Journal of Optimization in Industrial Engineering 14 (2014) 27-36

Multi-objective and Scalable Heuristic Algorithm for Workflow Task Scheduling in Utility Grids

Vahid Khajehvand[a,*], Hossein Pedram[b], Mostafa Zandieh[c]

[a] *Assistant Professor, Department of Computer Engineering and Information Technology, Qazvin Branch, Islamic Azad University, Qazvin, Iran*

[b] *Associate Professor, Department of Computer Engineering and Information Technology, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran*

[c] *Associate Professor, Department of Industrial Management, Shahid Beheshti University, G.C., Tehran, Iran*

Received 12 September, 2012; Revised 29 January, 2013; Accepted 20 February, 2013

Abstract

To use services transparently in a distributed environment, the Utility Grids develop a cyber-infrastructure. The parameters of the Quality of Service such as the allocation-cost and makespan have to be dealt with in order to schedule workflow application tasks in the Utility Grids. Optimization of both target parameters above is a challenge in a distributed environment and may conflict one another. We, therefore, present a novel heuristic algorithm for scheduling a workflow application on Utility Grids. Our proposed algorithm optimizes the allocation-cost and makespan in a scalable and very low runtime. The results of the wide-spread simulation indicate that the proposed algorithm is scalable against an increase in the application size and task parallelism of the application. The proposed algorithm effectively outperforms the current algorithms in terms of the allocation-cost, makespan and runtime scalability.

Keywords: Utility Grids, Resource Provisioning, Workflow Scheduling, Multi-objective Optimization, Scalability.

1. Introduction

Grid computing is capable of controlling a wide variety of heterogeneous distributed resources to execute computation and data intensive applications. Grid computing has recently been oriented towards pay-as-you-go models. In these models, the resource providers receive fees from the users for presenting computing and data services. That is why; the industry pioneers such as IBM, HP, Intel and SUN with a large share in this business are more inclined toward the grid computing. IBM, for instance exploits "e-business on demand" model, HP exploits "Adaptive enterprise" model and Sun Microsystems apply to "pay-as-you-go" model Garg et al. (2010). To conduct large-scale computations, the shared distributed infrastructures create the grid environment software and hardware resources. These infrastructures proved to be efficient for executing applications in sciences such as astronomy Katz et al. (2005), high energy physics Deelman et al. (2002) and others.

To describe an application in a high level form regardless of the distributed computing environment, the workflow is the most common approach. A workflow is represented in a "Direct Acyclic Graph" (DAG) with nodes and edges representing the tasks and data dependencies between the tasks, respectively. A DAG is defined as $G = (V, E)$, where V is a set of nodes, each node representing a task, and E is a set of links, each link representing the execution precedence order between two tasks. For example, a link $(i, j) \in E$ represents the precedence constraint that task v_i needs to be completed before task v_j starts. As a workflow may consist of sub-workflows with multiple entries and exits so the first thing to be done is to add two pseudo-tasks, a top task and a bottom task, with zero execution time indicated by 0 and $n + 1$, respectively. The top task spawns all actual entry tasks of the workflow to be linked to a single node, while the bottom task joins all actual exit tasks to a single node.

Once an application is transformed into the workflow structure, a workflow management system such as Pegasus Deelman et al. (2005) will be ready to control and manage the execution of workflow on the distributed infrastructure. In these environments, indeed, access to the shared computational resources is carried out through the queue-based Local Resource Management (LRM) system.

A competition develops among users caused by the resources-pricing policies so that users begin being involved in a competition with one another only to gain a resource with an affordable cost and an efficient processing capability. Similarly, resource providers are driven into a competition with one another to sell their idle resource slots to the users in order to gain more profits as well as enhance the resource utilization. The

* Corresponding author E-mail: Khajehvand@qiau.ac.ir

scheduling problem becomes highly complicated and NP-complete Ullman (1975) in such an environment. It is worth noting that the resource consumers and providers are acting independently with conflicting aims. The resource consumers seek the minimum time (makespan) and allocation-cost for scheduling application, whereas the resource providers seek the resource utilization gains. Thus, the users' main challenge in this environment, will be scheduling a workflow application on the heterogeneous resources. In this environment the users have no explicit control on resources to minimize time and allocation-cost.

There is a comprehensive introduction on the job scheduling strategies (Feitelson et al, 1995; Feitelson et al, 1997). Moreover, in Xhafa & Abraham (2010), the computational models are surveyed for Grid scheduling problems and their resolutions using the heuristic and meta-heuristic approaches.

In the queue-based systems, the users submit the tasks to the resource queues, whereas the resource allocation will subsequently be conducted due to the strategy of LRM system. In such systems neither has the user explicit control on the allocating resources to the tasks nor can the user optimize the performance. This delivered quality of service to the users is known as the best effort QoS.

The alternative approach is one of the planning-based systems Hovestadt et al. (2003). In these systems, according to agreements the start time of the task can be established in advance instead of the task waits in queue in order to get access to the resource Czajkowski et al. (2005). The above-mentioned agreements are based on an abstract description, so-called "slot" so that the slots are specified by the start time, the number of available processors, the cost and the duration parameters. In this paper, the planning-based system is exploited as the resource management strategy.

