Multi-Objective Ant Colony Optimization for Task Scheduling in Grid Computing

Nitu and Ritu Garg

Abstract Resource Management in Grid computing system is a fundamental issue in achieving high performance due to the distributed and heterogeneous nature of the resources. The efficiency and effectiveness of Grid resource management greatly depend on the scheduling algorithm. In this paper, the problem of scheduling is represented by a weighted directed acyclic graph (DAG). Ant Colony Optimization is used for scheduling tasks on resources in Grid which simultaneously pay attention to two objectives of makespan (schedule length) and the failure probability (reliability). These objectives are conflicting and it is not possible to minimize both objectives at the same time. With the help of concept of non-dominance, we are able to choose a trade-off between makespan minimization and reliability maximization. For evaluating the algorithm, ACO is compared with NSGA-II. The metrics for evaluating the convergence and diversity of the obtained non-dominated solutions by the two algorithms are reported. The results of simulation using JAVA programming language manifest that proposed approach can be used more efficiently for allocating the tasks as compared to NSGA-II.

Keywords Task scheduling · DAG · Makespan · Reliability

1 Introduction

Grid computing systems have emerged as a new environment for coordinated resource sharing and problem solving in multi-institutional virtual organizations while providing dependable, consistent, pervasive access to global resources. The

Nitu (✉) · R. Garg
Department of Computer Engineering, NIT, Kurukshetra, India
e-mail: me.nitumalhan@gmail.com

R. Garg
e-mail: ritu.59@gmail.com

M. Pant et al. (eds.), *Proceedings of the Third International Conference on Soft Computing for Problem Solving*, Advances in Intelligent Systems and Computing 259,
DOI: 10.1007/978-81-322-1768-8_12,

sharing ranges from simple file transfer to direct access to computers, software, data, and other network accessible resources. Grid resources are heterogeneous, dynamic, complex and self-autonomic in nature which makes the resource management a significantly challenging job. Job Scheduling and Resource Management are the critical issues in Grid Computing [1]. The multiprocessor task schedule problem is known to be NP-complete. In order to address this problem, many heuristics algorithms have been proposed. These heuristics are classified into different categories such as list scheduling algorithms, clustering algorithms, duplication based algorithms [2].

Ant colony algorithm is one of the effective techniques used to solve NP-complete problems. ACO is inspired by the behavior of real ant colonies in nature to search for food and to connect to each other by pheromone trails laid on paths traveled. ACO was used to solve many NP-hard problems such as traveling salesman problem, vehicle routing problem, graph coloring problem [3–5] and so on. In this paper, we apply this technique for dependent task scheduling. The input to the scheduling algorithm is a directed acyclic graph (DAG), in which the node weights represent task processing times and the edge weights represent data dependencies as well as the communication times between tasks. The multi-objective ACO algorithm uses non-dominance approach for tackling the two objectives (makespan and reliability).

2 Related Work

Task scheduling is known to be NP-complete problem and lots of heuristics and meta-heuristics techniques have been examined to solve it. Most of them can be applied to the Grid environment with suitable modifications.

Min-min [6] set the tasks which can be completed earliest with the highest priority. The main idea of Min-min is that it assigns tasks to resources which can execute tasks the fastest. Max–min [6] set the tasks which has the maximum earliest completion time with the highest priority. The main idea of Max–min is that it overlaps the tasks with long running time with the tasks with short running time.

In [7] the qualities of the different obtained solutions were compared against the Particle Swarm Optimization (PSO), the Simulated Annealing (SA) and the Genetic Algorithms (GA) Meta-heuristic. The results show that PSO and GA are highly efficient and effective in the task scheduling problem. Later these Meta-heuristics were compared against a MOEA algorithm, optimizing the makespan and flowtime objective functions. In [8], the authors propose an algorithm called Multi-Objective Resource Scheduling Approach—MORSA, which is a combination between NPGA and NSGA Algorithms. They combine the sorting algorithm of non-dominated solutions with the process of Niche Sharing to ensure diversity. In [9], a deadline-based model is proposed which first generates all feasible scheduling solutions with makespan less than a predefined deadline, and then finds the best possible solution with the maximum reliability between feasible solutions.

Ant colony optimization (ACO) is a meta-heuristic alternative for solving the complicated optimization problems [10]. There are many different kinds of ACO algorithm, i.e., Ant Colony System (ACS), Max–Min Ant System (MMAS), Rank-based Ant System (RAS), Fast Ant System (FANT) and Elitist Ant System (EAS).

