Research Journal of Applied Sciences, Engineering and Technology 11(7): 806-816, 2015
DOI: 10.19026/rjaset.11.2044
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2015 Maxwell Scientific Publication Corp.
Submitted: June 14, 2015 Accepted: July 8, 2015 Published: November 05, 2015

Research Article

Hybrid Ant Colony System and Genetic Algorithm Approach for Scheduling of Jobs in Computational Grid

Mustafa Muwafak Alobaedy and Ku Ruhana Ku-Mahamud
School of Computing, College of Art and Sciences, Universiti Utara Malaysia,
06010 Sintok, Kedah, Malaysia

Abstract: Metaheuristic algorithms have been used to solve scheduling problems in grid computing. However, stand-alone metaheuristic algorithms do not always show good performance in every problem instance. This study proposes a high level hybrid approach between ant colony system and genetic algorithm for job scheduling in grid computing. The proposed approach is based on a high level hybridization. The proposed hybrid approach is evaluated using the static benchmark problems known as ETC matrix. Experimental results show that the proposed hybridization between the two algorithms outperforms the stand-alone algorithms in terms of best and average makespan values.

Keywords: Hybrid metaheuristic algorithm, job scheduling, static grid computing

INTRODUCTION

Grid computing technology is considered as an intelligent multi-level platform that provides a wide range of services (Kołodziej and Khan, 2012). Grid computing is defined as "geographically distributed computers, linked through the Internet in a grid-like manner and are used to create virtual supercomputers of vast amount of computing capacity able to solve complex problems from e-Science in less time than known before" (Xhafa and Abraham, 2010). Another definition for grid computing is "a form of distributed computing that coordinates and provides the facility of resource sharing over various geographical locations" (Rajni and Chana, 2013). From these definitions, grid computing could be defined as a technology of connecting various resources distributed in different locations with the aim to provide various services.

Grid systems evolve from existing technology such as distributed computing, web service and the Internet (Magoules *et al*., 2009). Grid systems are classified as modern High Performance Distributed Systems (HPDSs) along with clusters and cloud systems (Kolodziej, 2012). However, there are crucial characteristics which differ between them, such as scale, network type, administrative domain, resources structure and security (Hsu *et al*., 2011; Kołodziej *et al*., 2014; Montes *et al*., 2012).

There are many different types of grid systems, such as:

- Sensor grid, which is based on sharing sensor resources in a sensor network (Li and Li, 2014)
- Campus grid, which is implemented in campus environments in order to facilitate unified access to the distributed and heterogeneous resources such as clusters, storage and scientific instruments (Bose *et al*., 2004)
- Data grid, which is mainly designed to provide data-intensive applications that need to access, transfer and modify massive data stored in distributed storage resources (Venugopal *et al*., 2006)
- Desktop grid, another important type of grid developed to connect Personal Computers (PCs) with large-scale networks using the Internet or any other high-speed connection media (Kolodziej, 2012)
- Utility grid, which is based on providing computing services to the users or organizations in return for regular payment (Babafemi *et al*., 2013; Garg *et al*., 2009).

Grid system has been utilized in various fields, such as the high energy physics grid in Japan (Sakamoto, 2007), molecular systems using grid environment (Costantini *et al*., 2014), multi-physics coupled applications using Batch Grid (Murugavel *et al*., 2011), Enzyme Design Process using University of California Grid (Wang *et al*., 2011), medical informatics using GridR environment which is based on embedding R software into grid framework (Wegener

Corresponding Author: Mustafa Muwafak Alobaedy, School of Computing, College of Art and Sciences, Universiti Utara Malaysia, 06010 Sintok, Kedah, Malaysia

et al., 2009), processing of scientific knowledge using high performance grid computing by means of natural language processing and text mining (Jeong *et al*., 2014), Climate Simulations for Europe and India regions based on grid computing environment (Cozzini *et al*., 2014), 3D electrophoresis coupled problem simulation based on Asynchronous grid computing environment (Chau *et al*., 2013), high-resolution agricultural systems modelling using grid computing and parallel processing (Zhao *et al*., 2013), services for neuroimaging analysis using Intelligent grid based on neuGRID project (Richard *et al*., 2013), The Earth System Grid Federation (ESGF) which is based on nodes that are geographically distributed around the world (Cinquini *et al*., 2012) and chemistry experiment tools based on PL-Grid environment (Eilmes *et al*., 2014).

