagorodnov and F. Berman, “Heuristics for scheduling parameter sweep applications in Grid environments”, *in: Heterogeneous ComputingWorkshop 2000,* IEEE Computer Society Press, 2000, pp. 349–363.

[14] R. Baraglia, R. Ferrini, and P. Ritrovato, “A static mapping heuristics to map parallel applications to heterogeneous computing systems”, Research articles. *Concurrency and Computation: Practice and Experience*, 17(13):1579–1605, 2005.

[15] Z. Xu, X. Hou and J. Sun, “Ant Algorithm-Based Task Scheduling in Grid Computing”, *Electrical and Computer Engineering, IEEE CCECE 2003, Canadian Conference*, 2003.

[16] E. Lu, Z. Xu and J. Sun, “An Extendable Grid Simulation Environment Based on GridSim”, *Second International Workshop, GCC 2003,* volume LNCS 3032, pages 205–208, 2004.

[17] H. Yan, X. Shen, X. Li and M. Wu, “An Improved Ant Algorithm for Job Scheduling in Grid Computing”, *In Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, 18-21 August 2005.

[18] D. Merkle, M. Middendorf, and H. Schmeck, “Ant colony optimization for resource-constrained project scheduling,” *IEEE Trans. Evol. Comput.*, vol. 6, no. 4, pp. 333–346, Aug. 2002.