MULTIOBJECTIVE EVOLUTIONARY ALGORITHMS FOR SCHEDULING JOBS ON COMPUTATIONAL GRIDS

Crina Grosan, Ajith Abraham and Bjarne Helvik
Centre for Quantifiable Quality of Service in Communication Systems,
Norvegian University of Science and Technology, Trondheim, Norway
{crina,ajith,bjarne}@ q2s.ntnu.no

Abstract. In a computational grid, at time *t*, the task is to allocate the user defined jobs efficiently by meeting the deadlines and making use of all the available resources. In the past, objectives were combined and the problem is very often simplified to a single objective problem. In this paper, we formulate a novel Evolutionary Multi-Objective (EMO) approach by using the Pareto dominance and the objectives are formulated independently. We report some preliminary experiments and the performance of the EMO approach is compared with simulated annealing and particle swarm optimization techniques. Empirical results indicate that the proposed EMO approach is very efficient.

1. INTRODUCTION

Grid Computing (GC) is the ultimate framework to meet the growing computational demands in the new millennium [1][3]. To meet the growing needs of the computational power, geographically distributed resources need to be logically coupled together to make them work as a unified resource. Computing resources are geographically distributed under different ownerships each having their own access policy, cost and various constraints. Every resource owners will have a unique way of managing and scheduling resources and the grid schedulers are to ensure that they do not conflict with resource owner's policies. In the worst-case situation, the resource owners might charge different prices to different grid users for their resource usage and it might vary from time to time. Traditionally, most of the schedulers followed system centric approach in resource selection and often completely ignore the user requirements.

Job scheduling is known to be NP-complete, therefore the use of heuristics is the *defacto* approach in order to cope in practice with its difficulty. Single heuristic approaches for the problem include Local Search [6], Simulated Annealing [5][7] and Tabu Search [5]. Genetic algorithms for grid scheduling are addressed in several works [5] [8]. Ritchie and Levine [9] combined an Ant Colony Optimization algorithm with a Tabu search algorithm for the problem. Ye et al. [4] formulated a multi-objective optimization approach to simultaneously optimize the completion time and the total execution cost. Other approaches for the problem include the use of AI techniques [10], Particle Swarm Optimization [1] Fuzzy based scheduling [11] and economic-based approaches [2]. In an economic-based approach an optimal schedule often relies on a trade off between cost and the user specified deadline.

Recently there has been a growing interest in Multi-Objective Evolutionary Algorithm (MOEA) algorithms which combines two major disciplines: evolutionary computation and the theoretical frameworks of multi-criteria decision making. Even though some real world problems can be reduced to a matter of single objective very often it is hard to define all the aspects in terms of a single objective. Defining multiple objectives often gives a better idea of the task. In Section 2, we introduce the grid scheduling problem and in Section 3 we present the proposed multiobjective approach for the problem. In Section 4, we demonstrate how the MOEA approach can be used to formulate scheduling independent tasks in a grid environment followed by some conclusions.

2. GRID RESOURCE MANAGEMENT AND SCHEDULING ISSUES

The grid resource broker is responsible for resource discovery, deciding allocation of a job to a particular resource, binding of user applications (files), hardware resources, initiate computations, adapt to the changes in grid resources and present the grid to the user as a single, unified resource. It finally controls the physical allocation of the tasks and manages the available resources constantly while dynamically updating the grid scheduler whenever there is a change in resource availability. When the computing power demand is much greater than the available resources only dynamic scheduling will be useful. To conceptualize the problem as an algorithm, we need to dynamically estimate the job lengths from user application specifications or historical data. We assume that the jobs and resources are arranged in an ascending order (and dynamically updated) according to the job lengths and processor speeds. The information related to job lengths may be derived from historical data, some kind of strategy defined by the user or through load profiling.

