Scheduling Jobs in Computational Grid using Hybrid ACS and GA Approach

Mustafa Muwafak Alobaedy
School of Computing
College of Art and Sciences
University Utara Malaysia, 06010 Sintok, Kedah
new.technology@hotmail.com

Ku Ruhana Ku-Mahamud
School of Computing
College of Art and Sciences
University Utara Malaysia, 06010 Sintok, Kedah
ruhana@uum.edu.my

***Abstract*—Metaheuristics algorithms show very good performance in solving various job scheduling problems in computational grid systems. However, due to the complexity and heterogeneous nature of resources in grid computing, stand-alone algorithm is not capable to find a good quality solution in reasonable time. This study proposes a hybrid algorithm, specifically ant colony system and genetic algorithm to solve the job scheduling problem. The high level hybridization algorithm will keep the identity of each algorithm in performing the scheduling task. The study focuses on static grid computing environment and the metrics for optimization are the makespan and flowtime. Experiment results show that the proposed algorithm outperforms other stand-alone algorithms such as ant system, genetic algorithms, and ant colony system for makespan. However, for flowtime, ant system and genetic algorithm perform better.**

Keywords— job scheduling; hybrid Ant Colony System; Genetic Algorithm; static grid computing.

I. INTRODUCTION

Computational grid is one of the main services provided by grid systems. Grid is defined as "Geographically distributed computers, linked through the internet in a Grid-like manner, are used to create virtual supercomputers of vast amount of computing capacity able to solve complex problem from e-Science in less time than known before" [1]. Grid systems evolve from existing technology such as distributed computing, web service, and Internet [2]. Grid systems are classified as modern High Performance Distributed Systems (HPDSs) along with the clusters and cloud systems [3]. However, there are crucial characteristics which differ between them such as scale, network type, administrative domain, resources structure, and security [4]. There are many different types of grid systems such as sensor grid, campus grid, global grid, pc grid, and utility grid [3], [5]. Grid computing system has been utilized in various fields such as scientific, education, and commercial fields [6], [7].

One of the main components in grid computing systems is resource management system which consists of grid information server, domain resource manager, and resource scheduler [8]. The scheduler has the main influence in grid computing performance [9]. The scheduler's responsibility is to map the submitted jobs from users to the suitable and available resources. The efficiency of the scheduler depends on the implemented algorithm. Scheduling could be done using simple algorithms such as greedy or random approach. However, using more sophisticated algorithms will enhance the scheduler's efficiency, which in turn will enhance the grid performance in general.

Scheduling jobs in grid computing are known as NP-complete problem due to the problem complexity and intractable nature of the problem [10]. Such a problem could be solved using metaheuristic algorithms. These types of algorithms have the ability to find near optimal solution in reasonable time rather than optimal solution in a very long processing time [11]. Metaheuristic algorithms such as Tabu Search (TS), Genetic Algorithm (GA), and Ant Colony Optimization (ACO) show very promising performance to solve various types of scheduling problems [12]. However, hybridizing two or more algorithms show better performance than applying a stand-alone algorithm [3]. This is due to the ability of hybrid approach to skip from local minima using more options available in the algorithms used in the hybridization. Hybrid approaches between ACO and GA have been studied in [13], [14]. However, these hybridized approaches are different from the proposed hybridized approach in this study. The ant system (AS) which is a variant of ACO has been used in [13] and [14] to solve university class scheduling and robot path planning. In this study, the ant colony system (ACS) which is another variant of ACO is used to solve job scheduling in static grid computing environment.

The rest of the paper is organized as follows. Section II presents the research on ant colony optimization and genetic algorithm in solving NP hard problems. The implementation of ACS and GA in grid computing is described in Section III. Section IV briefly explains the problem formulation and the benchmark for static grid scheduling. Section V presents the results of ACS hybrid with GA in grid computing. Finally, the conclusion is provided in Section VI.

II. METAHEURISTICS ALGORITHM FOR NP PROBLEMS

In computational grid systems, scheduler is an important component for resource management. Scheduler algorithm has the responsibility to schedule jobs efficiently [9]. Job scheduling is known as NP-complete problem which needs metaheuristics algorithms to be solved. One of the best metaheuristics algorithms in the field of optimization is ACO.

978-1-4799-4811-6/14/$31.00 ©2014 IEEE

ACO is considered as a swarm intelligence algorithm which mimics the behaviours of real biological ants. ACO is implemented to solve many problems such as routing, scheduling, and classification [15]. Many studies have implemented and enhanced ACO for job scheduling in grid computing. An ACO approach for job scheduling in grid system by [16] proposed two types of ants, namely the red and black ants for the purpose of sharing the search load. The performance of this algorithm was compared with Min-Min algorithm presented in [17] and first come first serve. Experimental results show that this algorithm outperforms the other two algorithms.

