Multi-objective Heuristic for Workflow Scheduling on Grids

Vahid Khajehvand[1], **Hossein Pedram**[2], **and Mostafa Zandieh**[3]
[1]Department of Computer Engineering and Information Technology, Qazvin Branch, Islamic Azad University, Qazvin, Iran
[2]Department of Computer Engineering and Information Technology, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran
[3]Department of Industrial Management, Shahid Beheshti University, G.C., Tehran, Iran

Abstract—*The Utility Grids develop a cyber-infrastructure for using services transparently in a distributed environment. The parameters of the Quality of Service such as the allocation-cost and turnaround time, needs to be taken care of for scheduling a workflow application in the Utility Grids. These target parameters are sometimes likely to be in conflict. In this paper, a multi-objective cost-based model along with a heuristic algorithm is presented for scheduling a workflow application in order to optimize the multi-objective allocation-cost and makespan in a very low runtime. The results of the wide-spread simulation indicate that the proposed algorithm is effective against an increase in the application size. The proposed algorithm effectively outperforms the current algorithms in terms of the allocation-cost, makespan and runtime scalability.*

Keyword*:* Utility Grids; Application Scheduling; Multi-objective Optimization.

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. Shared distributed infrastructures come up with the grid environment software and hardware resources, in order to conduct large-scale computations. These infrastructures turned out to be efficient for executing applications in sciences such as astronomy [1], high energy physics [2] and others.

The challenge faced by the scientists in these fields is how to use cyber-infrastructure for transferring knowledge from the scientific environments to the distributed computing environments. The workflow is the most common approach to describe an application in a high level form regardless of the distributed computing environment. A workflow is represented in a "Direct Acyclic Graph" (DAG) with nodes and edges representing the tasks and data dependencies between the tasks, respectively. Once an application is transformed into the workflow structure, a workflow management system 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.

The grid computing is an interactive environment in which at one end, the users are expecting to receive services for their applications, whereas the resource providers are ready to offer services to the users at the other. The resource providers advertise the available resources set to be planned by the users, brokers and the application-level schedulers who receive fees upon providing services. An environment characterized with the above-mentioned users and service providers is known as Utility Grids. 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 resources to the users in order to gain more profits as well as enhance the resource utilization.

The scheduling problem becomes highly complicated and NP-complete [3] in such an environment due to the different resource consumers and providers so that each side pursues its own profits. 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 main challenge confronted by the users in this environment, will be scheduling an application on the heterogeneous resources in which the users have no explicit control so that both time and allocation-cost can be minimized.

The present 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 makespan and allocation-cost can be minimized. In fact, the WPC model allows the users to make a trade-off between an application makespan and

allocation-cost. Next, a First-fit Cost-Time Trade-off (FCTT) heuristic algorithm is employed to solve the WPC model. The FCTT is a heuristic algorithm that schedules an application in a form that both the makespan and the allocation-cost can be optimized due to the trade-off factor. The trade-off factor shows the preference of the allocation-cost optimization to the turnaround-time. 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 simulation results show that the FCTT algorithm is effective due to an increase in workflow size. The main contributions of the present paper are as follows:

- Developing a WPC model based on provisioning the resources for scheduling a workflow, so that the application makespan and allocation-cost can be minimized.
- Developing a multi-objective FCTT heuristic algorithm based on the WPC model with an effective performance due to an increase in the workflow size.

The rest of this paper is organized as follows: Section 2 discusses the related works. Section 3 introduces an application scheduling problem and execution environment. A proposed detailed model and heuristic algorithm is described in section 4. Section 5 involves a simulation setup and its relevant experiments in order to evaluate the efficiency of the proposed algorithm. In section 6, the results have been analyzed. Finally, section 7 ends with a conclusion.

2 Related works

As shared multiprocessing systems advance, the issue of the application scheduling has been the main concern. To tackle the problem, the providers seek to maximize the utilization of the resources whereas the users seek to minimize turnaround time of the application. There is a comprehensive introduction on the job scheduling strategies [4, 5]. Moreover, in [6], 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 [7]. 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. 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.

In [8], 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. Due 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 [9] 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 [9, 10].

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 [11], a genetic algorithm is proposed to find an optimized mapping of the tasks to the resources which minimizes both financial cost and makespan. This approach is developed in [12, 13] which presents the cost-based model in which the resource providers advertise the available resource slots to the users. A multi-objective genetic algorithm is presented which is capable of provisioning a subset of the resource slots to minimize the application makespan under the minimum resource allocation-cost. The main difference between these cost minimization algorithms and the present paper’s algorithm lies in the fact that these minimization algorithms rely on a cluster with all processors which are homogeneous. Thus, in [12, 13], 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 [12, 13] 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.