To formulate the problem, we consider J_n independent user jobs n={1,2,….N} on R_m heterogeneous resources m={1,2,….,M} with an objective of minimizing the completion time and utilizing the resources effectively. The speed of each resource is expressed in number of cycles per unit time, and the length of each job in number of cycles. Each job J_n has processing requirement P_j cycles and resource R_m has speed of S_i cycles/second. Any job J_n has to be processed in resource R_m, until completion.

To formulate our objective, define C_j as the completion time the last job j finishes processing. Define C_{max} = max {C_j , j=1,…,N}, the makespan and ΣC_j , as the flowtime. An optimal schedule will be the one that optimizes the flowtime and makespan [8]. The conceptually obvious rule to minimize ΣC_j is to schedule Shortest Job on the Fastest Resource (SJFR). The simplest rule to minimize C_{max} is to schedule the Longest Job on the Fastest Resource (LJFR). Minimizing ΣC_j asks the average job finishes quickly, at the expense of the largest job taking a long time, whereas minimizing C_{max}, asks that no job takes too long, at the expense of most jobs taking a long time. In summary, minimization of C_{max} will result in maximization of ΣC_j.

3. EVOLUTIONARY MULTIOBJECTIVE APPROACH

Several optimization criteria can be considered for this problem, but in this paper, we consider a bi-objective minimization problem with the task of minimization of *makespan* and *flowtime*. The fundamental criterion is that of minimizing the *makespan*, that is, the time when the last task is finished. A secondary criterion is to minimize the *flowtime* of the grid system that is, minimizing the sum of completion times of all the tasks. The most common approaches of a multiobjective optimization problem use the concept of Pareto dominance as defined below:

Definition (*Pareto dominance*)

Consider a maximization problem. Let x, y be two decision vectors (solutions) from the definition domain.

Solution x *dominate* y (also written as $x \succ y$) if and only if the following conditions are fulfilled:

(i) $f_i(x) \geq f_i(y)$; $\forall i = 1, 2, \ldots, n$;

(ii) $\exists j \in \{1, 2, \ldots, n\} : f_j(x) > f_j(y)$.

That is, a feasible vector x is Pareto optimal if no feasible vector y can increase some criterion without causing a simultaneous decrease in at least one other criterion. Multiobjective evolutionary algorithms can yield a whole set of potential solutions, which are all optimal in some sense. The main challenge in a multiobjective optimization environment is to minimize the distance of the generated solutions to the Pareto set and to maximize the diversity of the developed Pareto set. A good Pareto set may be obtained by appropriate guiding of the search process through careful design of reproduction operators and fitness assignment strategies. To obtain diversification special care has to be taken in the selection process. Special care is also to be taken care to prevent non-dominated solutions from being lost.

3.1. Solution representation and genetic operators

In the proposed multiobjective approach, the solution is represented as a string of length equal to the number of jobs. The value corresponding to each position *i* in the string represent the machine to which job *i* was allocated. Consider we have 10 jobs and 3 machines. Then a chromosome and the job allocation can be represented as follows:

1	2	3	2	1	1	3	2	1	3

Machine 1	Job 1	Job 5	Job 6	Job 9
Machine 2	Job 2	Job 4	Job 8	
Machine 3	Job 3	Job 7	Job 10	

Mutation and crossover were used as operators. Binary tournament selection was used in the implementation. The Pareto dominance concept is used in order to compare 2 solutions. The one which dominates is preferred. In case of nondominance, the solution whose jobs allocation between machines is uniform is preferred. This means, there will not be idle machines as well as overloaded machines. The evolution process is similar to the evolution scheme of a standard genetic algorithm for multiobjective optimization. Reader is advised to refer [12] for more details about EMO approach.