A study presented by [18] proposed a Balanced ACO (BACO) algorithm for job scheduling in grid. The proposed algorithm is based on the basic ideas from ACO algorithm. Each ant in the system represents a job in the grid systems. In addition, the pheromone value represents the weight for a resource in the grid system. Higher weight means that the resource has a better computing capability. The study also considered the bandwidth speed available between the scheduler and resource. This algorithm has been implemented in the Taiwan UniGrid which consists of more than 20 campuses. The experimental results show that BACO algorithm outperforms the improved ACO [19], fastest processor to largest task first [20], and Suffrage [21].

A hybrid ACO approach (HACO) for job scheduling in grid computing proposed in [22] has integrated the heuristic information to make the algorithm converge faster to the solution. The experiments used the benchmark model known as Expected Time to Compute (ETC) model presented in [23]. The performance of HACO was compared with ACO in terms of makespan criterion. Empirical results show that HACO outperforms the existing ACO algorithm. A successful variant of ACO algorithm for job scheduling in computational grid [24], [25] is the Ant Colony System (ACS) by Dorigo [26]. ACS algorithm enhances ant system in three phases: first, the exploration mechanism became stronger due to the implementation of aggressive rule. Second, only the ant who found the best solution is allowed to deposit the pheromone trail to the arcs which belong to that solution. Third, evaporation process will be applied only to the arcs used by ants to increase the exploration of alternative arcs [27].

Besides ACO-based algorithm, there are many other algorithms that have been successfully applied to solve optimization problems. One of these algorithms is GA which is a metaheuristic algorithm that imitates the principle of genetic process in living organisms. GA mimics the evolutionary process by applying selection, recombination, and mutation to generate solution from the search space. Genetic algorithm is a well-known algorithm to solve various types of combinatorial optimization problems. Enhanced Genetic-based scheduling for grid computing is proposed in [28]. The authors presented an implementation of Hierarchic Genetic Strategy (HGS) for job scheduling in dynamic computational grid environment. HGS has the ability to search the solution space concurrently using various evolutionary processes. The study focused on bi-objective optimization specifically, makespan and flowtime simultaneously have been optimized. Experiments were conducted under heterogeneous, large scale, and dynamic environment using grid simulator. HGS was tested with static and dynamic grid computing environment. The experiment with static environment is based on ETC matrix model presented by [29] and for dynamic environment, the authors used a simulator presented by [30]. HGS was also compared with two other GA-based schedulers presented in [23], [31]. The results show that HGS outperforms the other GA-based schedulers. It is not known how HGS will perform against other metaheuristic algorithms since only GA-based algorithm was used for comparison.

A study presented by [32] proposed a hybrid approach between GA and TS for independent batch job scheduling in grid computing. The hybrid algorithm aims to optimize the makespan and flowtime as a bio-objectives scheduling problem. In addition, the authors proposed hierarchical and simultaneous approaches for optimizing makespan and flowtime. Two types of hybridization were provided, namely low and high level hybridization which are known as GA(TS) and GA+TS algorithms. The experiments conducted have considered static and dynamic grid computing environment using HyperSim-G simulator developed by [33]. The proposed algorithms were compared with GA presented by [31] and TS presented by [34]. Experimental results show that the proposed hybrid algorithms outperform the other stand-alone algorithms in terms of makespan criterion. However, in terms of flowtime criterion, GA and TS stand-alone algorithms outperform the proposed hybrid algorithm. Such a contradiction is normal for job scheduling in grid computing. In spite of the limitation on the experiments and benchmarking problem, the study has clearly illustrated the implementation of the hybrid algorithms.

III. Proposed ACS+GA for Job Scheduling

Hybridization is a term which refers to the approach that combines two or more algorithms in order to achieve a result which is not achievable using a stand-alone approach [35]. Algorithms could be hybridized fully or partially to be able to get the best features of the combined algorithms. There are two levels of hybridization between algorithms namely, high level and low level. In high level, which is also called loosely coupled hybridization, each algorithm preserves its identity. In other words, each algorithm operates fully in the hybridized approach. This type of hybridization can be seen as a chain of algorithm execution ($Algorithm_1 \rightarrow Algorithm_2 \rightarrow \cdots \rightarrow Algorithm_n$). This execution can be further looped in a certain number of iterations until the termination condition is satisfied. Through the algorithm execution, the output solution is passed from $Algorithm_1$ to $Algorithm_2$ and so on. In low level hybridization, also known as strongly coupled, the algorithms interchange their inner procedures. The level of hybridization reflects the degree of inner exchange among the hybridized algorithms. In low level hybridization, one of the algorithms is the main algorithm, which calls other algorithms at any time of execution (depending on the hybridization design). The low level hybridization algorithm could be presented as $Algorithm_1(Algorithm_2)$. In this representation, $Algorithm_1$ is the main algorithm and $Algorithm_2$ is the subordinated algorithm [36], [37].