Workflow scheduling algorithms are classified into two main groups: best effort and QoS constraint based scheduling Yu et al. (2008). The first group are further classified into four groups: list scheduling heuristics (Yu et al, 2005; Yu & Buyya, 2006; Falzon & Li 2010), clustering heuristics Yu et al. (2008), task duplication heuristics Yu et al. (2008) and guided random search Yu et al, 2008; Topcuoglu et al, 2002; Falzon & Li 2012). But in QoS-based, there are few works addressing workflow scheduling with QoS. They mainly consider the makespan or execution cost of the workflow as the major QoS attribute. As a result, they are suitable for community Grids, moreover, in utility Grids, there is much potential to study the combinations of QoS attributes. The current methods mostly are not designed with the aim of minimizing the cost and time. Also, the scalability relative to an increase in the workflow size, task parallelism and heterogeneous resources is scarcely considered.

In Rblitz et al. (2004), a resource model is adopted similar to the proposed model so that a client may seek the possible start-times for executing a task on a resource. This model differs from the proposed model so that in this model, the resource provisioning takes place just for one task on a single resource, whereas in the proposed model, the resources are provisioned for the entire application tasks. As the available resources are reported due to the slots in order to find a possible solution, the whole feasible and unfeasible combinations of the resource-slots need to be checked.

There are three classes of approaches to the problem of multi-objective scheduling Abrishami et al. (2012). The first class of the approaches extends the definition of optimality to pareto optimality (Singh et al, 2009; Jeannot et al, 2012). The second one is bi-criteria scheduling approaches, usually limited to optimizing two specific objectives (Abrishami et al, 2012; Jeannot et al, 2012; Dongara et al, 2007). The last one optimizes a linear combination of multiple scheduling objectives with a different weight value assigned to each one of them (Garg et al, 2010; Singh et al, 2009). This last class assumes that the user is able to specify the requirements in such a model.

In Garg et al. (2012), a heuristic algorithm is presented for scheduling many parallel applications on the Utility Grids so that it can manage and optimize the cost-to-time trade-off. This approach is close to the studies conducted for this paper and its main difference from that of the proposed approach lies scheduling the parallel applications, whereas the approach adopted by present paper is based on scheduling the workflow application. Due to the data dependencies among tasks, scheduling the workflow application becomes more complex than scheduling the parallel application.

The main objective of the conventional workflow scheduling is the minimization of the time. A large number of the workflow-based scheduling algorithms rest on the list-scheduling technique (Falzon et al, 2012; Topcuoglu et al, 2002). According to this technique, a rank is typically assigned to each application task, the tasks are, subsequently, sorted and scheduled in a descending order of the corresponding rank. The Heterogeneous Earliest Finish Time (HEFT) algorithm Topcouglu et al, (2002) is one of the most common list-based workflow scheduling algorithms. To obtain the list-scheduling, the HEFT takes the task runtime and the data transfer between the tasks and the heterogeneity of the resources into account. The HEFT schedules the workflow application with a high performance in the heterogeneous environment (Topcouglu et al, 2002; Wieczorek et al, 2005).

In Deelman et al. (2008), a study on the use of the Cloud computing for the scientific workflow explores the cost performance trade-off of the different execution modes and resource provisioning plans. This approach differs from the proposed approach, in that, it examines cost performance trade-offs disregarding the minimization of the multi-objective allocation-cost and makespan.

There is a handful of the different studies conducted on the cost optimization of the workflow scheduling close to the current paper's study. In Singh et al. (2006), the

proposed genetic algorithm finds an optimized mapping of the tasks to the resources minimizing both financial cost and makespan. This approach developed in [19, 25] presents the cost-based model in which the resource providers advertise the available resource slots to the users. To minimize the application makespan under the minimum resource allocation-cost, the presented multi-objective genetic algorithm is capable of provisioning a subset of the resource slots. The main difference between these cost minimization algorithms and our proposed algorithm is that these minimization algorithms rely on a cluster with homogeneous processors. Thus, in (Singh et al, 2009; Singh et al, 2007), the entire resources possess identical CPU ratings and cost processing whereas in the proposed model, all resources are constituted of the heterogeneous clusters with different processing cost and CPU ratings in the real-world Utility Grids environments. Hence, removing this resource homogeneity complicates the identification of an appropriate resource selection.

Since the above-mentioned cost optimization algorithms (Singh et al, 2009; Singh et al, 2007) are genetic-based ones, the runtime takes a longer time. In case, the slots' characteristics undergo a change during scheduling, the slots' characteristics are to be updated and a rescheduled resulting in a far longer runtime. Hence, these approaches do not serve the purpose in the dynamic environments such as the Grids.

This paper deals with developing a Workflow Planning Cost-based (WPC) model in order to effectively schedule an application in the Utility Grids so that the application time and allocation-cost can be minimized. In fact, the WPC model allows the users to make a trade-off between an application time and allocation-cost. Next, a First-fit Cost-Time Trade-off (FCTT) heuristic algorithm is employed to solve the workflow scheduling problem. The FCTT is a heuristic algorithm that schedules an application in a form that both the time and the allocation-cost can be optimized according to the trade-off factor. A trade-off factor shows the preference of the allocation-cost optimization to the turnaround-time. In Khajevand et al. (2012), we presented a preliminary version of the proposed algorithm so that it selects a task with a minimum first fit cost-makespan objective function. However, this paper was not considered the issue of scalability with different workflow sizes, task parallelism and heterogeneous resources. Finally, to study and evaluate the efficiency of the proposed algorithm on the proposed model, a handful of experiments have been conducted and simulated. The main contributions of the paper are as follows:

1) Developing a WPC model based on provisioning the resources for scheduling a workflow, so that the application makespan and allocation-cost can be minimized.
2) Developing a multi-objective FCTT heuristic algorithm based on the WPC model with the following characteristics: (a) the scalability and a better performance due to an increase in the workflow size. (b) the scalability and a better performance according to an increase in the degree of the task parallelism.