One of the main components in grid computing systems is the Resource Management System (RMS) which is required in providing and sharing the resources efficiently in the grid environment (Czerwinski *et al*., 2012; Siddiqui and Fahringer, 2005). RMS could be implemented with one or multiple resource management nodes called Resource Manager (RM) (Qu *et al*., 2005). Resource management in grid computing is a challenging task due to the heterogeneous, dynamic, autonomous and ephemeral grid resources (Li and Li, 2012). RMS has several services, such as Grid Information Services (GIS), monitoring the status of tasks and environment, resource scheduler, resource reservation, accounting and reporting (Czerwinski *et al*., 2012; Abraham *et al*., 2000). The scheduler has the main influence in grid computing performance (Amiri *et al*., 2014). The scheduler's responsibility is to map the submitted jobs from users to the suitable and available resources (Qureshi *et al*., 2014). The efficiency of the scheduler depends on the implemented scheduling 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 of jobs in grid computing is known as an NP-complete problem due to the complexity and intractable nature of the problem (Burkimsher *et al*., 2013; Prajapati and Shah, 2014), which could be solved using metaheuristic algorithms. These types of algorithms have the ability to find near optimal solution in reasonable time compared to optimal solution in a very long processing time (Xhafa *et al*., 2011a). 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 (Zapfel *et al*., 2010). However, hybridizing two or more algorithms show better performance than applying a stand-alone algorithm (Kolodziej, 2012). This is due to the ability of the hybrid approach to skip from local minima, using more options available in the algorithms used for the hybridization. Hybrid approaches between ACO and GA have been studied in Chaari *et al*. (2012) and Al-Mahmud and Akhand (2014). 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 Chaari *et al*. (2012) and Al-Mahmud and Akhand (2014) to solve robot path planning and university class scheduling respectively. 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.

METAHEURISTIC 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 (Amiri *et al*., 2014). Job scheduling is known as an NP-complete problem which needs metaheuristic algorithms to be solved (Folino and Mastroianni, 2010). One of the best metaheuristic algorithms in the field of optimization is ACO. ACO is considered as a swarm intelligence algorithm which mimics the behaviours of real biological ants. ACO is implemented to solve various problems, such as routing (Wang *et al*., 2013), scheduling (Neto and Filho, 2013) and classification (Michelakos *et al*., 2011). Many studies have implemented and enhanced ACO for job scheduling in grid computing. An ACO approach for job scheduling in grid system in Kant *et al*. (2010) 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 Liu *et al*. (2009) and first come first serve algorithm. Experimental results show that this algorithm outperforms the other two algorithms.

A study presented by Chang *et al*. (2009) proposed the Balanced ACO (BACO) algorithm for job scheduling in grid. The proposed algorithm is based on the basic ideas from the 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 availability 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 the BACO algorithm outperforms the improved ACO in Yan *et al*. (2005), fastest processor to largest task first (Menasce *et al*., 1995) and Sufferage (Silva *et al*., 2003).

A hybrid ACO approach (HACO) for job scheduling in grid computing proposed in Nithya *et al.* (2011) has integrated the heuristic information to make the algorithm converge faster to the solution. The experiments conducted have used the benchmark model known as Expected Time to Compute (ETC) model presented in Braun *et al.* (2001). 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 presented in Kumar and Sumathi (2011) is known as the ant colony system developed by Dorigo and Gambardella (1997). ACS algorithm enhances ant system in three phases: first, the exploration mechanism becomes stronger due to the implementation of the 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, the evaporation process will be applied only to the arcs used by ants to increase the exploration of alternative arcs (Dorigo and Stutzle, 2004).

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 (Sivanandam and Deepa, 2008). GA mimics the evolutionary process by applying selection, recombination and mutation to generate solutions 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 Kołodziej *et al.* (2011b). 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 flow time simultaneously which have been optimized. Experiments were conducted under heterogeneous, large scale and dynamic environments using the grid simulator. HGS was tested with static and dynamic grid computing environment. The experiment with static environment is based on the ETC matrix model presented by Ali *et al.* (2000a) and for dynamic environment, the authors used a simulator presented by Xhafa and Carretero (2009). HGS was also compared with two other GA-based schedulers presented in Braun *et al.* (2001) and Carretero *et al.* (2007). The results show that HGS outperforms the other GA-based schedulers. However, it is not known how HGS will perform against other metaheuristic algorithms, since only GA-based algorithms were used for comparison.

A study presented by Xhafa *et al.* (2011c) 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-objective 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 Xhafa *et al.* (2007a). The proposed algorithms were compared with GA presented by Carretero *et al.* (2007) and TS presented by Xhafa *et al.* (2009a). 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.

Kim *et al.* (2013) applied Artificial Bee Colony (ABC) for job scheduling in computational grid. The authors proposed Binary ABC (BABC), Efficient Binary Artificial Bee Colony (EBABC1) and flexible ranking strategy (EBABC2) algorithms. The study aimed to minimize the makespan criterion for job scheduling in grid computing. The experiments were conducted using a series of benchmark problems defined in Liu *et al.* (2010). The proposed algorithms were compared with genetic algorithm, simulated algorithm and particle swarm optimization algorithm. In terms of makespan criterion, EBABC1 and EBABC2 algorithms achieved the best results among all other algorithms with superior performance for EBABC2.