4. EXPERIMENT RESULTS

As a preliminary study, two sets of scheduling experiments were performed. Results obtained by MOEA are compared with a simple Genetic algorithm (GA), Simulated Annealing (SA) and Particle Swarm Optimization (PSO). Specific parameter settings for all the considered algorithms are described in Table 1. Each experiment was repeated 10 times with different random seeds. Each trial (except for MOEA) had a fixed number of 50 $*m*n$ iterations (m is the number of the grid nodes, n is the number of the jobs). The makespan values of the best solutions throughout the optimization run were recorded. In a grid environment, the main emphasis was to generate the schedules as fast as possible. So the completion time for 10 trials was used as one of the criteria to improve their performance. First we tested a small scale job scheduling problem involving 3 nodes and 13 jobs represented as (3,13). The node speeds of the 3 nodes are 4, 3, 2 CPUT, and the job length of 13 jobs are 6,12,16,20,24,28,30,36,40,42,48,52,60 cycles, respectively. The results (makespan) for 10 runs were as follows:

- GA: {47, 46, 47, 47.3333, 46, 47, 47, 47, 47.3333, 49}, average value = 47.1167.
- SA: {46.5, 46.5, 46, 46, 46, 46.6667, 47, 47.3333, 47, 47} average value = 46.6.
- PSO : {46, 46, 46, 46, 46.5, 46.5, 46.5, 46, 46.5, 46.6667}, average value = 46.2667.
- **MOEA: {46, 46, 46, 46, 46, 46, 46, 46, 46, 46}, average value = 46.**

The optimal result for (3,13) makespan is supposed to be 46 and the MOEA approach gave 46. The average total flow time obtained = 138. While GA provided the best results twice, SA and PSO provided the best result three and five times respectively. MOEA approach is obtaining the best result in each of the considered runs.

Table 1. Parameters used by the algorithms considered in experiments

Algorithm	Parameter	Value
GA	Population size Crossover probability Mutation probability Scale for mutations	20 0.8 0.02 0.1
SA	Number operations before temperature adjustment Number of cycles SA temperature reduction factor Vector for control step of length adjustment Initial temperature	20 10 0.85 2 50
PSO	Swarm size PSO Self-recognition coefficient $c1$ 1.49 Social coefficient $c2$ 1.49 Inertia weight w	20 1.49 1.49 $0.9 \rightarrow 0.1$
MOEA	Population size Number of generations Mutation probability Crossover probability	100 (500 for the second experiment) 200 (1000 for the second experiment) 1 (0.9 for the second experiment) 1 (0.9 for the second experiment)

Further, we tested the algorithms for the case (10, 50). All the jobs and the nodes were submitted at one time. The average makespan values for 10 trials are illustrated in Table 2. Although the average makespan value of SA was better than that of GA for (3,13), the case was reversed for this second case. Using the MOEA approach, the total average flow time obtained is = 348.07. Figures 1 (a) and (b) illustrate the makespan and flow time given by 31 non-dominated solutions from the final population. The user would have the option to go for a better flow time solution at the expense of a non-optimal makespan. As evident from the Figure, the lowest flow time was 343.72 with the makespan of 44.75 for solution no. 27.

Table 2. Performance comparison for the case (10, 50).

Algorithm	Average makespan
GA	38.04
SA	41.78
PSO	37.66
MOEA	**36.68**

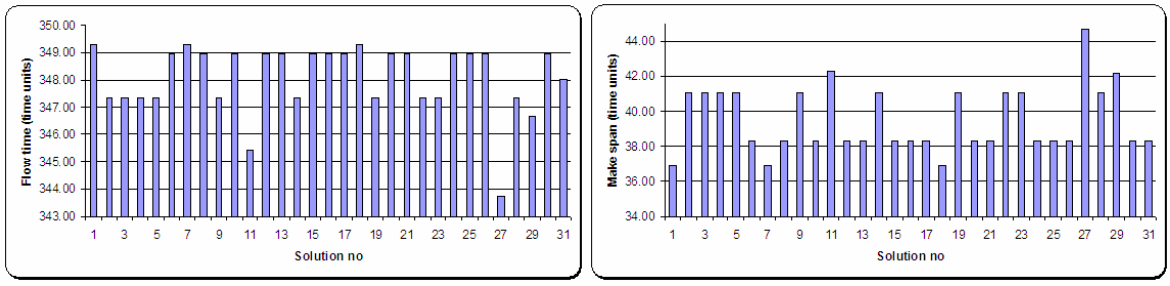

Figure 1. (a) Makespan and **(b)** Flowtime from 31 non-dominated solutions in the final population for (10, 50)