The rest of this paper is organized as follows: Section 2 introduces a workflow planning problem and execution environment. A proposed detailed model and heuristic algorithm is described in Section 3. Section 4 involves a simulation setup and its relevant experiments in order to evaluate the efficiency of the proposed algorithm. In Section 5, the results have been analysed statistically. Finally, section 6 ends with a conclusion.

2. Workflow Planning Problem

To execute large-scale applications by developments in computer science, the collaborative use of distributed resources managed by different autonomous domains is made possible. In this environment, the resources available to these applications are shared between multiple users, so the optimization of the throughput or utilization of the resources is of importance. Consequently, the user has no explicit control on the allocation of resources; hence the performance of the application will be unpredictable in advance. Similarly, the resources do not take the users' preferences into consideration. Thus more advanced methods are needed that would allow the resources to differentiate between the users and deliver multiple qualities of service.

Performance optimization for applications in such a distributed environment is a difficult problem. Sometimes, offline methods such as manual negotiation between resources and users are used for allowing users to achieve the desired performance. However, these offline methods are not scalable against an increase in the application size and task parallelism of the application.

The user submits the application characteristics to the application-level scheduler only to be executed on the grid environment. The user expects to have his application executed with the minimal time and allocation-cost. Certainly, the users exploit trade-off factor in order to show a preference for cost to time. In cases where this factor is not specified by the users, the default trade-off factor is considered as equal.

In fact, the application-level scheduler acts as a mediator between the resource providers and users. Due to the reports of the available slots obtained from the resource providers, the application-level scheduler plans the application. The entire slots exploited in planning the application, will be submitted to LRM in order to provision the resources. Each computational resource is equipped with a number of the processors, the memory and the network interfaces showing an independent processing unit. The entire resources are fully-connected while being capable of executing all application-tasks. All of the computational resources can act as a service-provider (site) for time-slots.

The application-tasks will be non-preemptively executed, so that one or a multiple of computational resources are exclusively applied to in order to be executed in due time. We suppose that the application-task performance models are clear on each resource. The execution time of a certain task, therefore, may be obtained from a certain resource due to application performance models. Also, the execution of a single task consists of three phases: (a) the input data retrieval from the resource executing the immediate predecessors of the task (b) the task execution and (c) the output data communication from the current resources to the resources presumed to execute successors of the task.

In this paper, the remote I/O (on demand) strategy Deelman et al. (2008) has been used to transfer the data between the application tasks. According to this strategy, the output data of a task from the resource executing the task is transferred to the resource supposed to execute the successor task. As the application tasks are assumed to be rigid, eventually, processors in need are simultaneously and exclusively handed over to desired task throughout the execution time.

3. Proposed Model and Heuristic Algorithm

In general, the users are in need of two QoS: the deadline and budget of their applications on the pay-per-use services Yu et al. (2005). The users normally tend to run their applications in as the minimum time and cost as possible. Thus, a trade-off factor indicating the significance of the cost to time will be used. In this section, the issue of application scheduling will be stated and the WPC model will be presented and then solved in order to optimize the application cost-time trade-off. Finally, a heuristic algorithm will be developed to conduct the scalable application scheduling with the aim of optimizing the cost and time.

3.1. The Proposed Multi-Objective Cost-Based Model

The execution model consists of a set of heterogeneous consumers and resource providers where the consumers seek to schedule their workflow applications with the minimum cost and time. In this model, R is a set of available heterogeneous resources and V is a set of the tasks of the workflow application. Each resource consists of a set of slots for executing the task v_i.

Services have different processing capabilities which are delivered with different prices. The time(v_{ij}) is the normalized completion time of v_i on the resource r_j and the cost(v_{ij}) is the normalized allocation-cost of v_i on the resource r_j. The normalization matters since it is not clear what value ranges the allocation-cost and finish time will take in a given solution. The value of alpha (α) is a number between 0 and 1 considered as a constant trade-off factor. This trade-off factor shows the degree of the significance of allocation cost to execution finish time. The scheduling optimization problem seeks to generate solution S, mapping every task v_i to a suitable resource r_j to achieve the multi-objective cost-based metric defined by

$$S_{ij} = \alpha \times time(v_{ij}) + (1-\alpha) \times \cos t(v_{ij}),\ \ \forall v_i \in V, \forall r_j \in R \qquad (1)$$

where normalized execution time (time) and normalized allocation-cost (cost) are the normalized versions of Execution Time (ET) and Allocation Cost (AC), so that they are computed by

$$time(v_{ij}) = \frac{ET(v_{ij}) - \min\limits_{\forall r_{j\in R}}\{ET(v_{ij})\}}{\max\limits_{\forall r_{j\in R}}\{ET(v_{ij})\} - \min\limits_{\forall r_{j\in R}}\{ET(v_{ij})\}}, \qquad (2)$$

$$\cos t(v_{ij}) = \frac{AC(v_{ij}) - \min\limits_{\forall r_{j\in R}}\{AC(v_{ij})\}}{\max\limits_{\forall r_{j\in R}}\{AC(v_{ij})\} - \min\limits_{\forall r_{j\in R}}\{AC(v_{ij})\}}. \qquad (3)$$

Thus, the objective function of the application scheduling problem is obtained through minimizing the multi-objective cost-based metrics for the whole application-tasks reached by

$$\min\ (S_{(n+1)j}),\ v_{n+1} \in V,\ \forall r_j \in R. \qquad (4)$$

where v_{n+1} is an exit pseudo-task with zero execution time and allocation cost. The application scheduling problem involves mapping each task v_i to the suitable slot of the resource r_j, so that the application makespan and allocation-cost can be minimized. Upon the completion of the whole application tasks, makespan and allocation-cost will be computed. In the following section, a WPC-based heuristic algorithm is presented to solve the workflow scheduling problem as a whole.