This study implemented a high level hybridization approach namely ACS + GA. ACS will start first for a specific

time, and after ACS finishes execution, GA will start to enhance the solution found by ACS. In other words, the solution found by ACS will be a part of the initial populations of GA.

For ACS implementation, the heuristic information needs to be defined. For static environment, heuristic η value is calculated from the ETC matrix using $\{1 / (\mathrm{ETC}_{ij} + \mathrm{Load}_j)\}$ where ETC_{ij} represents the expected time to compute task i on machine j, and Load_j is the previous load assigned to machine j [38]. Longer computing time and more loads will produce a smaller heuristic value, which will make the probability of selecting this machine smaller and vice versa. The probability of ant k to map task i to machine j is calculated by:

$$\mathrm{P}_{ij}^{\mathrm{Ant}k} = \begin{cases} \mathrm{argmax}\{[t_{ij}] \,.\, [\eta]^{\beta}\}, \text{if } q \leq q0; \\ J, \qquad \text{Otherwise;} \end{cases} \qquad Eq\ (1)$$

where t_{ij} is the pheromone value, η is the heuristic value, β is a parameter which determines the relative influence of the heuristic information, q is a random variable uniformly distributed between [0, 1], $q0$ ($0 \leq q0 \leq 1$) is a parameter which determines the exploration/exploitation rate, and J is a random variable selected according to the probability given by equation (2) with $\alpha = 1$ [27].

$$\mathrm{P}_{\mathrm{ij}}^{\mathrm{Antk}} = \frac{[\mathrm{t}_{ij}]^{\alpha} \,.\, [\eta)]^{\beta}}{\sum_{j=1}^{\mathrm{M}} [\mathrm{t}_{ij}]^{\alpha} \,.\, [\eta]^{\beta}} \qquad Eq\ (2)$$

For GA algorithm implementation, the output from ACS algorithm will be a part of the initial population of GA. The solution will be in the form of a vector. The index of each element represents the task number while the value of the vector element represents the machine number assigned to it. Therefore, the vector size is equal to the total number of tasks and the values in each element will be any value of non-negative integer number in the range of (0 to m-1) where m is the total number of machines in the grid. Figure 1 depicts the skeleton of the proposed algorithm.

```
Procedure ACS + GA
        Initialize parameters and pheromone trails;
        While (Termination condition not met) Do
                Construct new solution;
                Update pheromone trails;
        End while;
        Initialize population (P);
        Add(best-so-far solution from ACS to P);
        Evaluate (P);
        While (termination condition not met)
                Ṕ ← Recombine(P);
                Mutate(Ṕ);
                Evaluate(Ṕ);
                P ← Replace(Ṕ ∪ P);
        End While;
        Return the best solution;
End Procedure;
```

Fig. 1. ACS+GA skeleton.

IV. Problem Formulation

The problem in job scheduling for grid computing is known as a multi-objective problem due to the various criteria in computational grid such as makespan, flowtime, load balancing, utilization, matching proximity, turnaround time, total weighted completion time, and average weighted response time [39]. In this study, two criteria were implemented namely: makespan and flowtime with the priority to makespan as the main optimization objective. Makespan metric measures the general productivity of the grid computing. The best scheduling algorithm is the one that can produce a small value of makespan, which means that the algorithm is able to map tasks to machines in a good and efficient way. Therefore, the objective in this study is to minimize the makespan. Makespan is defined as the time when the last task finishes execution, formally defined as:

$$minimization\ of\ makespan: min\ S_{i \in Sched}\{max_{j \in Jobs} F_j\}$$

where $Sched$ is the set of all possible schedules, $Jobs$ is the set of all jobs to be scheduled, and F_j denotes the time when task j finalizes [39]. Flowtime is the second criteria used in this study which refers to the response time to the user submissions of task executions. Flowtime is defined as the sum of finalization times of all tasks, formally defined as:

$$flowtime: min\ S_{i \in Sched}\{\sum_{j \in Jobs} F_j\}.$$

These criteria could conflict with each other since limited resources could be the bottleneck of the system [39].

