STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIV**, Number 1, 2009

A MULTIOBJECTIVE METAHEURISTIC FOR JOB-SHOP SCHEDULING

CRINA GROŞAN

Abstract. In this paper, we introduce a nature inspired meta-heuristic for scheduling jobs on computational grids. Our approach is to dynamically generate an optimal schedule so as to complete the tasks in a minimum period of time as well as utilizing the resources in an efficient way. The approach proposed is a variant of particle swarm optimization which uses mutation operator. The mutation operator can affect both particle's personal best and the swarm's global best. The experiments performed show the efficiency of the proposed approach over the standard PSO and other metaheuristics considered (namely genetic algorithms and simulated annealing).

1. Introduction

Grid Computing (GC) is the ultimate framework to meet the growing computational demands in the new millennium [2], [3], [4], [5], [6], [7]. 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 a grid environment knowing the processing speeds of the available resources and the job length of user applications is a tedious task. Usually it is easy to get information about the speed of the available resources but quite complicated to know the computational processing time requirements from the user. 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. Soft computing techniques like fuzzy logic, evolutionary computation and artificial neural networks

Received by the editors: April 3, 2009.

might be of useful aid in the parameters estimation process especially in times of uncertainty and vague data.

The paper is organized as follows. Section 2 deals with some theoretical foundations related to job scheduling. A variation of the particle swarm optimization heuristic is introduced in Section 3. In Section 4, experiment results and discussions are provided. Finally, we conclude our work.

2. Scheduling Problem Formulation

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 $\sum C_j$ as the flowtime.

An optimal schedule will be the one that optimizes the flowtime and makespan [8]. The conceptually obvious rule to minimize $\sum C_j$ 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 $\sum C_j$ 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 $\sum C_j$.

Several optimization criteria can be considered for this problem, certainly the problem is multiobjective. The fundamental criterion is that of minimizing the makespan, that is, the time when finishes the latest task. A secondary criterion is to minimize the flowtime of the grid system that is, minimizing the sum of finalization times of all the tasks:

- minimization of makespan;
- minimization of flowtime.

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 if and only if the following conditions are fulfilled:

(i) $f_i(x) \geq f_i(y); i = 1, 2, ..., n$;
(ii) $\exists j \in 1, 2, ..., n : f_j(x) > f_j(y)$.

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.

3. Proposed Particle Swarm Model for Job Scheduling (PSJS)

The classical particle swarm model consists of a swarm of particles, which are initialized with a population of random candidate solutions. They move iteratively through the d-dimension problem space to search the new solutions, where the fitness, f, can be calculated as the certain qualities measure.

For a d-dimensional search space the position of the i-th particle is represented as: $X_i = (x_{i1}, ..., x_{id})$.

Each particle maintains a memory of its previous best position $P_{best\_i} = (p_{i1}, p_{i2}, ..., p_{id})$.

The best one among all the particles in the population is represented as $P_{gbest} = (p_{g1}, p_{g2}, ..., p_{gd})$.

The velocity of each particle is represented as: $V_i = (v_{i1}, v_{i2}, ..., vid)$.

In each iteration, the P vector of the particle with best fitness in the local neighborhood, designated g, and the P vector of the current particle are combined to adjust the velocity along each dimension and a new position of the particle is determined using that velocity. Equations of velocity vector and position vector given by:

$$v_{id} = w \cdot v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \tag{1}$$

$$x_{id} = x_{id} + v_{id}. \tag{2}$$

First part of equation 1 represents the inertia of the previous velocity, second part is the cognition part, third part represents the cooperation among particles and is called social component. Acceleration constants c_1, c_2 and inertia weight w are the predefined by the user. r_1, r_2 are the uniformly generated random numbers between 0 and 1.

Even tough PSO is known as a good optimization technique, the method still lacks in several aspects and require further improvements in order to performed a better convergence. In this paper we propose the usage of mutation

which can affect both the global best particle as well as the best known position of each particle.

The personal best position of each particle is mutated at the end of each iteration while the global best position is mutated only after a couple of iterations (this is a parameter of the algorithm).