3.2. The Proposed Heuristic Algorithm

The FCTT is an algorithm selecting the most appropriate slots for each task according to the list-scheduling ready to be executed. According to the HEFT algorithm topcouglu et al. (2002), firstly, the list-scheduling is obtained by the act of assigning a rank to each task in the workflow using a bottom-up traversal, so that, the child tasks are the first ones to be assigned a rank prior to the parent tasks assignment Yu et al. (2005). The rank of each task v_i is defined by

$$Rank(v_i) = \begin{cases} w_i & if\ succ(v_i) = \varnothing \\ w_i + \max\limits_{v_k \in succ(v_i)} \left(t_{ik} + Rank(v_k)\right) & otherwise \end{cases}, \qquad (5)$$

where w_i is the average runtime of the task v_i on all the resources in the system and t_{ik} is the average data-transfer time between tasks v_i and v_k using the average bandwidth and latency time between the resources in the system. This task ranking system is derived from the HEFT algorithm Yu et al. (2005). Then, the tasks are sorted by ranking in a non-increasing manner that also ensures a

topological sort. Eventually, the FCTT algorithm selects a task whose execution of all parents' tasks is completed from the obtained list scheduling in order to schedule the task.

There are many choices for each task, only the choice capable of minimizing the multi-objective cost metric of Eq. (1) will be selected as the best solution. According to the best solution, the Earliest Start Time (EST) needs to be computed to execute immediate successor tasks and this procedure will be carried on so long as the execution of the whole application tasks will be finished.

The FCTT pseudo-code is presented in algorithm 1 operating according to the WPC model. The FCTT algorithm obtains the list-scheduling, application characteristic and the available list of the slots of all resources as an input parameter (lines 1, 2). Moreover, the EST is initialized with simulation current time (line 3). The application-level scheduler carries out the planning of each application task due to available slots list characteristics with an eye on the multi-objective cost metric presented in Eq. (1), (lines 4 to 14). Initially, a list of unplanned tasks eligible to be executed is selected (line 5). Next, the eligible tasks are defined as the ones whose parents' tasks execution is completed. These very same tasks have not been executed yet. The available slots list of each resource is obtained by line 7. In line 8, the EST of the task T on all the resources is computed. Eventually, the Earliest Finish Time (EFT) of the task T is computed, (line 9).

The EST is computed on the basis of the completion-time of the latest parents tasks T. Next, the best slot capable of executing the task is selected for each task T on each resource. In cases, the selected resource does not match with the resource executing the parents' tasks, the data-transfer time needs to be added to the EST.

Once the best slot to execute task T is obtained on each resource, the resource minimizing the multi-objective cost metric in Eq. (1) will be selected as the best resource (line 10). Now, it comes to allocating the task T to a selected resource (line 11) as well as updating the slots list of the selected resource (line 12). This procedure needs to be continued as long as there still exists an eligible task (lines 4 to 14). At the end of the completion of the whole application tasks, the slots assigned to the application tasks will be released.

4. Simulation Setup

To conduct an experimental evaluation of the efficiency of algorithm 1, the GridSim [28, 29] is used to simulate the application-level scheduler in the Utility Grids environment. The Grids environment modelled in this simulation consists of ten sites. These sites belong to a subset of the European Data Grid (EDG) spread across five interconnected countries via a high-speed network [1, 30]. The characteristics of the resources are shown in table 1. The mean bandwidth value of the resources is 10 Gb/s with a mean latency time of 150 s.

The workload simulated on these sites follows the workload model generated by Lublin Lublin et al. (2003). The main purpose of the use of this model is to create a realistic simulation environment where the tasks compete with one another. Table 2 shows the workload parameters values applied to in the Lublin model.

To conduct experiments, a parameterized graph generator is used to create a synthetic workflow application Topcouglu et al. (2002). Table 3 shows the workflow application characteristics as in Singh et al. (2009).

At this stage, the scheduling algorithm using the best-effort QoS for scheduling, is simulated and tagged as the BE. As the number of the resources is m and the resources are heterogeneous in terms of CPU rating and allocating-cost, a heuristic algorithm needs to be taken into account to select a suitable resource in the best-effort QoS. In BE, the exploited heuristic method selects a resource with the minimum number of tasks in the waiting and running queues. The majority of the resource management systems make it possible for users to obtain the number of the tasks in the waiting and running queues Singh et al. (2009).

For our experiments, we use the GridSim to simulate benchmark algorithms, testbed platform environment and workload on an Intel Core 2 Duo CPU T9600, 2.80 GHz computer.

Algorithm 1: The pseudo-code for the FCTT algorithm	
Input:	A list-scheduling obtained from the HEFT algorithm An application characteristics The resource characteristics and the available slots to each resource
Output:	The workflow scheduling
1	Get the list of the available time slots for all resources
2	*UnScheduledTask* = get the list of the tasks which is sorted by the rank obtained from the HEFT algorithm due to Eq. (5).
3	Assign the simulation current time to the *EST*.
4	**While** *UnScheduledTask* is not empty **do**
5	*EligibleTasks* = select all tasks which executions of their parents have been completed.
6	**for** each *T* in the *EligibleTasks* **do**
7	Acquire the available slots of each resource.
8	Compute the *EST* of the task T on each resource.
9	Compute the Earliest Finish Time (*EFT)* of the task T according to best EST.
10	Find a Time Slot (*TS*) which is feasible for the task *T* while minimizing the multi-objective cost-based metrics defined by Eq. (1).
11	Allocate the task *T* to the *TS* on the resource *r*.
12	Update the list of available slots to the resource *r*.
13	**end for**
14	**end while**
15	Compute the makespan and allocation-cost of the application.