In order to test the proposed algorithm, a suitable benchmark is required to reflect the robustness of the algorithm. The benchmark should reflect the environment attributes such as resources and jobs heterogeneity. The considered benchmark for static grid computing is based on the successful model known as ETC to generate benchmarks on grid computing introduced by [23]. This model is widely accepted by researchers to be used for job scheduling in grid [23], [28], [40]. The benchmark defines a matrix called Expected Time to Compute. Each row in the $ETC\ [i, j]$ matrix contains the expected time to compute task $[i]$ on machine $[j]$. Therefore, ETC has $n * m$ entries where n represents the number of tasks and m represents the number of machines. ETC matrix is again defined using three metrics, namely task heterogeneity, machine heterogeneity, and consistency. The task heterogeneity measures the variance in execution time among tasks while machine heterogeneity measures the variance in machine speed among machines. The heterogeneity of tasks and machines is represented with two values of "high" and "low" respectively. In addition, ETC matrix captures other possible features of real heterogeneous computing system using three more metrics to measure the consistencies, namely consistent, inconsistent, and semi-consistent. The ETC matrix is considered consistent whenever a machine r_j executes a task t_i faster than another machine r_k, therefore, machine r_j will execute all other tasks faster than machine r_k. ETC matrix is considered inconsistent when a machine r_j could execute some tasks faster than machine r_k and some other slower. Finally, semi-consistent ETC matrix is an inconsistent matrix which has a consistent submatrix of specific size. Combining all these

matrices will generate 12 distinct types of possible ETC matrix [23].

V. EXPERIMENTS AND RESULTS

Metaheuristics algorithms such as ACS and GA have many parameters that need to be tuned. The values of the parameters need a lot of tuning in order to achieve the desired performance [12]. Therefore, the best values have been adopted from the literature. In this experiment, the parameters values for ACS and GA were selected based on recommended values from [27], [41] respectively. Table I presents the parameters values for ACS algorithm.

TABLE I. ACS PARAMETERS VALUES.

Run time	Beta	Evaporation rate	No of ants	q
45second	8	0.6	10	0.9

Table II shows the parameters values for GA. The total population size of GA set to 10 while the selected population size as an intermediate population was set to 6. The probability to operate a crossover operation is 0.9 while the probability to operate a mutation operation is 0.4 [41].

TABLE II. GA PARAMETERS VALUES.

Run time	Population size	Intermediate size	Crossover rate	Mutation rate
45second	10	6	0.9	0.4

Important operators in GA are presented in Table III. To select a population from the population pool, many operators are available such as the roulette wheel and ranking. This study has implemented a tournament operator with value 3 as a selection operator. For crossover operator, fitness based operator is found as the best operator compared with m-point crossover and uniform crossover [41]. Finally, a Re-balanced operator is used as a mutation operator, which is considered better than random mutation.

TABLE III. GA IMPLEMENTED OPERATORS

Elitism	Selection operator	Crossover operator	Mutation operator
True	Tournament = 3	Fitness based	Re-balanced

Experiments have been conducted using Intel® Core (TM) i7-3612QM CPU @ 2.10GHz and 8G RAM. The grid computing simulator is developed using visual C#. The time given for each experiment is 90 seconds (45 seconds for each algorithm). This time restriction is a very important requirement to mimic the real environment for job scheduling in grid computing [31], [42]. Each algorithm was executed 10 times in order to calculate the average values as well as to get the best run. The first column of each table represents the instance name with an abbreviation code: x-yyzz as follows:

x represent the type of consistency; c means consistent, i means inconsistent, and s means semi-consistent.

yy represents the heterogeneity of the tasks; hi means high and lo means low.

zz represents the heterogeneity of the machines; hi means high and lo means low.

For example: c_hilo means consistent environment, hi heterogeneity in tasks and low heterogeneity in machines.

The results show that the proposed algorithm was able to reduce the makespan significantly on seven instances as illustrated in Table IV which shows the best makespan values.

TABLE IV. BEST MAKESPAN VALUES.

	GA	AS	ACS	ACS+GA
c_hihi	11215488.93	11210553.9	10794610.75	**10533616.36**
c_hilo	182232.04	184701.33	**179762.4**	180289.84
c_lohi	374685.96	367182.79	346838.43	**345233.25**
c_lolo	6138.52	6224.75	6051.82	**6001.86**
i_hihi	3995843.41	3946883.19	4066163.68	**3924281.6**
i_hilo	91682.28	**90968.26**	93829	91709.93
i_lohi	134151.08	**133825.44**	137176.54	134796.3
i_lolo	**3045.32**	3140.97	3208.97	3164.29
s_hihi	6223749.51	5991234.31	6119601.97	**5854357.25**
s_hilo	120447.26	**118988.3**	120539.13	119123.89
s_lohi	181155.5	176800.44	178584.84	**172225.04**
s_lolo	4246.4	4296.32	4350.38	**4225.71**

Table V depicts the average values for makespan. The proposed algorithm was able to achieve good results on five instances. However, GA also performs well on four instances.

