Journal of Scientific & Industrial Research
Vol 74, July 2015, pp. 377-380

A Hybrid Ant Colony Optimization Algorithm for Job Scheduling In Computational Grids

E S Kumar[1]*, A Sumathi[2] and H A Zubar[3]

*[1]Adhiyamaan College of Engineering, Hosur- 635109, India
[2]Adhiyamaan College of Engineering, Hosur- 635109, India
[3]Knowledge Institute of Technology, Salem- 637504, India

Received 10 January 2014; revised 26 April 2015; accepted 18 June 2015

Grid computing links disparate computers having free resources to form a low cost infrastructure. Grid computing can provide enormous opportunities for organizations to use resources from multiple geographical locations. For efficient utilization of available resources, grid scheduling plays an important role in the grid system. Scheduling is challenging in grid due to the unique characteristics. Also, the complexity of scheduling algorithm is NP-Complete. In this study, a local search heuristic by way of multipoint mutation is introduced on the popular swarm intelligence inspired meta-heuristic, Ant Colony Optimization. Experiments show the proposed technique improves the Makespan and converges faster than conventional Ant Colony Optimization.

Keywords: Grid computing, Grid Scheduling, Ant Colony Optimization (ACO), Makespan, local search

Introduction

Grid computing is a high performance computing environment to solve large scale computational demands. The major components in the grid architecture for effective utilization of resources include the user interface to manage and upload jobs, directory brokering, Grid information services[1], resource allocation and monitoring, job scheduling[2], secure access to resources and services and the actual fabric containing clock cycles, storage and sensors. To effectively utilize the distributed resources and maintain quality of service, scheduling plays a very important role. However scheduling algorithms in Grid face issues of heterogeneity for both jobs and resources.The phases of grid scheduling include resource discovery, resource selection, scheduling and monitoring. The resource monitor queries the Resource Discovery System like MDS of Globus toolkit and collects the resources available in the network. The data thus collected is available locally for the resource mapper to map the ideal resources for the job in hand. Once the resources are selected, the scheduling algorithm maps the job to specific resources. Once the job is allocated to resources, the task manager allocates the task and monitors till output is given back to the originating application.Grid scheduling algorithms can either be local or global. In the global scheduling scenario, the scheduling technique can be either static or dynamic. In static technique every task is assigned once to the resource leading to a good prediction of computation cost. Dynamic scheduling techniques are used when the jobs arrival is dynamic in nature. Various heuristic based scheduling algorithms have been proposed in literature. The scheduling outcome of heuristic algorithms is suboptimal in nature proving Grid scheduling as NP-complete. Appropriate scheduling algorithm is selected in a given grid based on the task, machine and network connectivity. With grid technology maturing, educational institutions, public institutions and organizations requiring high resources will be rarely exploited.[3] Heuristic algorithms are a breed of algorithms which try to find best solutions among all possible solutions[4,5]. Heuristic algorithms do not guarantee that they will find the best solutions. Heuristic algorithms are generally fast and can be broadly classified into divide and conquer, branch and bound, dynamic programming and local search. For NP complete problems branch and bound, dynamic programming techniques are generally not preferred due to their high time complexity.Meta-heuristic algorithms are heuristic algorithms designed to find a lower-level heuristic that may provide a sufficiently

*Author for correspondence
E-mail: saraninfo@gmail.com

good solution to an optimization problem. Swarm Intelligence[6] and Evolutionary Algorithms[7] are popular Meta-heuristic algorithms.

The collective decentralized behavior exhibited by natural or artificial phenomena is the foundation for Swarm Intelligence. Evolutionary algorithms are based on biological evolution such as reproduction, recombination and selection. Meta-heuristic like Simulated Annealing (SA), Genetic Algorithm (GA)[8,9,10], ACO[11,12,13], Particle Swarm Optimization (PSO)[14,15] have been proposed for grid scheduling as they generally produce higher quality results than simple heuristics[16]. In this study the popular ACO algorithm is investigated for the computational grid scheduling problem. An improved ACO based on mutation concept of Evolutionary Algorithm in ACO is proposed.

ANTS in Grid Scheduling

Ant Colony Optimization (ACO) is a popular meta-heuristic algorithm used extensively to solve NP problems[17, 18]. ACO has been applied to solve discrete, static, dynamic and combinational optimization problems. ACO is inspired by the foraging behavior of ants which enables the ant to find the shortest path from its nest to the food source[19, 20]. Ants use pheromones to mark their trail as they walk from nest to food source. Other ants select trails with high pheromone concentrations[21].