Table 1
Simulated EDG testbed resources

Site name (Location)	Number of CPUs	Single CPU rating(MIPS)	Processing cost(G$)
RAL(UK)	20	1140	0.0061
Imperial College(UK)	26	1330	0.1799
NorduGrid(Norway)	265	1176	0.0627
NIKHEF(Netherlands)	54	1166	0.0353
Lyon(France)	60	1320	0.1424
Milano(Italy)	135	1000	0.0024
Torina(Italy)	200	1330	1.856
Catania(Italy)	252	1200	0.1267
Padova(Italy)	65	1000	0.0032
Bologna(Italy)	100	1140	0.0069

Table 2
Lublin workload model parameter values

Workload parameter	Value
JobType	Batch Jobs
Maximum number of CPUs required by a job(p)	1000
uHi	$Log_2(p)$
uMed	uHi-1.5
Other parameters	As created by Lublin model

Table 3
Workflow application characteristics

Workflow characteristics	Value
The number of tasks in the workflow application (n)	100
The average runtime of each task	1000 sec
The average number of processors per task	25
The average depth of the workflow graph	$\sqrt{n}$
The average number of tasks per level	$\sqrt{n}$
The mean value of data transfer among the tasks	1000 *Gb*

Table 4
The pattern of the test problems

Test problems	I	J	K
p1	1	100	10
p2	25	25	10
p3	25	50	10
p4	25	100	10
p5	50	100	10
p6	75	100	10
p7	100	100	10
p8	25	200	10
p9	25	300	10
p10	25	500	10

An application scheduling algorithm using cost model is presented by Singh (Singh et al, 2009; Singh et al, 2007). Their algorithm has provisioned a set of the slots to optimize performance under the minimum allocation-cost in order to execute application on the provisioned slots. This cost-modelled algorithm makes a trade-off between scheduling and allocation-cost based on trade-off factor. After that, the scheduling takes place using a multi-objective genetic algorithm Fonseca et al. (1993), as well as simulating the algorithm. It is tagged as the MOGA for brevity Singh (Singh et al, 2009; Singh et al, 2007).

The FCTT, the MOGA and the BE algorithms are simulated and their performance is evaluated through conducting a number of experiments. Finally, the results from the algorithms are compared with one another. In the next section, simulation the compared results will be thoroughly analysed.

5. Analysis of Results

In this subsection, first test problems are generated in Table 4. In this table, 10 test problems are generated randomly. These problems are categorized based on the number of required processors (I), the number of tasks (J) and the number of available resources (K). Each test problem is employed ten times and the average solution values are obtained and used for performance evaluations. In fact, Table 4 contains different values of these parameters. In this table, according to the presented characteristics in the section 3, a synthetic workflow application is generated considering trade-off factor equal to 0.5.

Then, the performance of the algorithms is studied on numerous test problems through different metrics, including makespan, allocation-cost, and runtime. The proposed algorithm is called FCTT algorithm and is compared with the MOGA and the BE algorithms Singh (Singh et al, 2009; Singh et al, 2007).. To do so, Table 5 presents the outputs of the algorithms on different considered metrics. In this table, for each test problem, the first column show name of the test problems and the rest columns present the outputs of the algorithms on the metrics.

As the data are not of homogeneity of variance and there is a small number of instances, so the parametric test is not suitable. To compare the performance of algorithms, therefore, we used a nonparametric test known as Kruskal-Wallis. Table 6 statistically investigates obtained information of Tables 5 (Hedges et al, 1985; Wolfe et al, 1973). Obviously, in all of the statistical tests the significant level is set as 0.05.

Besides, some figures are illustrated to show the performance of the algorithms more explicitly. The first group of these figures, from Fig.1 to Fig. 3, summarized the manner outputs of the algorithms on the test problems for all metrics graphically. In this figures, the Y-axis is drawn in logarithmic scale to make the experiments results discernible. Another group of the figures, from Fig.4 to Fig.6, is dedicated to the visualization of the statistical tests via boxplots of the outputs of the algorithms on the metrics.

Now, according to these test problems and the three considered metrics the proposed algorithm is compared with the two other considered algorithms.

In last row of Table 5, the outputs of the algorithms on the test problems are summarized via an average metric. According to this average values, it is clear that for all metrics, the proposed algorithms is superior. Moreover, to have a better sense of performance of the algorithms, Fig. 1 to Fig.3 are plotted for makespan, allocation-cost and runtime, respectively. According to these figures, on the makespan and allocation-cost FCTT and MOGA have close manners and outperforms BE. However, in runtime the proposed algorithm dominates the two other algorithms considerably.