4. Experiments

We performed two experiments. Results obtained by PSJS are compared to the ones obtained by Genetic Algorithm (GA), Simulated Annealing (SA) and the standard multiobjective Particle Swarm Optimization (PSO). We should mention that in the case of SA and GA the objectives are aggregated using weighted sum method while in the case of both particle swarm approaches we used Pareto dominance.

Specific parameter settings of all the considered algorithms are described in Table 1.

Each experiment (for each algorithm except for the MOEA) was repeated 10 times with different random seeds. Each trial (except for MOEA) had a fixed number of $50 \cdot m \cdot n$ 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, with an average value of 47.1167.
- SA: 46.5, 46.5, 46, 46, 46, 46.6667, 47, 47.3333, 47, 47 with an average value of 46.6.
- PSO : 46, 46, 46, 46, 46.5, 46.5, 46.5, 46, 46.5, 46.6667,with an average value of 46.2667.
- PSJS: 46, 46, 46, 46, 46, 46, 46, 46, 46, 46, with an average value of 46.

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

TABLE 1. Parameters used by the algorithms considered in experiments

Algorithm	Parameter	Value
GA	Population size	20
	Crossover probability	0.8
	Mutation probability	0.02
	Scale for mutations	0.1
SA	Number operations before temperature adjustment	20
	Number of cycles	10
	SA Temperature reduction factor	0.85
	Vector for control step of length adjustment	2
	Initial temperature	50
PSO	Swarm size	20
	PSO Self-recognition coefficient c_1	1.49
	Social coefficient c_2	1.49
	Inertia weight w	$0.9 \rightarrow 0.1$
PSJS	Swarm size	20
	PSO Self-recognition coefficient c_1	1.49
	Social coefficient c_2	1.49
	Inertia weight w	$0.9 \rightarrow 0.1$
	Number of iterations for a global best mutation	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.

Results obtained by GA, SA and PSO are taken from [1]. As evident from the data obtained above, MOEA results clearly outperform results obtained by the other techniques considering a single objective approach.

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

Algorithm	Average makespan
GA	38.04
SA	41.78
PSO	37.66
PSJS	36.7

5. CONSLUSIONS

In this paper a new variant of multiobjective PSO which uses mutation operator affection both personal best of each particle and the swarm's global

best is proposed for the scheduling problem in grid computing. Based on the experimental results presented we can conclude that PSJS performs better than standard PSO and it also obtains better results than the approaches which aggregate the objective functions and consider the problem as single objective instead of using Pareto dominance.

Acknowledgement

The author acknowledge the support from the CNCSIS Grant IDEI 2412/2008.

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 and Intelligent Information and Engineering Systems, England, pp. 500-507, 2006.

[2] Abraham A, Liu H, Grosan C, Xhafa F, Nature Inspired Metaheuristics for Grid Scheduling: Single and Multiobjective Optimization Approaches, Metaheuristics for Scheduling: Distributed Computing Environments, Studies in Computational Intelligence, Springer Verlag, Germany, ISBN: 978-3-540-69260-7, pp. 247-272, 2008.

[3] Baker M, Buyya R, Laforenza D, The Grid: International Efforts in Global Computing, International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, Rome, Italy, 2000.

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

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

[6] Liu H, Abraham A, An Hybrid Fuzzy Variable Neighborhood Particle Swarm Optimization Algorithm for Solving Quadratic Assignment Problems, Journal of Universal Computer Science, Volume 13, No. 7, pp. 1032-1054, 2007.

[7] Pant M, Thangaraj R, Abraham A, Particle Swarm Optimization Using Adaptive Mutation, 2nd International Workshop on Evolutionary Techniques in Data-processing (DEXA'08/ETID '08), IEEE Computer Society Press, USA, ISBN 978-0-7695-3299-8, pp. 519-523, 2008.

Department of Computer Science, Babes-Bolyai University, Kogalniceanu 1, Cluj-Napoca 400084, Romania

E-mail address: cgrosan@cs.ubbcluj.ro