Nayak *et al.* (2012) proposed an algorithm which combined the merits of genetic algorithm and bacterial foraging optimization algorithm called Genetic Bacterial Foraging (GBF). The proposed algorithm implemented a dynamic mutation as presented in Michalewicz (1996) and crossover operator developed by Michalewicz (1999). The aim of the study is to reduce the execution time as a cost function. The experiment was conducted using a dynamic environment generated with a simulator developed by the authors. The proposed algorithm was compared with Bacterial Foraging Optimization (BFO) algorithm. The experiment results show that the proposed GBF algorithm outperforms BFO algorithm. However, the experiment scenario was very small, using only four resources and five tasks. Therefore, more studies are required to understand the behavior of Bacterial Foraging Optimization algorithm.

Rajni and Chana (2013) conducted a study on Bacterial Foraging Optimization (BFO) algorithm for resource scheduling on computational grid systems. The study aimed to optimize makespan and cost values by considering Resource Provisioning (RP) units adopted from Aron and Chana (2012). The proposed approach was implemented using the GridSim simulator developed by Buyya and Murshed (2002).

The experiments were conducted by generating a workload using a model defined in Lublin and Feitelson (2003) and the expected time to compute the model presented in Ali *et al*. (2000b). The authors compared the proposed algorithm with genetic algorithm, simulated annealing and GA-TS algorithms. The experiment results show that the proposed BFO algorithm outperforms other algorithms in terms of makespan and cost values for both low and high machine heterogeneity benchmark problems. In addition, the results show that the Coefficient of Variation (CV) of the proposed algorithm is in the range 0-2%, which confirms the stability of the proposed algorithm.

A comparison of four metaheuristic algorithms for task scheduling in computational grid system was presented by Meihong *et al*. (2010). The algorithms used in their study for comparison are genetic algorithm, ant colony optimization algorithm, particle swarm optimization algorithm and simulated annealing algorithms. The evaluation criteria are makespan and mean response time. The authors conducted experiments using static environment. The results show that the PSO algorithm has the best performance among the other algorithms. However, the experiments were conducted using very small scenarios (5 users and 3 resources). Therefore, the robustness of the compared algorithms is not proven. In addition, only classical versions of the algorithms are used, while enhanced versions are better in terms of performance. In order to obtain a clear picture about which metaheuristic is better, more investigations and experiments are required using a known benchmark such as the one presented in Braun *et al*. (2001).

Izakian *et al*. (2010) proposed a discrete particle swarm optimization for job scheduling in grid computing. Their approach aims to minimize the makespan and flowtime simultaneously in grid computing. In their study, they provided two representations for mapping between problem solution and PSO particle. The first representation used a direct encoding that is a vector with the size equal to the number of tasks. Each element in the vector represents the machine number. The second representation used a binary matrix size of (jobs number * machines number). The matrix was represented with values of either 0 or 1. The benchmark problem used to evaluate the proposed algorithm is based on the expected time to compute the model presented by Braun *et al*. (2001). The proposed algorithm was compared with GA, ACO, PSO and Fuzzy PSO algorithms. The experiment results show that the proposed algorithm achieved good results in makespan reduction, while for flowtime, the algorithm performed the worst. Although the study aims to minimize makespan and flowtime, the contradiction is clear between them such that the algorithm could not reduce both of them simultaneously. This contradiction is mentioned by Xhafa and Abraham (2010) in grid computing as well. In general, the proposed algorithm performs better than other algorithms.

Another study using fuzzy particle swarm optimization for job scheduling in grid computing has been proposed in Liu *et al*. (2010). In their algorithm, they extended the velocity and position of particles from the real vectors to fuzzy matrices. The advantages of using fuzzy matrices in PSO are the speed of convergence and the increase of the ability to find a faster and feasible solution. The study used the makespan criterion to measure the algorithm's performance. The performance of the proposed algorithm was compared with genetic algorithm and simulated annealing algorithm. The experiment results show that the proposed algorithm outperforms the other algorithms especially in terms of execution time. However, the study did not use a common benchmark in order to evaluate the proposed algorithm with other approaches. In addition, only genetic algorithm and simulated annealing algorithms were used for comparison, which are also not enough to give a complete picture.

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 (Xhafa *et al*., 2009b). Algorithms could be fully or partially hybridized 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 (Xhafa *et al*., 2011c). 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 into 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 (Jourdan *et al*., 2009; Xhafa *et al*., 2011b).