Besides non-statistical test, we need to have a statistical test for comparing the algorithms. Table 6 shows a statistical comparison of the algorithms. This table presents the *P*-values of the tests. In the statistical difference to the MOGA algorithm hypothesis tests of this table, like any other hypothesis tests, *P*-values are compared with the level of 0.05 significance. In case the null hypothesis is rejected (*P*-value of the test becomes less than our considered significant level), the boxplot will be assessed and the superior algorithm will be determined. The MOGA algorithm is proved to be better than BE algorithm [19, 25]. In order to prove that the proposed approach is more effective than the MOGA and BE algorithms, we performed a Kruskal-Wallis test between metrics of the MOGA and proposed algorithms. Table 7 shows the results of this test. Due to the table, the proposed algorithm operates in two metrics- makespan and cost- the same as the MOGA does. However, in terms of runtime, the proposed algorithm shows a significant

Table 5
The outputs of the algorithms on the metrics

Test problems	BE			MOGA			Proposed Approach (FCTT)		
	Makespan	Cost	Runtime	Makespan	Cost	runtime	Makespan	Cost	Runtime
p1	38	185488	67033	16	8273	182380	16	5266	70
p2	199	890175	72421	50	51552	125317	47	29959	43
p3	455	1817690	207761	67	111106	179351	56	87689	71
p4	967	3116929	507615	83	243622	188335	72	170474	83
p5	2573	5944557	3222543	136	324209	169125	135	340184	86
p6	3889	7186587	7245006	352	1250115	178488	319	533121	55
p7	3944	9014009	5834740	352	4163217	163963	406	1188330	72
p8	2081	5200360	2423552	99	457395	266074	85	422030	202
p9	2092	7746812	2460495	108	632793	384885	98	474088	315
p10	4027	10100000	9851479	122	1246753	4233005	98	857269	735
Average	2026.5	5120261	3189265	138.5	848904	607092	133.2	410841	173.2

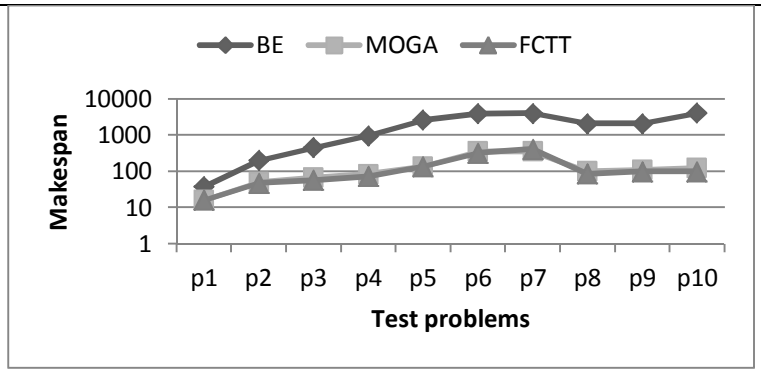

Fig. 1. comparision of the three algorithms on the makespan through test problems

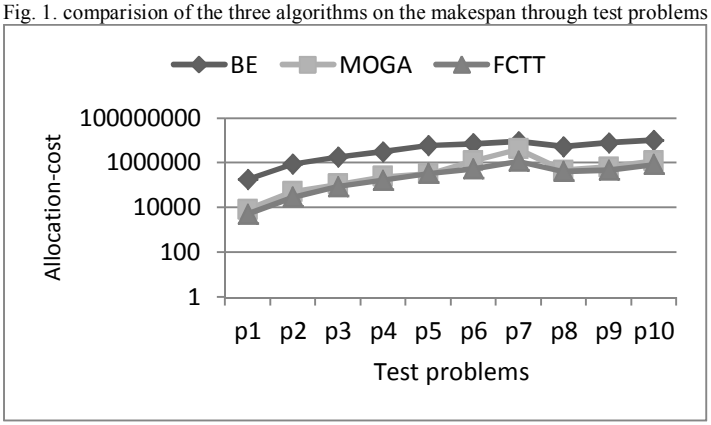

Fig. 2. comparision of the three algorithms on the allocation-cost through test problems

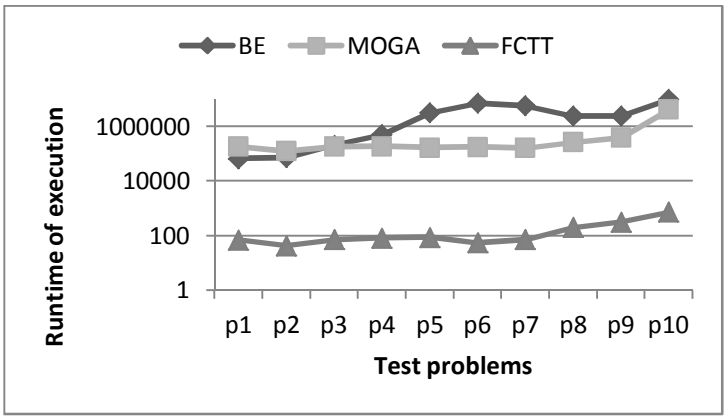

Fig. 3. comparision of the three algorithms on the runtime through test problems

Table 6
Summary of Kruskal-Wallis test for different metrics of three algorithm according to Table 5

Metrics	***Kruskal-Wallis***	
	P-value	Result
Makespan	0.003	H_0 is rejected
Cost	0.002	H_0 is rejected
Runtime	0.000	H_0 is rejected

Table7
Summary of Kruskal-Wallis test for different metrics of the MOGA and FCTT algorithms according to Table 5

Metrics	***Kruskal-Wallis***	
	P-value	Result
Makespan	0.0623	H_1 is rejected
Cost	0.545	H_1 is rejected
Runtime	0.000	H_0 is rejected