3 Application scheduling problem

Workflow execution planning is carried out prior to the workflow execution. It intends to examine users' execution requirements and to generate suitable execution schedules. The formulation of the optimization problem of the workflow execution and execution environment is presented for the workflow planning problem beneath.

The agreement-based resource management allows an application-level scheduler to attain the resources in the desired time. The workflow management system, therefore, ensures access to the desired resources within the agreed time and cost. In the most resources, an abstract-agreed structure is reached between the provider and consumer in terms of available time slots. In clusters, for instance a slot indicates the availability of a number of the related processors, start time, duration and cost. Once a slot is obtained, it can later on be used without an extra interaction between the provider and consumer. For example, a slot on a cluster is likely to be used to execute a workflow consisting of a number of tasks.

A workflow-application is represented in a DAG. 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. The data is a $V*V$ matrix of the communication data, where d_{ij} is the amount of the data required to be transmitted from the task v_i to the task v_j. 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.

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 which reveals 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.

To transfer the data between the application-tasks, three data-management strategies have been proposed by Deelman et al. [14] known as the regular, dynamic cleanup and the remote I/O (on demand). In this paper, the remote I/O (on demand) strategy has been used, so that the output data are submitted to the resource that is seeking to execute immediate the successor-tasks from the immediate predecessor-tasks using the existing high-speed network among the resources. 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.

4 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 [15]. 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 application scheduling with the aim of optimizing the cost and time.

4.1 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 scheduling optimization problem seeks to generate solution S, which maps 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 α is a trade-off factor that indicates a preference of the allocation-cost to the execution time of the workflow scheduling. Thus, the objective function of the application scheduling problem is obtained by the minimization of the sum of the multi-objective cost-based metrics for the whole application-tasks reached by

$$\min\Big(\sum_{\forall v_i \in V} \min_{\forall r_j \in R} S_{ij}\Big). \qquad (2)$$

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 heuristic algorithm is presented to solve the WPC model as a whole.

4.2 The proposed heuristic algorithm

The FCTT is an algorithm which selects the most appropriate slots for each task, which are ready to be executed. There is a handful of choices for each task, among which the choice capable of minimizing the multi-objective cost metric of (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 algorithm pseudo-code is presented in algorithm 1 which operates according to the WPC model. The algorithm obtains the available slot lists to all resources and the unscheduled tasks 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 (1), (lines 4 to 14). Initially, a list of unplanned tasks which are eligible to be executed is selected (line 5). Next, the eligible tasks are defined as the ones whose parents' tasks execution is completed, though the 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 which executes 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 which minimizes the multi-objective cost metric in (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). Finally, when the entire application tasks are planned, the time and allocation-cost need to be computed. At the end of the completion of the whole application tasks, the slots assigned to the application tasks will be released.

5 Simulation setup

To conduct an experimental evaluation of the efficiency of algorithm 1, the GridSim [16] is used to simulate the application-level scheduler in the Utility Grids environment. The Grids environment which is modeled in this simulation consists of ten sites belonging to a subset of the European Data Grid (EDG) spread across five countries which are interconnected via a high-speed network [8, 17]. The workload simulated on these sites follows the workload model generated by Lublin [18]. The main purpose of the use of this model is to create a realistic simulation environment where the tasks compete with one another.

The Lublin workload model determines the arrival-time, the number of required processors and the estimated runtime parameters. This model is derived from the trace of the existing model to do rigid tasks. Table 1 shows the workload parameters values applied to in the Lublin model. Table 2 shows resource configuration on the Grids test-bed in order to simulate the distributed system as well as the cost of using a processor, a CPU rating, the number of CPUs and the site-location of each resource.

This resource configuration is used in order to show the heterogeneity of the execution environment. The entire

resources are simulated using the advance reservation policy and the conservative backfilling policy in order to improve response time. In general, in the real-world, the resource pricing is controlled by different economic factors, thus, the time and allocation-cost minimization is likely to conflict with one another.

To conduct experiments, a parameterized graph generator is used to create a synthetic workflow application [9]. The application characteristics contain n=100 tasks with an average execution time of 1000 *s* [13]. The workflow on the average consists of $\sqrt{n}$ levels (the workflow graph depths) and $\sqrt{n}$ tasks at each level. Each task on the average needs 25 CPUs for executing. The mean value of the data transfer among the tasks is 1000 *Gb*. The mean bandwidth value among resources is 10 *Gb/s* with a mean latency time of 150 *s*.