This study implements a high level hybridization approach, namely ACS+GA. ACS will start first for a specific time and after ACS finishes execution, GA will

```
Procedure ACS+GA
Initialize the number of ants $n$;
Initialize parameters and pheromone trails;
While (Termination condition not met) Do
        For i = 1 to $n$ Do
                Construct new solution;
                Apply local pheromone update;
        End For
        Apply pheromone evaporation;
        Apply Global pheromone update
        Update best found solution $s^*$;
End while;
// Genetic algorithm start here;
Initialize population (P);
Add (best found solution from ACS to P);
Evaluate (P);
While (termination condition not met)
        $\acute{P}$← Select (P);
        Crossover ($\acute{P}$);
        Mutate ($\acute{P}$);
        Evaluate ($\acute{P}$);
        P ← Replace ($\acute{P}$ ∪ P);
End While;
Return the best solution;
End procedure;
```

Fig. 1: ACS+GA (high level) pseudocode

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 $\{1 / (ETC_{ij} + Load_j)\}$ where ETC_{ij} represents the expected time to compute $task\ i$on $machine\ j$ and $Load_j$ is the previous load assigned to $machine\ j$ (Ku-Mahamud and Alobaedy, 2012). 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:

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

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], $q0$ $(0 \le q0 \le 1)$ is a parameter which determines the exploration/exploitation rate and J is a random variable selected according to the probability given by Eq. (2) with $\alpha = 1$ (Dorigo and Stutzle, 2004):

$$\mathrm{P}_{ij}^{\mathrm{Antk}} = \frac{[t_{ij}]^{\alpha}.[\eta]^{\beta}}{\sum_{j=1}^{M}[t_{ij}]^{\alpha}.[\eta]^{\beta}} \tag{2}$$

For GA algorithm implementation, the output from the 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 pseudocode of the proposed algorithm.

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 (Xhafa and Abraham, 2008). In this study, two criteria are implemented, namely makespan and flowtime, with the priority to makespan as the main optimization objective. Makespan metric measures the general productivity of 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:

$$minimization\ of\ makespan: \\ min\ S_{i\,\in\,Sched}\{max_{j\,\in Jobs}F_j\} \tag{3}$$

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 (Xhafa and Abraham, 2008). 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 time of all tasks, formally defined as:

$$flowtime: min\ S_{i\,\in\,Sched}\{\textstyle\sum_{j\,\in Jobs} F_j\} \tag{4}$$

These criteria could conflict with each other since limited resources could be the bottleneck of the system (Xhafa and Abraham, 2008).

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 Braun *et al.* (2001). This model is widely accepted by researchers to be used for job scheduling in

Table 1: ACS parameter values

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

Table 2: GA parameter values

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

Table 3: GA implemented operators

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

grid (Braun *et al*., 2001; Garg *et al*., 2010; Kolodziej *et al*., 2011a, 2011b; Ritchie and Levine, 2004). 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. 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, the ETC matrix captures other possible features of a 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 others slower. Finally, the 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 (Braun *et al*., 2001).

EXPERIMENTS AND RESULTS

Metaheuristic 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 (Zapfel *et al*., 2010). Therefore, the best values have been adopted from the literature. In this experiment, the parameter values for ACS and GA are selected based on the recommended values from Dorigo and Stutzle (2004) and Xhafa *et al*. (2007b) respectively. Table 1 presents the parameter values for the ACS algorithm.

Table 2 shows the parameter values for GA. The total population size of GA is set to 10, while the selected population size as an intermediate population is set to 6. The probability to operate a crossover operation is 0.9, while the probability to operate a mutation operation is 0.4 (Xhafa *et al*., 2007a).

Important operators in GA are presented in Table 3. 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, the fitness-based operator is found as the best operator compared with m-point crossover and uniform crossover (Xhafa *et al*., 2007b). Finally, a Re-balanced operator is used as a mutation operator, which is considered better than random mutation.

Experiments have been conducted using Intel® Core(TM) i7-3612QM CPU @ 2.10 GHz and 8G RAM. The grid computing simulator is developed using visual C#. The time given for each experiment is 90 sec (45 sec for each algorithm). This time restriction is a very important requirement to mimic the real environment for job scheduling in grid computing (Carretero *et al*., 2007; Xhafa and Duran, 2008). Each algorithm is 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 represents 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 is able to reduce the makespan significantly on seven instances as illustrated in Table 4, which shows the best makespan values.

Table 5 depicts the average values for makespan. The proposed algorithm is able to achieve good results on five instances. However, GA also performs well on four instances.

The experiments show different performance for flowtime objective. The AS algorithm outperforms the

Table 4: 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 5: Average 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

Table 6: 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 7: Average 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

other algorithms for the best and average flowtime values as shown in Table 6 and 7 respectively. This behavior is expected due to the contradiction between makespan and flowtime.

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 (Izakian *et al*., 2009). Figure 2 displays the results of the proposed algorithm, which is the best among other algorithms for best makespan values. In addition, Fig. 3 shows the same for average makespan values.

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

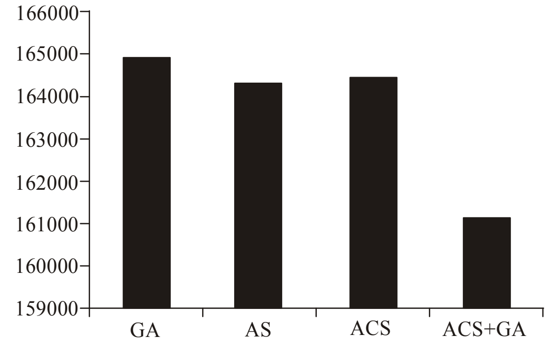

Fig. 2: Geometric mean of best makespan for 12 instances.