The ACO flow is shown

a) Start
b) Initialize the pheromone table
c) Select ants randomly to walk to specific jobs
d) Every ant must walk to next job, depending on probability distribution
e) Compute the length of the path traveled by each ant, and allocates a quantity amount of pheromone to the path, according to the length of its path
f) Perform a local update
g) Compute whether a better solution is obtained in this time step than the last; if so, then perform a global update on the solution; repeat Steps c to g
h) End

Weight of a resource is the pheromone value present on the ant system's path where the larger weight is considered as the resource with better computation power. Data is collected form information server by the scheduler and is used to calculate the weight of a resource. Each pheromone of resource is stored in scheduler where it uses the value as a parameter for ACO algorithm. Finally the resource is selected by scheduler with scheduling algorithm and the job is sent to selected resources [22].Let M be the resource $\{m_1,m_2,\ldots.m_m\}$ and J be the set of jobs $\{j_1,j_2,\ldots.j_n\}$ where n>m. The relationship between jobs and tasks can be represented by the graph $G = (M,\{CT_{iJ}\}_{mxn}$ where CT is the completion time of job i in resource j.

The initial pheromone trail can be represented by equation (1)

$$\tau 0=\frac{1}{m.(\Sigma_{i=1}^{m} Ti_{Actual}+\Sigma_{i=1}^{m}(\Sigma_{j=1}^{n} Tij)_{Expected})} \quad \ldots (1)$$

Where $(\Sigma_{i=1}^{m} Ti_{Actual}+\Sigma_{i=1}^{m}(\Sigma_{j=1}^{n} Tij)_{Expected})$ is the actual and expected tardiness.

Artificial ants are created where each ant starts with unscheduled job in machine and job route is built in machine until feasible solution is obtained. For next resource m to be scheduled from current resource i the rule applied by the ant as in equation (2):

$$m = \begin{cases} \arg\max[\tau(i,u).[_{u\subset U}\eta(i,u)]^{\beta} & \text{if } q\leq qo \\ S & \text{otherwise} \end{cases} \quad \ldots (2)$$

Where q is the random number in range [0, 1] and qo is the parameter with $O\leq\rho\leq1$, $\tau(i,u)$, is the pheromone value associated for the resource path i to u when u does not have any scheduled jobs. S is computed by (3):

$$p(i,m) = \begin{cases} \frac{[\tau(i,m)][\eta(i,m)]^{\beta}}{\Sigma ucu[\tau(i,u)][\eta(i,u)]^{\beta}} & \text{if } m\in U \\ & \text{otherwise} \end{cases}$$

$$\eta(i,m) = [\frac{Mj}{Bandwidth_i}+\frac{Tj}{CPU\_speed_{i^*}(1\text{-}load_i)}]^{-1} \quad \ldots (3)$$

Where η (i,m) is based on the grid system parameters and is given equation (3)

Mj is the size of a given job j, Tj is the CPU time needed for job j, CPU_speed_i is the CPU speed, $load_i$ is the current load and $Bandwidth_i$ is the bandwidth between the scheduler and resource. Other techniques to estimate resource execution time was also presented[23,24,25,26,27].To update the local rule pheromone level are changed as the ants visit each edge and resources become less desirable when the pheromone values are decreased. The updation is given by

$$\tau(i,u) = (1\text{-}\rho),\tau(i,j)+\rho\tau_0$$

Where $\rho(O<\rho\leq1)$

Proposed ACO

In proposed mechanism each ant is allocated a set of jobs it has to visit in a sequence to form initial solution of ants with jobs. A 3-opt mutation operation on the parent solutions to create child solutions to search optimal solutions in the search space are introduced. Mutation aims at mutating the resources randomly for producing a new solution. The mutation operation proposed is given in the steps described:

- Two resources are selected form parent solution and mutation points from each mutating resource are selected as in fig.1.
- By exchanging the tasks in different resources, the child solutions are generated as in fig.1.
- The 3-opt local search is applied on the children to improve the local optimality.

The mutation operation may violate resource capability which is avoided by providing a high cost to the candidate solution so that it may not be selected in the forthcoming search. The obtained solutions are tested if an overall improvement in the desired objectives is obtained. Mutation introduces more diversity and thus escaping from local optima during later stages is achieved.