TABLE V. AVG MAKESPAN VALUES.

	GA	AS	ACS	ACS+GA
c_hihi	11266455.65	11492186.36	10947366.92	**10849427.27**
c_hilo	183264.856	186640.051	181434.422	**180970.805**
c_lohi	375322.186	373766.649	**353670.849**	353882.764
c_lolo	6152.468	6281.502	6120.002	**6074.341**
i_hihi	4029108.699	**4021032.464**	4261681.833	4115442.339
i_hilo	**91682.28**	92311.613	94832.7	93513.988
i_lohi	**135625.029**	136721.893	144178.472	138746.886
i_lolo	**3051.006**	3198.568	3279.985	3232.719
s_hihi	6317823.165	**6114693.995**	6322969.763	6119177.625
s_hilo	120664.355	121995.849	122440.437	**120576.822**
s_lohi	181734.596	178990.539	181737.421	**177965.139**
s_lolo	**4249.935**	4369.079	4399.443	4326.294

The experiments show different performance for flowtime objective. AS algorithm outperform the other algorithms for the best and average flowtime values as shown in Table VI and Table VII. This behavior was expected due to the contradiction between makespan and flowtime.

TABLE VI. BEST FLOWTIME VALUES.

	GA	AS	ACS	ACS+GA
c_hihi	175890174.2	170869481	**167168928**	167921346.2
c_hilo	2885387.55	**2839818.65**	2839974.6	2855393.95
c_lohi	5862262.04	5600439.31	5481314.05	**5475878.29**
c_lolo	97154.47	95877	95871.53	**94911.38**
i_hihi	63759167.63	**60169758.16**	64092691.04	62544930.6
i_hilo	1461297.38	**1403670.42**	1451182.04	1463099.33
i_lohi	2141505.91	**2032456.42**	2150374.03	2152416.88
i_lolo	**48547.9**	48773.48	50707.62	50529.25
s_hihi	98814397.03	**90312215.73**	95998535.04	92830865.83
s_hilo	1909954.11	**1832927.6**	1893970.67	1891505.22
s_lohi	2867157.87	**2682621.46**	2800124.77	2746952.11
s_lolo	67508.13	**65545.51**	68232.02	67152.14

TABLE VII. AVG FLOWTIME VALUES.

	GA	AS	ACS	ACS+GA
c_hihi	176638718.7	174513587.9	**171594188.4**	171864310.1
c_hilo	2893345.641	2866863.113	**2865314.197**	2867622.027
c_lohi	5867869.085	5712409.208	**5587489.199**	5597133.705
c_lolo	97298.915	96857.627	96697.087	**96332.486**
i_hihi	64261850.79	**61409716.3**	66654183.7	65559896.86
i_hilo	1461683.727	**1422434.616**	1489277.24	1492734.607
i_lohi	2163840.832	**2068376.494**	2256605.345	2212084.909
i_lolo	**48579.506**	49416.302	51606.347	51580.551
s_hihi	99887497.75	**92951306.33**	98799209.66	97232283.39
s_hilo	1915659.179	**1867344.085**	1934073.416	1917926.183
s_lohi	2871564.91	**2738879.14**	2869869.222	2830359.079
s_lolo	67548.438	**67048.336**	69185.328	68780.228

In order to represent the performance of the proposed algorithm visually, a geometric mean is used to normalize the makespan and flowtime values of the 12 instances [43]. Figure 2 displays the results of the proposed algorithm, which is the best among other algorithms for best makespan values. In addition, Figure 3 shows the same for average flowtime values.

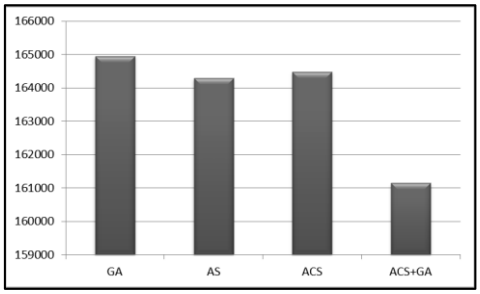

Fig. 2. Geometric mean of best makespan for 12 instances.