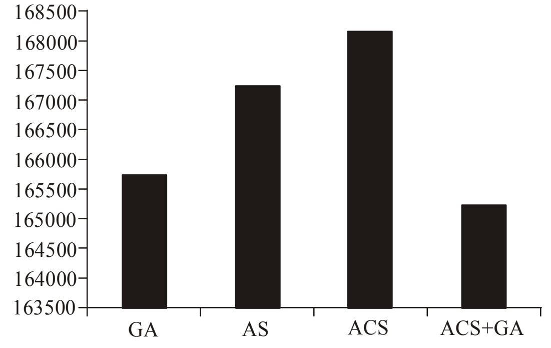

Fig. 3: Geometric mean of average makespan for 12 instances

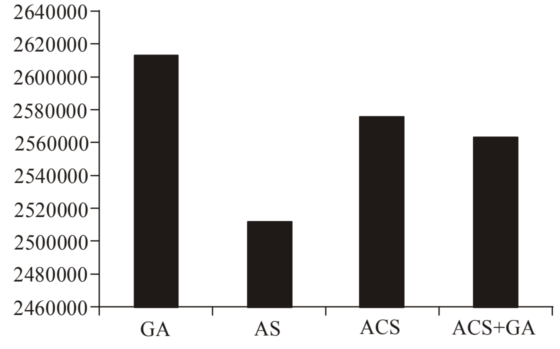

Fig. 4: Geometric mean of best flowtime for 12 instances

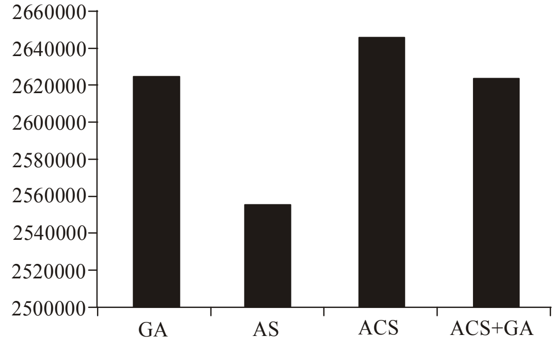

Fig. 5: Geometric mean of AVG flowtime for 12 instances

CONCLUSION

Job scheduling in grid computing system needs a metaheuristic algorithm to be solved efficiently. Due to the complexity of the problem, stand-alone algorithm is insufficient for some cases. However, hybrid metaheuristic 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

Abraham, A., R. Buyya and B. Nath, 2000. Nature’s heuristics for scheduling jobs on computational grids. Proceeding of the 8th IEEE International Conference on Advanced Computing and Communications. New York, pp: 45-52.

Ali, S.S., H.J. Siegel, M. Maheswaran, D. Hensgen and W. Lafayette, 2000a. Task execution time modeling for heterogeneous computing systems. Proceeding of the 9th Heterogeneous Computing Workshop Cancun, pp: 185-199.

Ali, S., H.J. Siegel, M. Maheswaran, D. Hensgen and S. Ali, 2000b. Representing task and machine heterogeneities for heterogeneous computing systems. Tamkang J. Sci. Eng., 3(3): 195-207.

Al-Mahmud and M.A.H. Akhand, 2014. ACO with GA operators for solving university class scheduling problem with flexible preferences. Proceeding of the International Conference on Informatics, Electronics and Vision. Dhaka, pp: 1-6.

Amiri, E., H. Keshavarz, N. Ohshima and S. Komaki, 2014. Resource allocation in grid: A review. Proc. Soc. Behav. Sci., 129(1): 436-440.

Aron, R. and I. Chana, 2012. Formal QoS policy based grid resource provisioning framework. J. Grid Comput., 10(2): 249-264.

Babafemi, O., M. Sanjay and M. Adigun, 2013. Towards developing grid-based portals for e-commerce on-demand services on a utility computing platform. IERI Proc., 4(1): 81-87.

Bose, A., B. Wickman, C. Wood and A. Arbor, 2004. MARS: A metascheduler for distributed resources in campus grids. Proceeding of the 5th IEEE/ACM International Workshop on Grid Computing. Los Alamitos, pp: 110-118.

Braun, T.D., H.J. Siegel, N. Beck, L.L. Boloni, M. Maheswaran, A.I. Reuther and R.F. Freund, 2001. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distr. Com., 61(6): 810-837.

Burkimsher, A., I. Bate and L.S. Indrusiak, 2013. A survey of scheduling metrics and an improved ordering policy for list schedulers operating on workloads with dependencies and a wide variation in execution times. Future Gener. Comp. Sy., 29(8): 2009-2025.

Buyya, R. and M. Murshed, 2002. GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurr. Comp-Pract. E., 14(13): 1175-1220.