Algorithm 1: The pseudo-code for the FCTT algorithm

Input: An application characteristics with an instruction length for each task and the required CPUs
The resource characteristics and the available slots to each resource

Output: The workflow scheduling

```
Get the list of the available time slots for all resources
UnScheduledTask = get the list of the tasks which have not been scheduled yet.
Assign the simulation current time to the Earliest Start Time(EST).
While UnScheduledTask is not empty do
    EligibleTasks = select all tasks which executions of their parents have been completed.
    for each T in the EligibleTasks do
        Acquire the available slots of each resource.
        Compute the EST of the task T on each resource.
        Compute the Earliest Finish Time (EFT) of the task T.
        Find a time slot (TS) which is feasible for the task T while minimizing the multi-objective cost-based metrics defined by (1).
        Allocate the TS on the resource r to the task T
        Update the list of available slots to the resource r
    end for
end while
Compute the makespan and allocation-cost of the application.
```

Table 1: Lublin workload model parameter values.

Workload parameter	Value
JobType	Batch JOBS
Maximum number of CPUs required by a job(p)	1000 $Log_2(p)$
uHi	uHi-2.5
uMed	As created by Lublin model
Other parameters	

Table 2: Simulated EDG testbed resources.

Resource 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

At this stage, the scheduling algorithm which uses 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 [13].

An application scheduling algorithm using cost model is presented by Singh et al. [12, 13]. 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-modeled 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 [19], as well as simulating the algorithm. It is tagged as the MOGA for brevity [12, 13].

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 results which are compared will be thoroughly analyzed.

6 Analysis of results

In this section, the application performance results are compared and analyzed with criteria such as the makespan, allocation-cost and runtime of the proposed FCTT algorithm along with the MOGA and the BE algorithms [12, 13]. Also, it will be shown how the proposed heuristic schedules the application through optimizing the makespan and the allocation-cost in the minimum runtime. According to the presented characteristics in the section 3, a synthetic workflow application is generated considering *"trade-off factor=0.5"*. The rest of the simulation parameters is compatible with the setups in the section 5. The Y-axis is drawn in logarithmic scale to make the experiments results discernable.

A few experiments have been conducted to determine the impact of the workflow size on the allocation-cost, makespan and runtime in terms of the number of the application tasks. It is followed by an analysis of the comparison between the FCTT, MOGA and BE algorithms. The experiments were conducted with the application tasks' sizes of 25, 50, 100, 200, 300 and 500 in order to study the impacts on the allocation-cost, makespan and runtime in the application scheduling due to the increasing number of the application tasks.

Figs. 1 and 2 show the impact of the workflow size on the allocation-cost and makespan in the application

scheduling, respectively. As Figs. 1 and 2 indicate, the allocation cost and makespan of the proposed algorithm which the average of all its instances are around 37% and 1% less than the MOGA algorithm, respectively, and one order of magnitude less than the BE algorithm. The low cost and makespan in proposed algorithm is explained by the fact that it selects a slot with the earliest start-time to run the eligible task from the whole existing slots according to the multi-objective cost metric of (1). However, the MOGA algorithm randomly selects a subset of the slots for scheduling the whole tasks. Due to the existing data dependency among tasks, if the execution of an eligible task is postponed, it will result in lengthening the makespan. In the BE algorithm, as long as the executions of the parent tasks are not completed, child-tasks will not be submitted. As the workflow graph depth is $\sqrt{n}$, the higher the number of the tasks n is, the deeper the workflow will be. Eventually, an increase in the workflow graph depth leads to an increase in the number of the times a task needs to wait, causing an increase in the makespan.

Fig. 3 reveals the FCTT, MOGA and BE algorithms' runtime relative to an increase in the number of the application tasks. As the figure shows, the proposed algorithm in all instances is almost three orders of magnitude less than the MOGA and the BE algorithms. The low time-complexity of the proposed algorithm is explained by the fact that it seeks the best slot for a single task just once, while the MOGA algorithm is implemented based on the genetic algorithm. One of the disadvantages of the genetic algorithms is length of their runtime. Moreover, in order to seek a subset of proper slots, the MOGA algorithm needs to repetitively plan the whole chromosomes of each generation of the population so that the best solution of each generation can be selected. The whole process involves a very high time-complexity. Therefore, the higher the number of the application tasks is, the longer the runtime of the algorithm will be. Due to Fig. 3, if the number of the tasks increases from 300 tasks to 500 tasks in the MOGA algorithm, its runtime will increase around one order of magnitude. The BE algorithm employs the best-effort service while neglecting the cost metric optimization. After the executions of all the parent tasks of a single task are completed the execution of the desired task will start which results in a longer runtime.