Results and Discussion

Experiments were carried out using 8 ants with two scenarios. In the first scenario, 5 resources were used and jobs were from 100-900 with increments of 200. In the second scenario 50 resources were used. A total of ten runs were conducted for each experiment. Table 1 and Table 2 list the obtained makespan. From Table 1 it can be observed that the performance of proposed scheduling algorithm decreases makespan by an average of 2.66% compared to ACO. The performance improvement is seen as the number of tasks is increased. The best improvement is seen when the number of jobs is 700. With 700 jobs the proposed algorithm decreases makespan by 5.87% when compared to Min-Min scheduling and by 2.79% when compared to ACO. From Table 2, the performance improvement is much more significant when the number of resources is increased across all the job scenarios. The average improvement in the makespan is 6.99% of the proposed algorithm compared to Min-Min scheduling and 4.23% when compared with ACO. The performance improvement is uniform across all job scenarios. From fig. 2 and 3 shows the convergence

	Task			
Resources	3	1	2	
	4	5	6	7
	8	**9**	14	
	10	11	**12**	13

	Task			
Resources	3	1	2	
	4	5	6	7
	8	**9**	**14**	
	10	11	**12**	13

Fig.1-Parent Solution with Mutation points & Mutation Process (Parent and Child Solutions)

Table 1- Makespan with 5 resources

Number of jobs	Min-Min Scheduling	ACO scheduling	Proposed ACO scheduling
100	88.91	86.37	85.96
300	276.21	271.7	267.45
500	453	453.71	442.68
700	619.93	604.04	587.41
900	809.76	796.97	771.22

Table 2- Makespan with 50 resources

Number of jobs	Min-Min Scheduling	ACO scheduling	Proposed ACO scheduling
100	9.52	9.38	9.13
300	30.89	29.74	28.98
500	48.14	46.96	44.69
700	67.76	65.12	62.66
900	87.46	85.94	81.85

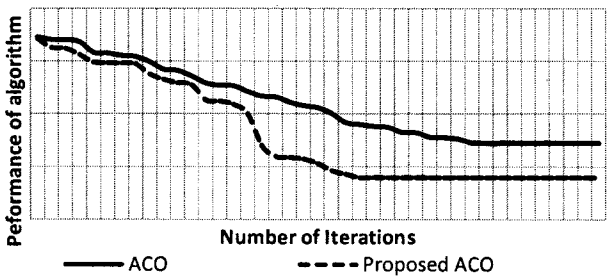

Fig.2 -Convergence characteristics when 5 resources used

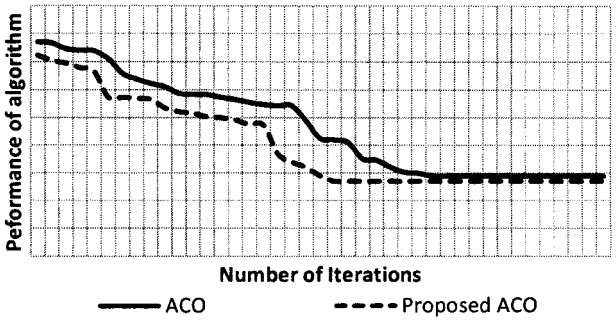

Fig.3 -Convergence characteristics when 50 resources used

characteristics when the number of jobs is fixed at 900 and the number of resources used are 5 and 50 respectively. In both the scenarios the convergence is at least 25% faster than ACO. This is due to the mutation operation improving the global search and the 3-opt optimization which searches locally better.

Conclusion

This study investigated performance improvement of grid scheduling using Min-Min algorithm and ACO. An improved ACO with mutation and 3 opt local search capability was proposed. A multi objective function is used to optimize the scheduling algorithm. The proposed algorithm was tested across various scenarios by varying the number of resources and jobs. Results obtained shows significant decrease in makespan of the proposed algorithm.

References

1 Fang X, Misra S, Xue G, & Yang D , Managing smart grid information in the cloud: opportunities, model, and applications, *Network, IEEE*, **26(4)** (2012) 32-38.

2 Rosemarry P, Singhal P, & Singh R, A Study of Various Job & Resource Scheduling Algorithms in Grid Computing, *Int J Comp Sci & Info Technol*, **6(3)** (2012) 5504-5507.