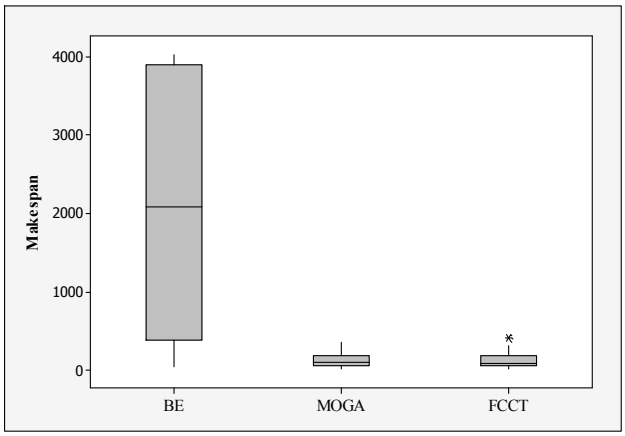

Fig. 4. Makespan boxplot of the three algorithms on the test problems

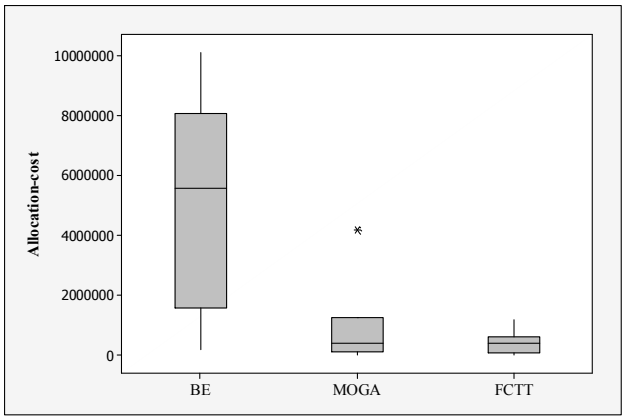

Fig. 5. Allocation-cost boxplot of the three algorithms on the test problems

6. Conclusion

The present paper involves designing, implementing and evaluating the FCTT heuristic algorithm for scheduling a workflow application. The paper seeks to optimize the multi-objective cost-time based on the proposed WPC model. To develop a real distributed environment, the resources workload is simulated based on the Lublin model. Due to many experiments conducted on a generated syntactic workflow, it was shown that the FCTT heuristic algorithm is far more effective than the

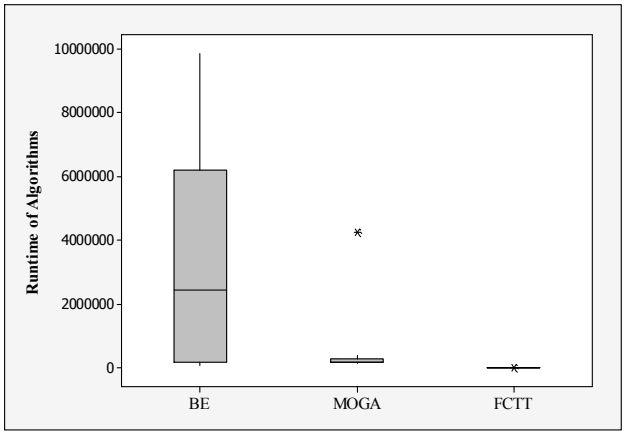

Fig. 6. Runtime boxplot of the three algorithms on the test problems

existing algorithms in terms of the cost-time optimization and scalability for scheduling the workflow application.

To determine the impact of the workflow size and task size on the allocation-cost, makespan and runtime in terms of the number of the application tasks, in this paper, a few experiments were conducted. Next, it is followed by an analysis of a comparison between the FCTT, MOGA and BE algorithms. This comparison includes both environments, the statistical and the non-statistical test. Moreover, in the statistical part, a non-parametric test is conducted.

As a result, it is shown the FCTT algorithm is scalable due to an increase in the workflow tasks as well as capable of scheduling huge workflow applications with the lowest runtime in the heterogeneous environment. However, in terms of runtime, the proposed algorithm shows a significant difference to the existing algorithms.

References

[1] Abrishami, S. Naghibzadeh, M. and Epema, D. H. (2012), "Cost-driven scheduling of grid workflows using partial critical paths," Parallel and Distributed Systems, IEEE Transactions on, vol. 23, pp. 1400-1414.

[2] Buyya R. and Murshed, M. (2002), "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," Concurrency and Computation: Practice and Experience, vol. 14, pp. 1175-1220.

[3] Czajkowski, K. Foster, I. and Kesselman, C. (2005), "Agreement-based resource management," Proceedings of the IEEE, vol. 93, pp. 631-643.

[4] Deelman, E. Kesselman, C. Mehta, G. Meshkat, L. Pearlman, L. Blackburn, K. et al., (2002), "GriPhyN and LIGO, building a virtual data grid for gravitational wave scientists," in 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11), Edinburgh, Scotland, UK.

[5] Deelman, E. Singh, G. Su, M. H. Blythe, J. Gil, Y. Kesselman, C., et al., (2005), "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," Scientific Programming, vol. 13, pp. 219-237.

[6] Deelman, E., Singh, G., Livny, M., Berriman, B. and Good, J. (2008), "The cost of doing science on the cloud: the montage example," in Proceedings of the 2008 ACM/IEEE conference on Supercomputing, NJ, USA , pp. 1-12.

[7] Dongarra, J. J. Jeannot, E. Saule, E. and Shi, Z. (2007), "Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems," in Proceedings of

the nineteenth annual ACM symposium on Parallel algorithms and architectures, pp. 280-288.

[8] Falzon, G. and Li, M. (2012), "Enhancing genetic algorithms for dependent job scheduling in grid computing environments," The Journal of Supercomputing, vol. 62, pp. 290-314.

[9] Falzon, G. and Li, M., (2012), "Enhancing list scheduling heuristics for dependent job scheduling in grid computing environments," The Journal of Supercomputing, vol. 59, pp. 104-130.