Results obtained by GA, SA and PSO were adapted from [1]. As evident from the data obtained above, MOEA have given excellent results when compared to other techniques modeled using a single objective approach. Due to space limitations, more results (also Pareto fronts etc.) could not be presented in this paper.

5. CONCLUSIONS

The grid scheduling problem involves simultaneous optimization of several objectives including completion time, resource utilization, QoS metrics, costs, reliability factors etc. So, by its nature, it is a multiobjective optimization problem. All the existing approaches for dealing with grid scheduling problem transform this problem into a single objective problem (either by using weighted sum method or other similar methods). In this paper a new multiobjective approach for grid scheduling problem is proposed. The results obtained by EMOA are compared with the results obtained by three well known global optimization techniques namely simulated annealing, genetic algorithm and particle swarm optimization. Even though the MOEA approach obtained better results for the considered test problems, more conclusions could be drawn only after extensive validation using bigger problem sizes and more objectives etc. Our future research plan is to extend the approach for more complicated experiments with more objectives.

References

[1] Abraham A, Liu H, Zhang W, Chang TG, Scheduling Jobs on Computational Grids Using Fuzzy Particle Swarm Algorithm, Proceedings of 10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems, England, pp. 500-507, 2006.

[2] Buyya R, Abramson D, Giddy J, Grid Resource Management, Scheduling, and Computational Economy, International Workshop on Global and Cluster Computing, Japan, 2000.

[3] Foster I, Kesselmann C, (Eds.), The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, USA, 1999.

[4] Guangchang Ye, Ruonan Rao, Minglu Li, A Multiobjective Resources Scheduling Approach Based on Genetic Algorithms in Grid Environment, Fifth International Conference on Grid and Cooperative Computing Workshops, pp. 504-509, 2006.

[5] Abraham A., Buyya R. and Nath B., Nature's Heuristics for Scheduling Jobs in Computational Grids, In Proceedings of 8th IEEE International Conference on Advanced Computing and Communications, (ADCOM2000), Sinha P.S. and Gupta R. (Editors), ISBN 0070435480, Tata McGraw-Hill Publishing Co. Ltd, New Delhi, pp. 45-52, 2000.

[6] G. Ritchie and J. Levine. A fast, effective local search for scheduling independent jobs in heterogeneous computing environments. Technical report, Centre for Intelligent Systems and their Applications, School of Informatics, University of Edinburgh, 2003.

[7] A. Yarkhan and J. Dongarra. Experiments with scheduling using simulated annealing in a grid environment. In *3rd International Workshop on Grid Computing (GRID2002)*, 232{242, 2002.

[8] A.Y. Zomaya and Y.H. Teh. Observations on using genetic algorithms for dynamic load-balancing, *IEEE Transactions On Parallel and Distributed Systems*, 12(9):899{911, 2001.

[9] G. Ritchie and J. Levine. A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments. In *23rd Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG 2004)*, 2004.

[10] J. Cao, D.P. Spooner, S.A. Jarvis and G.R. Nudd, Grid load balancing using intelligent agents, *Future Generation Computer Systems*, 21(1), 135-149, 2005.

[11] K.P. Kumar, A. Agarwal, and R. Krishnan. Fuzzy based resource management framework for high throughput computing. In Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid, 555-562, 2004.

[12] K. Deb, Multiobjective Optimization using Evolutionary Algorithms, John Wiley & Sons, UK, 2001.