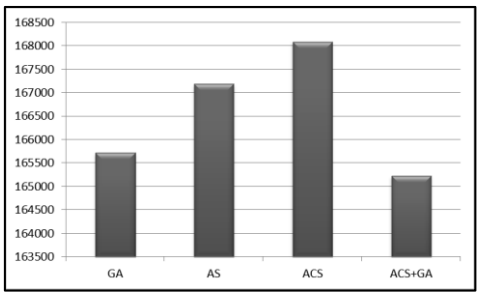

Fig. 3. Geometric mean of AVG makespan for 12 instances.

For the best and average flowtime values, Figures 4 and 5 present the geometric mean values of 12 instances respectively. The results show that AS algorithm outperforms other algorithms.

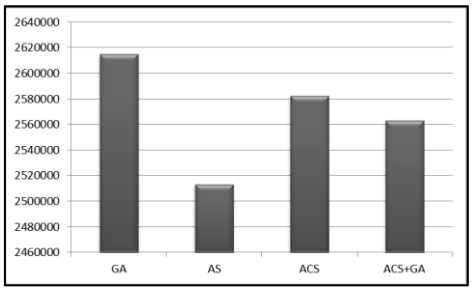

Fig. 4. Geometric mean of best flowtime for 12 instances

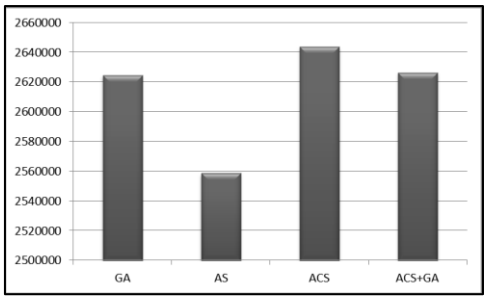

Fig. 5. Geometric mean of AVG flowtime for 12 instances.

VI. CONCLUSION

Job scheduling in grid computing system needs a metaheuristics algorithm to be solved efficiently. Due to the complexity of the problem, stand-alone algorithm is insufficient for some cases. However, hybrid metaheuristics algorithms perform better than stand-alone algorithm in solving many combinatorial problems. This study has implemented a high level hybridization between ACS and GA to solve job scheduling in grid computing system. The results showed that the proposed algorithm outperforms other algorithms in terms of makespan reduction. Future work related to the proposed hybridization algorithm will focus on hybrid ACS with local search algorithms and the implementation of the hybrid algorithm in dynamic grid computing environment.

ACKNOWLEDGMENT

The authors wish to thank the Ministry of Higher Education Malaysia for funding this study under the Fundamental Research Grant Scheme, S/O codes 12819 and 11980, and RIMC, Universiti Utara Malaysia, Kedah, for the administration of this study.

REFERENCES

[1] F. Xhafa and A. Abraham, "Computational Models and Heuristic Methods for Grid Scheduling Problems," J. Futur. Gener. Comput. Syst., vol. 26, no. 4, pp. 608–621, 2010.

[2] F. Magoules, I. Pan, K.-A. Tan, and A. Kumar, Introduction to Grid Computing. Boca Raton: CRC Press, 2009.

[3] J. Kolodziej, Evolutionary Hierarchical Multi-Criteria Metaheuristics for Scheduling in Large-Scale Grid Systems. Berlin New York: Springer, 2012.

[4] J. Montes, A. Sanchez, and M. S. Perez, "Riding Out the Storm: How to Deal with the Complexity of Grid and Cloud Management," J. Grid Comput., vol. 10, no. 3, pp. 349–366, Aug. 2012.

[5] O. Babafemi, M. Sanjay, and M. Adigun, "Towards Developing Grid-based Portals for e-Commerce on-Demand Services on a Utility Computing Platform," J. IERI Procedia, vol. 4, pp. 81–87, Jan. 2013.

[6] M. L. Bote-Lorenzo, Y. A. Dimitriadis, and E. Gomez-Sanchez, "Grid Characteristics and Uses: A Grid Definition," in Proceedings of the 1st European Across Grids Conference, 2004, vol. 2970, pp. 291–298.

[7] D. Neumann, J. Stober, C. Weinhardt, and J. Nimis, "A Framework for Commercial Grids—Economic and Technical Challenges," J. Grid Comput., vol. 6, no. 3, pp. 325–347, 2008.

[8] A. Abraham, R. Buyya, and B. Nath, "Nature's Heuristics for Scheduling Jobs on Computational Grids," in Proceedings of the 8th IEEE International Conference on Advanced Computing and Communications, 2000, pp. 45–52.