Carretero, J., F. Xhafa and A. Abraham, 2007. Genetic algorithm based schedulers for grid computing systems. Int. J. Innov. Comput. I., 3(6): 1-19.

Chaari, I., A. Koubaa, H. Bennaceur, S. Trigui and K. Al-Shalfan, 2012. SmartPATH: A hybrid ACO-GA algorithm for robot path planning. Proceeding of the IEEE Congress on Evolutionary Computation. Brisbane, pp: 1-8.

Chang, R., J. Chang and P.S. Lin, 2009. An ant algorithm for balanced job scheduling in grids. Future Gener. Comp. Sy., 25(1): 20-27.

Chau, M., T. Garcia and P. Spiteri, 2013. Asynchronous grid computing for the simulation of the 3D electrophoresis coupled problem. Adv. Eng. Softw., 60-61(1): 111-121.

Cinquini, L., D. Crichton, C. Mattmann, J. Harney, G. Shipman, F. Wang and R. Schweitzer, 2012. The earth system grid federation: An open infrastructure for access to distributed geospatial data. Proceeding of the 8th International Conference on E-Science. Chicago, pp: 1-10.

Costantini, A., O. Gervasi, F. Zollo and L. Caprini, 2014. User interaction and data management for large scale grid applications. J. Grid Comput., 12(3): 485-497.

Cozzini, S., D. Vaddi, S. Goel, F. De Giorgi and S.K. Dash, 2014. Regional climate simulations on EU-India grid infrastructures: Methodologies and performance. J. Grid Comput., 12: 303-320.

Czerwinski, D., S. Przylucki and P. Matejczuk, 2012. Resource management in grid systems. Proceeding of the 19th International Conference on Computer Networks. Szczyrk, pp: 101-110.

Dorigo, M. and L.M. Gambardella, 1997. Ant colonies for the travelling salesman problem. BioSystems, 43(2): 73-81.

Dorigo, M. and T. Stutzle, 2004. Ant Colony Optimization. MIT Press, Cambridge, Mass.

Eilmes, A., M. Sterzel, T. Szepieniec, J. Kocot, K. Noga and M. Golik, 2014. Comprehensive Support for Chemistry Computations in PL-grid Infrastructure. In: Bubak, M., J. Kitowski and K. Wiatr (Eds.), eScience on Distributed Computing Infrastructure: Achievements of PLGrid Plus Domain-specific Services and Tools. Springer, International Publishing, Switzerland, pp: 250-262.

Folino, G. and C. Mastroianni, 2010. Special section: Bio-inspired algorithms for distributed systems. Future Gener. Comp. Sy., 26(6): 835-837.

Garg, S.K., R. Buyya and H.J. Siegel, 2009. Scheduling parallel applications on utility grids: Time and cost trade-off management. Proceeding of the 32nd Australasian Conference on Computer Science. Wellington, pp: 151-160.

Garg, S.K., R. Buyya and H.J. Siegel, 2010. Time and cost trade-off management for scheduling parallel applications on utility grids. Future Gener. Comp. Sy., 26(8): 1344-1355.

Hsu, C., K. Huang and F. Wang, 2011. Online scheduling of workflow applications in grid environments. Future Gener. Comp. Sy., 27(6): 860-870.

Izakian, H., A. Abraham and V. Snsel, 2009. Performance comparison of six efficient pure heuristics for scheduling meta-tasks on heterogeneous distributed environments. Neural Netw. World, 6(9): 695-711.

Izakian, H., B.T. Ladani, A. Abraham and V. Snasel, 2010. A discrete particle swarm optimization approach for grid job scheduling. Int. J. Innov. Comput. I., 6(9): 1-15.

Jeong, C.H., Y.S. Choi, H.W. Chun, S.K. Song, H. Jung, S. Lee and S.P. Choi, 2014. Grid-based framework for high-performance processing of scientific knowledge. Multimed. Tools Appl., 71(2): 783-798.

Jourdan, L., M. Basseur and E.G. Talbi, 2009. Hybridizing exact methods and metaheuristics: A taxonomy. Eur. J. Oper. Res., 199(3): 620-629.

Kant, A., A. Sharma, S. Agarwal and S. Chandra, 2010. An ACO approach to job scheduling in grid environment. Proceeding of the 1st International Conference on Swarm, Evolutionary and Memetic Computing. Chennai, pp: 286-295.

Kim, S.S., J.H. Byeon, H. Liu, A. Abraham and S. McLoone, 2013. Optimal job scheduling in grid computing using efficient binary artificial bee colony optimization. Soft Comput., 17(5): 867-882.

Kolodziej, J., 2012. Evolutionary Hierarchical Multi-criteria Metaheuristics for Scheduling in Large-scale Grid Systems. Springer, New York.

Kołodziej, J. and S.U. Khan, 2012. Multi-level hierarchic genetic-based scheduling of independent jobs in dynamic heterogeneous grid environment. Inform. Sciences, 214(1): 1-19.