3 Sharma R, Soni V K, Mishra M K, & Bhuyan P, A survey of job scheduling and resource management in grid computing, *World Acad Sci Eng Technol*, **64(4)** (2010) 461-466.

4 Askarzadeh A & Rezazadeh A, A new heuristic optimization algorithm for modeling of proton exchange membrane fuel cell: bird mating optimizer, *Int J Ene Res*, **37(2)** (2013) 1196-1204.

5 Bagloee S A, Tavana M, Ceder A, Bozic C, & Asadi M, A hybrid meta-heuristic algorithm for solving real-life transportation network design problems, *Int J of Log Sys & Manage*, **16(1)** (2013) 41-66.

6 Parpinelli R S & Lopes H S, New inspirations in swarm intelligence: a survey, *Int J Bio-Insp Comp,* **3(1)** (2011) 1-16.

7 Derrac J, García S, Molina D & Herrera F, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, *Swarm Evol Comp*, **1(2)** (2011) 3-18.

8 Holland J H, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*, U Michigan Press, (1975).

9 Gkoutioudi K Z & Karatza H D, Multi-criteria job scheduling in grid using an accelerated genetic algorithm, *J G Comp*, **10(2)** (2012) 311-323.

10 Delavar A G, Nejadkheirallah M & Motalleb M, A new scheduling algorithm for dynamic task and fault tolerant in heterogeneous grid systems using Genetic Algorithm, *In Proc IEEE Int Conf Comp Sci & Info Technol,* **9** (2010) 408-412.

11 Bonabea E, Dorigo M, & Theraulaz G, Inspiration for optimization from social insect behaviour, *Nature*, **406** (2000) 39-42.

12 Liu X F, Zhan Z H, Du K J & Chen W N, Energy aware virtual machine placement scheduling in cloud computing based on ant colony optimization approach, *In Pro ACM Conf GECCO*, (2014) 41-48.

13 Jain A & Singh R, An innovative approach of Ant Colony optimization for load balancing in peer to peer grid environment, *In Proc IEEE Iss & Chal in Intel Compu Techs,* (2014) 1-5.

14 Kennedy J, Particle swarm optimization, *In Encyc of Mach Learn*, (2010) 760-766.

15 Izakian H, Ladani B T, Abraham A & Snasel V, A discrete particle swarm optimization approach for grid job scheduling, *Int J Innov Compu Info & Cntrl*, **6(9)** (2010) 4219-4233.

16 Kashyap R & Vidyarthi D P, Security-driven scheduling model for computational Grid using genetic algorithm, *J Grid Computing,* **11(4)** (2013) 382-387.

17 Talreja S, A Heuristic Proposal in the Dimension of Ant Colony Optimization, *Appl Math Sci*, **7(41)** (2013) 2017-2026.

18 Saleem K, Fisal N, Baharudin M A, Ahmed A A, Hafizah S & Kamilah S, Ant colony inspired self-optimized routing protocol based on cross layer architecture for wireless sensor networks, *WSEAS Trans Com*, **9(10)** (2010) 669-678.

19 Dorigo M & Birattari M, Ant colony optimization, *In Encyc of Mach Lea,* (2010) 36-39.

20 Lorpunmanee S, Sap M N, Abdullah A H, & Chompoo-inwai C, An ant colony optimization for dynamic job scheduling in grid environment, *Int J Comp & Info Sci Engg*, **1(4)** (2007) 207-214.

21 Kousalya K, & Balasubramanie P, An enhanced ant algorithm for grid scheduling problem, *Int J Comp Sci & Netw Sec*, **8(4)** (2008) 262-271.

22 Globus Toolkit v4, http://www.globus.org/toolkit/downloads/4.0.4/

23 Sarkar V, Determining average program execution times and their variance, *In Proc ACM Conf PLDI*, (1989) 298–312.

24 Park C Y, Predicting program execution times by analyzing static and dynamic program paths, *Real-Time Sys,* **5(2)** (1993) 31–62.

25 Engblom J & Ermedahl A, Modeling complex flows for worst-case execution time analysis, *In Proc IEEE Real-Time Sys Symp*, (2000) 163–174.

26 Stappert F, Altenbernd P, Complete worst-case execution time analysis of straight-line hard real-time programs, *J Sys Arch,* **46** (2000) 339–355.

27 Zhang Y, Sun W & Inoguchi Y, Predict task running time in grid environments based on CPU load predictions, *Future Gener Comp Sy,* **24(6)** (2008) 489–497.