[10] Feitelson, D. and Rudolph, L. (1995), "Parallel job scheduling: Issues and approaches," in 1st Workshop on Job Scheduling Strategies for Parallel Processing, Santa Barbara, CA, pp. 1-18.

[11] Feitelson, D. Rudolph, L. Schwiegelshohn, U. Sevcik, K. andWong, P. (1997), "Theory and practice in parallel job scheduling," in 3rd Workshop on Job Scheduling Strategies for Parallel Processing, Geneva, Switzerland, pp. 1-34.

[12] Fonseca, C. M. and Fleming, P. J. (2005), "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in Proceedings of the 5th International Conference on Genetic Algorithms, Urbana-Champaign, IL, USA, 1993, pp. 416–423.

[13] Garg, S. K. Buyya, R. andSiegel, H. J., (2010), "Time and cost trade-off management for scheduling parallel applications on Utility Grids," Future Generation Computer Systems, vol. 26, pp. 1344-1355.

[14] Hedges, L. V. Olkin, I. and Statistiker, M. (1985), "Statistical methods for meta-analysis," ed: Academic Press New York.

[15] Hoschek, W. Jaen-Martinez, J., Samar, A., Stockinger, H. and Stockinger, K. (2000), "Data management in an international data grid project," in Grid Computing - GRID 2000: First IEEE/ACM International Workshop, Bangalore, India, , pp. 333-361.

[16] Hovestadt, M. Kao, O. Keller, A. and Streit, A. (2003), "Scheduling in HPC resource management systems: Queuing vs. planning," in 9th Workshop on Job Scheduling Strategies for Parallel Processing, Seattle, WA, pp. 1-20.

[17] Jeannot, E. Saule, E. and Trystram, D. (2012), "Optimizing performance and reliability on heterogeneous parallel systems: Approximation algorithms and heuristics," Journal of Parallel and Distributed computing, vol. 72, pp. 268-280.

[18] Katz, D. S. acob, J. C. J., Berriman, G. B. Good, J. Laity, A. C., Deelman, E. et al., (2005), "A comparison of two methods for building astronomical image mosaics on a grid," in proceedings of the 34th International Conference on Parallel Processing Workshops (ICPP 2005 Workshops), Oslo, Norway.

[19] Khajevand, V. Pedram, H. and Zandieh, M. (2012), "Provisioning-Based Resource Management for Effective Workflow Scheduling on Utility Grids," in Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on, Ottawa, Canada, pp. 719-720.

[20] Lublin, U. and Feitelson, D. G. (2003), "The workload on parallel supercomputers: modeling the characteristics of rigid jobs," Journal of Parallel and Distributed Computing, vol. 63, pp. 1105-1122.

[21] Rblitz, T. Schintke, F. and Wendler, J. (2004), "Elastic Grid reservations with user-defined optimization policies," in Proceedings of the Workshop on Adaptive Grid Middleware, Antibes Juan-les-Pins, France.

[22] Singh, G., Kesselman, C. and Deelman, E. (2006), "Application-level resource provisioning on the grid," in E-SCIENCE '06 Proceedings of the Second IEEE International Conference on e-Science and Grid Computing Amsterdam, The Netherlands, pp. 83-83.

[23] Singh, G. Kesselman, C. and Deelman, E. (2007), "A provisioning model and its comparison with best-effort for performance-cost optimization in grids," in Proceedings of the 16th international symposium on High performance distributed computing, Monterey, CA, USA, pp. 117-126.

[24] Singh, G. Kesselman, C. and Deelman, E. (2009), "An end-to-end framework for provisioning-based resource and application management," Systems Journal, IEEE, vol. 3, pp. 25-48.

[25] Sulistio, A. Cibej, U. Venugopal, S. Robic, B. and Buyya, R. (2008), "A toolkit for modelling and simulating data Grids: an extension to GridSim," Concurrency and Computation: Practice and Experience, vol. 20, pp. 1591-1609.

[26] Topcuoglu, H., Hariri, S., and Wu, M., (2002), "Performance-effective and low-complexity task scheduling for heterogeneous computing," IEEE Transactions on Parallel and Distributed Systems, vol. 13, pp. 260-274.

[27] Ullman, J. D. (1975), "NP-complete scheduling problems, "Journal of Computer and System Sciences, vol. 10, pp. 384-393.

[28] Wieczorek, M. Prodan, R. andFahringer, T. (2005), "Scheduling of scientific workflows in the ASKALON grid environment," ACM SIGMOD Record, vol. 34, pp. 56-62.

[29] Wolfe, D. A., Hollander, M., (1973), "Nonparametric statistical methods," Nonparametric statistical methods.

[30] Xhafa, F. and Abraham, A. (2010), "Computational models and heuristic methods for Grid scheduling problems," Future Generation Computer Systems, vol. 26, pp. 608-621.

[31] Yu, J. and Buyya, R., (2005), "A taxonomy of workflow management systems for grid computing," Journal of Grid Computing, vol. 3, pp. 171-200.

[32] Yu, J. and Buyya, R. (2006), "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms," Scientific Programming, vol. 14, pp. 217-230.

[33] Yu, J., Buyya, R. and Ramamohanarao, K. (2008), "Workflow scheduling algorithms for grid computing," Technical Report, Grids-TR-2007-10, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia.

[34] Yu, J. Buyya, R. and Tham, C. K. (2005), "Cost-based scheduling of scientific workflow application on utility grids," in First International Conference on e-Science and Grid Technologies (e-Science'05), Melbourne, Australia, pp. 140-147..