[9] E. Amiri, H. Keshavarz, N. Ohshima, and S. Komaki, “Resource Allocation in Grid: A Review,” J. Procedia - Soc. Behav. Sci., vol. 129, no. 1, pp. 436–440, 2014.

[10] H. B. Prajapati and V. A. Shah, “Scheduling in Grid Computing Environment,” in Proceedings of the 4th International Conference on Advanced Computing & Communication Technologies, 2014, pp. 315–324.

[11] F. Xhafa, B. Duran, and J. Kolodziej, “On Exploitation vs Exploration of Solution Space for Grid Scheduling,” in Proceedings of the 3rd International Conference on Intelligent Networking and Collaborative Systems, 2011, pp. 164–171.

[12] G. Zapfel, R. Braune, and M. Bogl, Metaheuristic Search Concepts a Tutorial with Applications to Production and Logistics. Berlin, Heidelberg: Springer, 2010.

[13] I. Chaari, A. Koubaa, H. Bennaceur, S. Trigui, and K. Al-Shalfan, “SmartPATH: A Hybrid ACO-GA Algorithm for Robot Path Planning,” in Proceedings of the IEEE Congress on Evolutionary Computation, 2012, no. 1, pp. 1–8.

[14] Al-Mahmud and M. A. H. Akhand, “ACO with GA Operators for Solving University Class Scheduling Problem with Flexible Preferences,” in Proceedings of the International Conference on Informatics, Electronics & Vision, 2014, pp. 1–6.

[15] I. Michelakos, N. Mallios, E. Papageorgiou, and M. Vassilakopoulos, “Ant Colony Optimization and Data Mining,” in Next Generation Data Technologies for Collective Computational Intelligence, N. Bessis and F. Xhafa, Eds. Berlin Heidelberg: Springer, 2011, pp. 31–60.

[16] A. Kant, A. Sharma, S. Agarwal, and S. Chandra, “An ACO Approach to Job Scheduling in Grid Environment,” in Proceedings of the 1st International Conference on Swarm, Evolutionary, and Memetic Computing, 2010, vol. 6466, pp. 286–295.

[17] K. Liu, J. Chen, H. Jin, and Y. Yang, “A Min-Min Average Algorithm for Scheduling Transaction-Intensive Grid Workflows,” in Proceedings of the 7th Australasian Symposium on Grid Computing and e-Research, 2009, no. AusGrid, pp. 41–48.

[18] R. Chang, J. Chang, and P.-S. Lin, “An Ant Algorithm for Balanced Job Scheduling in Grids,” J. Futur. Gener. Comput. Syst., vol. 25, no. 1, pp. 20–27, Jan. 2009.

[19] H. U. I. Yan, X. Shen, X. Li, and M. Wu, “An Improved Ant Algorithm for Job Scheduling in Grid Computing,” in Proceedings of the 4th International Conference on Machine Learning and Cybernetics, 2005, no. August, pp. 2957 – 2961.

[20] D. A. Menasce, D. Saha, S. C. D. S. Porto, V. A. F. Almeida, and S. K. Tripathi, “Static and Dynamic Processor Scheduling Disciplines in Heterogeneous Parallel Architectures,” J. Parallel Distrib. Comput., vol. 28, no. 1, 1995.

[21] D. P. da Silva, W. Cirne, and F. V. Brasileiro, “Trading Cycles for Information: Using Replication to Schedule Bag-of-Tasks Applications on Computational Grids,” in Proceedings of the 9th International Euro-Par Conference on Parallel Processing, 2003, vol. 2790, pp. 169–180.

[22] L. M. Nithya and A. Shanmugam, “Scheduling in Computational Grid with a New Hybrid Ant Colony Optimization Algorithm,” Eur. J. Sci. Res., vol. 62, no. 2, pp. 273–281, 2011.

[23] T. D. Braun, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and B. Yao, “A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems,” J. Parallel Distrib. Comput., vol. 61, no. 6, pp. 810–837, 2001.

[24] E. S. Kumar and A. Sumathi, “EACS Approach for Grid Workflow Scheduling in a Computational Grid,” in Proceedings of the 1st International Conference on Computational Intelligence and Information Technology, 2011, vol. 250, pp. 276–280.

[25] L. Mou, “An Efficient Ant Colony System for Solving the New Generalized Traveling Salesman Problem,” in Proceedings of the IEEE International Conference on Cloud Computing and Intelligence Systems, 2011, pp. 407 – 412.

[26] M. Dorigo and L. M. Gambardella, “Ant Colonies for the Travelling Salesman Problem,” J. Biosyst., vol. 43, no. 2, pp. 73–81, 1997.