3 Scheduling Problem and Formulation

Generally grid applications in e-science and e-business falls in the category of workflow applications modeled by an ordered graph called Directed Acyclic Graph (DAG). The application can be represented by G (V, E) where set V represents n number of the subtasks of the application and a set E of edges shows the dependencies among the subtasks. Each and every task is related by directed edge representing the communication directions between tasks and precedence constraints (i.e. data dependency). A directed edge $e_{ij} \in E$ indicates the data dependency constraint exists between the tasks v_i and v_j. In this model, a task can't start executing until all its parents have been completed. The value assigned to the edge represents the amount of data to be transferred between tasks if they are not executing on the same resource. If both parent and offspring tasks processed on a same resource, the value of communication time between them is considered to be zero.

In any DAG, there is always an input node, v_{entry} as a node with no parent and an output node, v_{exit} as a node with no offspring. When DAG has several entry or exit tasks, these tasks are connected to a pseudo entry-task or pseudo exit-task that has zero load weight and zero capacity edges.

A schedule is a function S: V → P assigns a task to the processor that executes it. Let $V(j, s) = \{i|s(i) = j\}$ be the set of tasks assigned to processor j.

The completion time of a processor j is calculated as

$$C_j(s) = \sum\nolimits_{i \in v(j,s)} (st_{ij} + w_{ij})$$

where st_{ij} denotes the start time of the task i on processor j. The start time of the entry task is assumed to be zero. Other tasks' start time can be computed by considering the completion time of all immediate predecessors of the task. The communication time, st_{ij} is added to the start time, if the dependent tasks are allocated to different processors. The makespan of a schedule is the time where all tasks are completed

$$C_{max} = max_j C_j(s)$$

Every processor has a constant failure rate, and let λ_j denote the failure rate of processor j. The probability that a processor j executes all its tasks successfully is given by

$$P^j_{succ}(S) = e^{-\lambda_j C_j(s)}$$

It is assumed that faults are independent, therefore, the probability that schedule S has finished correctly is

$$P_{succ} = e^{-rel(s)}$$

The reliability index (rel) is defined by

$$rel(s) = \sum_{j} \lambda_j c_j(s)$$

Minimizing the objective function rel is equivalent to maximizing the reliability of the schedule. For solving the task scheduling problem the objectives Cmax and rel are to be minimized simultaneously.

4 Proposed Algorithm

In this section we are presenting the algorithm for the dependent task scheduling in Grid environment using Ant Colony Optimization technique, which aims at achieving high reliability and reducing the makespan. The algorithm consists of two mechanisms, a ranking mechanism [11], which is a modified version of the HEFT [2, 12] and a processor assignment mechanism.

Steps for Scheduling Algorithm:

1. Set the computation cost of tasks and communication cost among them.
2. Compute upward RRank value for all tasks by traversing graph, starting from the exit task using Ranking method ().
3. Sort the task in a scheduling list by non-increasing order of upward rank value.
4. While there are unscheduled tasks in the list do

   Select the first task v_i from the list for scheduling
   For each processor p_m in the processor set $p_m \in P$ do

   - Calculate the heuristic information (n_{ij})
   - Calculate current pheromone trail value $\Delta\tau_{ij}$
   - Update the pheromone trail matrix
   - Calculate the probability matrix
   - Select the task with highest probabilities of i and j as the next task v_i to be executed on the resource P_j.
   - Remove the task v_i from the unscheduled list
   - Modify the resource free time
5. For each local solution generated by all ants, find the makespan and reliability index.
6. Apply the concept of non-dominated sorting on local solutions to find the globally best solution.

4.1 Ranking Method ()

Our algorithm has used reliability rank (RRank) attribute to compute priorities of tasks. The RRank [11] is a rank of a task, from the exit task to itself, and equal to the sum of average communication costs of edges, average computation costs and reliability overhead of tasks over all processors. Communication costs between tasks scheduled on the same processor are assumed zero, and the execution constraints are preserved.

The RRank is recursively defined as:

$$RRank(v_i) = \overline{w(v_i)} + max_{v_j \in succ(v_i)} \left\{ \overline{w(e_{i,j})} + RRank(v_j) \right\} + RC_{vi}$$

where $succ(v_i)$, the set of immediate successors of task v_i, $\overline{w(v_i)}$ is the average computation cost of task v_i, and $\overline{w(e_{i,j})}$ is the average communication cost of edge $e_{i,j}$. The RC_{vi} is the reliability overhead of task v_i and can be computed by

$$RC_{vi} = (1 - \prod_{n=1}^{m} \exp\{-\lambda_{P_n} \times w(v_i)/w(p_n)\}) \times \overline{w(v_i)}$$

The rank is computed recursively by traversing the task graph upward, starting from the exit task. For the exit task v_{exit}, the rank value is equal to

$$RRank(v_{exit}) = \overline{w(v_{exit})} + RC_{Vexit}$$

4.2 Task Assignment Mechanism

In this mechanism, tasks are assigned to the processors in such a way that makespan is reduced and system reliability is improved. In order to achieve these goals, we find the best mapping of tasks and processors applying ACO. The ant is placed at first task in the generated order and that task is mapped on one of the available resources required by that task. When each of the tasks of the system is mapped to any specific resource then solution construction for that ant is completed and a complete and feasible solution is created. Each task is mapped to a specific resource and preemption is not allowed, that resource is selected based on probability rule that depends on both pheromone on edges between tasks and resources and heuristic information that depends on objective functions.