Kolodziej, J., S.U. Khan and F. Xhafa, 2011a. Genetic algorithms for energy-aware scheduling in computational grids. Proceeding of the International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. Barcelona, pp: 17-24.

Kołodziej, J., F. Xhafa and J. Kolodziej, 2011b. Enhancing the genetic-based scheduling in computational grids by a structured hierarchical population. Future Gener. Comp. Sy., 27(8): 1035-1046.

Kołodziej, J., S.U. Khan, L. Wang, M. Kisiel-Dorohinicki, S.A. Madani, E. Niewiadomska-Szynkiewicz and C.Z. Xu, 2014. Security, energy and performance-aware resource allocation mechanisms for computational grids. Future Gener. Comp. Sy., 31(1): 77-92.

Ku-Mahamud, K.R. and M.M. Alobaedy, 2012. New heuristic function in ant colony system for job scheduling in grid computing. Proceeding of the 17th International Conference on Applied Mathematics. Montreux, pp: 47-52.

Kumar, E.S. and A. Sumathi, 2011. EACS approach for grid workflow scheduling in a computational grid. Proceeding of the 1st International Conference on Computational Intelligence and Information Technology. Pune, 250: 276-280.

Li, C. and L. Li, 2012. Design and implementation of economics-based resource management system in ad hoc grid. Adv. Eng. Softw., 45(1): 281-291.

Li, C. and L. Li, 2014. Sensor grid resource management: Model and implementation issues. ISA T., 53(4): 1261-1267.

Liu, H., A. Abraham and A.E. Hassanien, 2010. Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. Future Gener. Comp. Sy., 26(8): 1336-1343.

Liu, K., J. Chen, H. Jin and Y. Yang, 2009. A min-min average algorithm for scheduling transaction-intensive grid workflows. Proceeding of the 7th Australasian Symposium on Grid Computing and e-Research. Wellington, pp: 41-48.

Lublin, U. and D.G. Feitelson, 2003. The workload on parallel supercomputers: modeling the characteristics of rigid jobs. J. Parallel Distr. Com., 63(11): 1105-1122.

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

Meihong, W., Z. Wenhua, M. Wang and W. Zeng, 2010. A comparison of four popular heuristics for task scheduling problem in computational grid. Proceeding of the 6th International Conference on Wireless Communications Networking and Mobile Computing. Chengdu, pp: 1-4.

Menasce, D.A., D. Saha, S.C.D.S. Porto, V.A.F. Almeida and S.K. Tripathi, 1995. Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures. J. Parallel Distr. Com., 28(1): 1-18.

Michalewicz, Z., 1996. Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, Heidelberg.

Michalewicz, Z., 1999. Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, New York.

Michelakos, I., N. Mallios, E. Papageorgiou and M. Vassilakopoulos, 2011. Ant Colony Optimization and Data Mining. In: Bessis, N. and F. Xhafa (Eds.), Next Generation Data Technologies for Collective Computational Intelligence. Springer, Heidelberg, pp: 31-60.

Montes, J., A. Sanchez and M.S. Perez, 2012. Riding out the storm: How to deal with the complexity of grid and cloud management. J. Grid Comput., 10(3): 349-366.

Murugavel, S.S., S.S. Vadhiyar and R.S. Nanjundiah, 2011. Adaptive executions of multi-physics coupled applications on batch grids. J. Grid Comput., 9(4): 455-478.

Nayak, S.K., S.K. Padhy, S.P. Panigrahi, S. Kumari and S. Prasada, 2012. A novel algorithm for dynamic task scheduling. Future Gener. Comp. Sy., 28(5): 709-717.

Neto, R.F.T. and M.G. Filho, 2013. Literature review regarding ant colony optimization applied to scheduling problems: Guidelines for implementation and directions for future research. Eng. Appl. Artif. Intell., 26(1): 150-161.

Nithya, L.M., T. Nadu and A. Shanmugam, 2011. Scheduling in computational grid with a new hybrid ant colony optimization algorithm. Eur. J. Sci. Res., 62(2): 273-281.

Prajapati, H.B. and V.A. Shah, 2014. Scheduling in grid computing environment. Proceeding of the 4th International Conference on Advanced Computing and Communication Technologies. Rohtak, pp: 315-324.

Qu, Y., C. Lin, Y. Li and Z. Shan, 2005. Survivability analysis of grid resource management system topology. Proceeding of the 4th International Conference on Grid and Cooperative Computing. Beijing, pp: 738-743.

Qureshi, M.B., M.M. Dehnavi, N. Min-Allah, M.S. Qureshi, H. Hussain, I. Rentifis and A.Y. Zomaya, 2014. Survey on grid resource allocation mechanisms. J. Grid Comput., 12(2): 399-441.

Rajni and I. Chana, 2013. Bacterial foraging based hyper-heuristic for resource scheduling in grid computing. Future Gener. Comp. Sy., 29(3): 751-762.