[27] M. Dorigo and T. Stutzle, Ant Colony Optimization. Cambridge, England: MIT Press, 2004.

[28] J. Kolodziej and F. Xhafa, “Enhancing the Genetic-Based Scheduling in Computational Grids by a Structured Hierarchical Population,” J. Futur. Gener. Comput. Syst., vol. 27, no. 8, pp. 1035–1046, 2011.

[29] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen, and S. Ali, “Task Execution Time Modeling for Heterogeneous Computing Systems,” in Proceedings of the 9th Heterogeneous Computing Workshop, 2000, pp. 185–199.

[30] F. Xhafa and J. Carretero, “Experimental Study of GA-Based Schedulers in Dynamic Distributed Computing Environments,” in Optimization Techniques for Solving Complex Problems, E. Alba, C. Blum, P. Isasi, C. Leon, and J. A. Gomez, Eds. Hoboken, N.J: Wiley, 2009, pp. 423–441.

[31] J. Carretero, F. Xhafa, and A. Abraham, “Genetic Algorithm Based Schedulers for Grid Computing Systems,” Int. J. Innov. Comput. Inf. Control, vol. 3, no. 6, pp. 1–19, 2007.

[32] F. Xhafa, J. Kolodziej, L. Barolli, V. Kolici, R. Miho, and M. Takizawa, “Evaluation of Hybridization of GA and TS Algorithms for Independent Batch Scheduling in Computational Grids,” in Proceedings of the International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 2011, pp. 148–155.

[33] F. Xhafa, J. Carretero, L. Barolli, and A. Durresi, “Requirements for an Event-Based Simulation Package for Grid Systems,” J. Interconnect. Networks, vol. 08, no. 02, pp. 163–178, Jun. 2007.

[34] F. Xhafa, J. Carretero, B. B. Dorronsoro, and E. Alba, “Tabu Search Algorithm for Scheduling Independent Jobs in Computational Grids,” J. Comput. Informatics, vol. 28, no. 2, pp. 237–250, 2009.

[35] F. Xhafa, J. A. Gonzalez, K. P. Dahal, and A. Abraham, “A GA(TS) Hybrid Algorithm for Scheduling in Computational Grids,” in Proceedings of the 4th International Conference on Hybrid Artificial Intelligence Systems, 2009, vol. 5572, pp. 285–292.

[36] F. Xhafa, J. Kolodziej, L. Barolli, and A. Fundo, “A GA+TS Hybrid Algorithm for Independent Batch Scheduling in Computational Grids,” in Proceedings of the 14th International Conference on NetworkBased Information Systems, 2011, pp. 229–235.

[37] L. Jourdan, M. Basseur, and E.-G. Talbi, “Hybridizing Exact Methods and Metaheuristics: A Taxonomy,” Eur. J. Oper. Res., vol. 199, no. 3, pp. 620–629, Dec. 2009.

[38] K. R. Ku-Mahamud and M. M. Alobaedy, “New Heuristic Function in Ant Colony System for Job Scheduling in Grid Computing,” in Proceedings of the 17th International Conference on Applied Mathematics, 2012, pp. 47–52.

[39] F. Xhafa and A. Abraham, “Meta-heuristics for Grid Scheduling Problems,” in Metaheuristics for Scheduling in Distributed Computing Environments, F. Xhafa and A. Abraham, Eds. Berlin Heidelberg: Springer, 2008, pp. 1–37.

[40] G. Ritchie and J. Levine, “A Hybrid Ant Algorithm for Scheduling Independent Jobs in Heterogeneous Computing Environments,” in Proceedings of the 23rd Workshop of the UK Planning and Scheduling Special Interest Group, 2004, pp. 1–7.

[41] F. Xhafa, L. Barolli, and A. Durresi, “An Experimental Study on Genetic Algorithms for Resource Allocation on Grid Systems,” J. Interconnect. Networks, vol. 8, no. 4, pp. 427–443, 2007.

[42] F. Xhafa and B. Duran, “Parallel Memetic Algorithms for Independent Job Scheduling in Computational Grids,” in Recent Advances in Evolutionary Computation for Combinatorial Optimization, vol. 153, C. Cotta and J. van Hemert, Eds. Berlin Heidelberg: Springer, 2008, pp. 219–239.

[43] H. Izakian, A. Abraham, and V. Snsel, “Performance Comparison of Six Efficient Pure Heuristics for Scheduling Meta-Tasks on Heterogeneous Distributed Environments,” J. Neural Netw. World, vol. 6, no. 09, pp. 695–711, 2009.