$$P_{ij} = \left[\tau_{ij}^{\alpha} \eta_{ij}^{\beta} \right] / \sum \tau_{il}^{\alpha} \eta_{il}^{\beta}$$

where τ is the value of pheromone on edge between tasks i and resource j and η is the value of heuristic information, α and β are two constants that represents the

relative importance of pheromone trail values and heuristic information values that depends on problem considered.

After each iteration of the algorithm i.e. after all the ants completed their solution construction, Global pheromone updating rule is performed to increase the value of pheromone on the edges of the solution that is found to be global best in case of single objective and Non-dominated solutions in case of multi-objective problem, so that the probability of edges of the best solution to be traversed by ants in next iteration of algorithm increases as probability depends on pheromone value also.

4.3 Non-Dominated Sorting

We have used concept of non-dominance. In [13], a sorting technique is proposed to sort a population into non-dominated fronts, the first non-dominated front is found and removed from the population, then the second is found from the remaining members and removed, then the third, and so on, until every member of the population has been assigned to its proper front.

For a population of individuals with M = 2 objectives, the individual with the lowest first-objective score must be part of the NDF, since it cannot be dominated by any other individual. If two or more individuals tie for the lowest first-objective score, then these solutions must also be compared in the second objective, and the individual(s) which score(s) best in the second objective are definitely in the population's first non-dominated front. The algorithm repeatedly applies this idea to efficiently find the population's NDF.

5 Experimental Results and Discussion

In order to assess the effectiveness of the proposed scheduling method, we have obtained the solutions for random task graph. Random task graph were generated using the method as proposed in [2]. There are few parameters involved in the random task graph generation were set with the following values:

- $SET_N = \{20, 40, 60\}$,
- $SET_{CCR} = \{0.1, 1\}$,
- $SET_{\alpha} = \{0.2, 0.4\}$,
- $SET_{out\_degree} = \{1, 2, 3, 4, n\}$,
- $SET_{\beta} = \{0.1, 0.5, 1.0\}$

where N is number of nodes (tasks) in the graph, α is shape parameter of the graph. CCR is the ratio of the average communication cost to the average computation cost of the application DAG. We have also generated randomly a set of processors, where λ is chosen uniformly in the range $[10^{-3}, 10^{-4}]$.

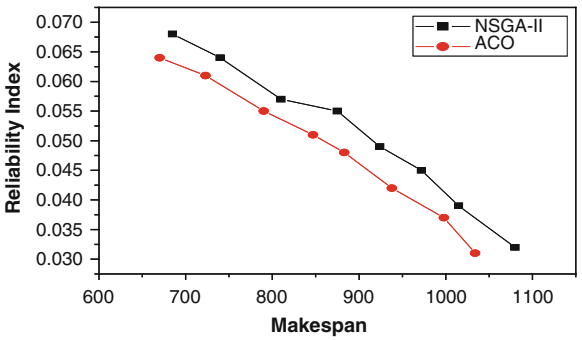

Fig. 1 Obtained Pareto solutions for 20 numbers of tasks

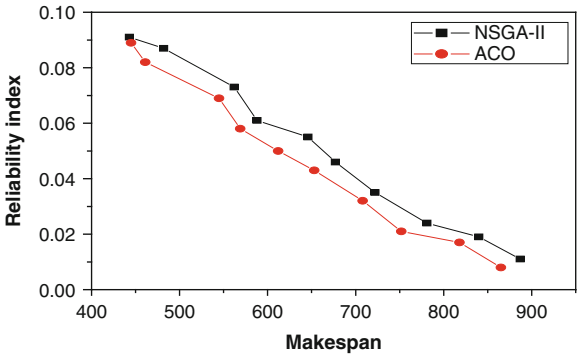

Fig. 2 Obtained Pareto solutions for 40 numbers of tasks

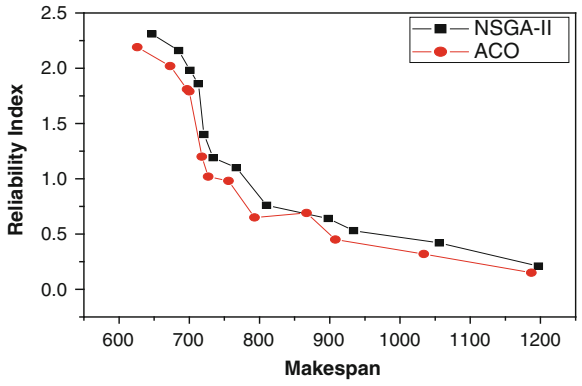

Fig. 3 Obtained Pareto solutions for 60 numbers of tasks

The obtained global Pareto solutions using multi-objective ACO and NSGA-II are as shown in Figs. 1, 2 and 3. The results show that the failure probability of the random task graph increases in proportion to the size of the application. This is due to the fact that when the size of an application increases, processors have to be failure-free for longer periods of time.