Richard, M., I. Habib, A. Anjum, K. Munir, A. Branson, P. Bloodsworth and S.L. Kiani, 2013. Intelligent grid enabled services for neuroimaging analysis. J. Neurocomput., 122(1): 88-99.

Ritchie, G. and J. Levine, 2004. A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments. Proceeding of the 23rd Workshop of the UK Planning and Scheduling Special Interest Group. Cork, pp: 1-7.

Sakamoto, H., 2007. Data grid deployment for high energy physics in Japan. Comput. Phys. Commun., 177(1-2): 239-242.

Siddiqui, M. and T. Fahringer, 2005. GridARM: Askalon's grid resource management system. Proceeding of the European Grid Conference. Amsterdam, pp: 122-131.

Silva, D.P. da, W. Cirne and F.V. Brasileiro, 2003. Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. Proceeding of the 9th International Euro-Par Conference on Parallel Processing. Klagenfurt, pp: 169-180.

Sivanandam, S.N. and S.N. Deepa, 2008. Introduction to Genetic Algorithms. Vol. 2. Springer, Heidelberg.

Venugopal, S., R. Buyya and K. Ramamohanarao, 2006. A taxonomy of data grids for distributed data sharing, management and processing. ACM Comput. Surv., 38(1): 3-es.

Wang, J., P. Korambath, S. Kim, S. Johnson, K. Jin, D. Crawl and K.N. Houk, 2011. Facilitating e-Science Discovery Using Scientific Workflows on the Grid. In: Yang, X., L. Wang and W. Jie (Eds.), Guide to e-Science. Springer, London, pp: 353-382.

Wang, Y., J. Zhang, Y. Zhao, J. Wang and W. Gu, 2013. ACO-based routing and spectrum allocation in flexible bandwidth networks. Photonic Netw. Commun., 25(3): 135-143.

Wegener, D., T. Sengstag, S. Sfakianakis, S. Ruping and A. Assi, 2009. GridR: An R-based tool for scientific data analysis in grid environments. Future Gener. Comp. Sy., 25(4): 481-488.

Xhafa, F. and A. Abraham, 2008. Meta-Heuristics for Grid Scheduling Problems. In: Xhafa, F. and A. Abraham (Eds.), Metaheuristics for Scheduling in Distributed Computing Environments. Heidelberg, Springer, pp: 1-37.

Xhafa, F. and B. Duran, 2008. Parallel Memetic Algorithms for Independent Job Scheduling in Computational Grids. In: Cotta, C. and J. van Hemert (Eds.), Recent Advances in Evolutionary Computation for Combinatorial Optimization. Springer, Heidelberg, pp: 219-239.

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

Xhafa, F. and A. Abraham, 2010. Computational models and heuristic methods for grid scheduling problems. Future Gener. Comp. Sy., 26(4): 608-621.

Xhafa, F., J. Carretero, L. Barolli and A. Durresi, 2007a. Requirements for an event-based simulation package for grid systems. J. Interconnect. Netw., 8(2): 163-178.

Xhafa, F., L. Barolli and A. Durresi, 2007b. An experimental study on genetic algorithms for resource allocation on grid systems. J. Interconnect. Netw., 8(4): 427-443.

Xhafa, F., J. Carretero, B.B. Dorronsoro and E. Alba, 2009a. A tabu search algorithm for scheduling independent jobs in computational grids. Comput. Inform., 28(2): 237-250.

Xhafa, F., J.A. Gonzalez, K.P. Dahal and A. Abraham, 2009b. A GA(TS) hybrid algorithm for scheduling in computational grids. Proceeding of the 4th International Conference on Hybrid Artificial Intelligence Systems. Salamanca, pp: 285-292.

Xhafa, F., B. Duran and J. Kolodziej, 2011a. On exploitation vs exploration of solution space for grid scheduling. Proceeding of the 3rd International Conference on Intelligent Networking and Collaborative Systems. Fukuoka, pp: 164-171.

Xhafa, F., J. Kolodziej, L. Barolli and A. Fundo, 2011b. A GA+TS hybrid algorithm for independent batch scheduling in computational grids. Proceeding of the 14th International Conference on Network-Based Information Systems. Tirana, pp: 229-235.

Xhafa, F., J. Kolodziej, L. Barolli, V. Kolici, R. Miho and M. Takizawa, 2011c. Evaluation of hybridization of GA and TS algorithms for independent batch scheduling in computational grids. Proceeding of the International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. Barcelona, pp: 148-155.

Yan, H.U.I., X. Shen, X. Li and M. Wu, 2005. An improved ant algorithm for job scheduling in grid computing. Proceeding of the 4th International Conference on Machine Learning and Cybernetics. Guangzhou, pp: 2957-2961.

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

Zhao, G., B.A. Bryan, D. King, Z. Luo, E. Wang, U. Bende-Michl and Q. Yu, 2013. Large-scale, high-resolution agricultural systems modeling using a hybrid approach combining grid computing and parallel processing. Environ. Modell. Softw., 41(1): 231-238.