Table 1 Metrics for evaluating the diversity and convergence of ACO and NSGA-II

No. of tasks	Spacing		GD	
	ACO	NSGA-II	ACO	NSGA-II
20	1.15	1.19	0.25	0.28
40	0.86	0.94	0.43	0.49
60	0.46	0.49	0.54	0.57

There are many performance metrics proposed in the literature. One of the performance metrics namely, *spacing* [14, 15] is used to measure the diversity among obtained non-dominated solutions. The *Generational Distance* Metric (GD) [14, 15] used for measuring the convergence of the obtained non-dominated solutions. The spacing and GD values for the random task graphs are given in Table 1. The values confirm that ACO is better for the problem under study.

6 Conclusion

Scheduling is a critical issue for the execution of performance driven Grid applications. The work presented in the paper focuses on an efficient algorithm for multi-objective Grid Scheduling by assigning the submitted jobs to appropriate resources. In multi-objective optimization problem, multiple trade-off pareto solutions are produced for the maximum satisfaction of user. In this work, we proposed the use of Ant Colony Optimization algorithm using concept of non-dominance to solve bi-objective workflow scheduling problems. In our scheduling problem, we have considered the two major objectives: minimization of makespan (execution time) and maximization of reliability (to incorporate failure affect of resources in scheduling decision). But we have formulated the reliability in terms of minimization of reliability index.

The pareto solutions obtained by multi-objective ACO are compared with the solutions obtained by NSGA-II and a statistical analysis of their results has been presented to show the quality of each algorithm on different number of tasks. To measure the quality of these obtained solutions, we selected two metrics called GD (Convergence metric) and spacing (Diversity metric). A statistical analysis showed that multi-objective ACO outperforms NSGA-II in terms of both convergence towards pareto optimal front and maintaining good spread between solutions.

References

1. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid. Int. J. Supercomput. Appl. **15**(3) (2001)
2. Topcuoglu, H., Hariri, S., Wu, M.Y.: Performance-effective and low complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. **13**(3), 260–274 (2002)

3. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. **1**(1), 53–66 (1997)
4. Salari, E., Eshghi, K.: An ACO algorithm for graph coloring problem. In: Conference on Computational Intelligence Methods and Applications, December 2005
5. Zhang, X., Tang, L.: CT-ACO-hybridizing ant colony optimization with cycle transfer search for the vehicle routing problem. In: Conference on Computational Intelligence Methods and Applications, December 2005
6. Maheswaran, M., Ali, S., Siegel, H.J., Hensgen, D., Freund, R.: Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing system. J. Parallel Distrib. Comput. **59**, 107–131 (1999)
7. Xhafa, F., Abraham, A.: Computational models and heuristic methods for grid scheduling problems. J. Future Gener. Comput. Syst. **26**, 608–621 (2010)
8. Ye, G., Rao, R., Li, M.: A multiobjective resources scheduling approach based on genetic algorithms in grid environment. In: 5th International Conference on Grid and Cooperative Computing Workshops. pp. 504–509 (2006)
9. Dai, Y.S., Levitin, G.: Performance and reliability of tree structured grid services considering data dependence and failure correlation. IEEE Trans. Comput. **56**(7), 925–936 (2007)
10. Sallim, J., Shahrir, W.M., Hussin, W.: A background study on ant colony optimization metaheuristic and its application principles in resolving three combinatorial optimization problems. In: National Conference on Software Engineering and Computer Systems, Legend Resort Kuantan (2007)
11. Tang, X., Li, K., Li, R., Veeravalli, B.: Reliability-aware scheduling strategy for heterogeneous distributed computing systems. J. Parallel Distrib. Comput. **70**, 941–952 (2010)
12. Liu, G.Q., Poh, K.L., Xie, M.: Iterative list scheduling for heterogeneous computing. J. Parallel Distrib. Comput. **65**(5), 654–665 (2005)
13. Mazurek, M., Wesolkowski, S.: Non-dominated sorting on two objectives. Defence R&D Canada—CORA, Technical Note 027, pp. 1–13, July 2009
14. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. Wiley, New York (2001). ISBN: 0-471-87339-X
15. Deb, K., Jain, S.: Running performance metrics for evolutionary multi-objective optimization. In: Simulated Evolution and Learning, pp. 13–20 (2002)