According to Fig. 3, due to an increase in the application tasks even when it is running 500 tasks the FCTT algorithm requires much lower runtime. The runtime required by the FCTT algorithm is around 0.7 second for 500 tasks to be executed, whereas in the MOGA algorithm, the application runtime takes almost one hour and twenty minutes. As a result, the FCTT algorithm is scalable caused by an increase in the application tasks as well as capable of scheduling huge applications with the lowest runtime in the heterogeneous environment.

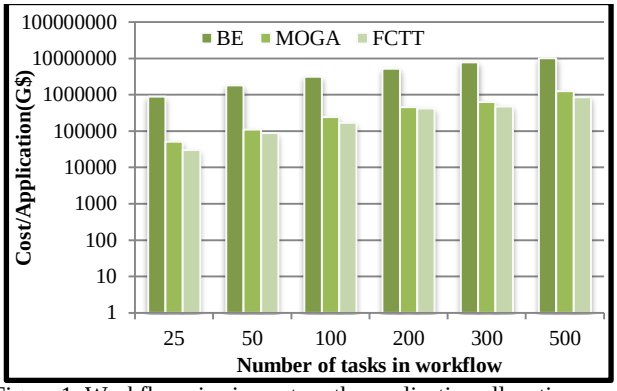

Figure 1. Workflow size impact on the application allocation-cost.

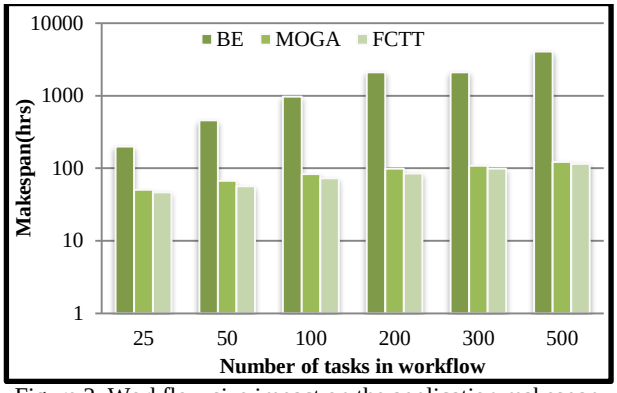

Figure 2. Workflow size impact on the application makespan.

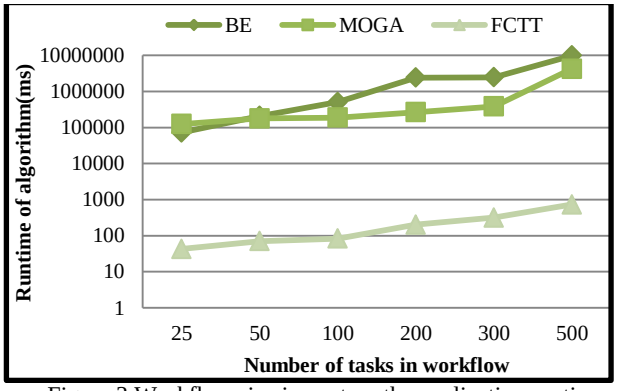

Figure 3.Workflow size impact on the application runtime.

7 Conclusion

The present paper deals with designing, implementing and evaluating the FCTT heuristic algorithm in order to schedule 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 existing algorithms in terms of the cost-time optimization and scalability for scheduling the workflow application.

Also, in this paper, a few experiments have been conducted to determine the impact of the workflow size on the allocation-cost, makespan and runtime in terms of the number of the application tasks. Next, it is followed by an analysis of a comparison between the FCTT, MOGA and BE algorithms. As a result, it was shown the FCTT algorithm is scalable due to an increase in the application tasks as well as capable of scheduling huge applications with the lowest runtime in the heterogeneous environment.

8 References

[1] D. S. Katz, J. C. Jacob, G. B. Berriman, J. Good, A. C. Laity, E. Deelman, C. Kesselman, G. Singh, M. H. Su, and T. A. Prince, "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, 2005.

[2] E. Deelman, C. Kesselman, G. Mehta, L. Meshkat, L. Pearlman, K. Blackburn, P. Ehrens, A. Lazzarini, R. Williams, and S. Koranda, "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, 2002.

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

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

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

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

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

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

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

[10] M. Wieczorek, R. Prodan, and T. Fahringer, "Scheduling of scientific workflows in the ASKALON grid environment," *ACM SIGMOD Record*, vol. 34, pp. 56-62, 2005.

[11] G. Singh, C. Kesselman, and E. Deelman, "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, 2006, pp. 83-83.

[12] G. Singh, C. Kesselman, and E. Deelman, "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, 2007, pp. 117-126.

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

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

[15] J. Yu, R. Buyya, and C. K. Tham, "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, 2005, pp. 140-147.

[16] R. Buyya and M. Murshed, "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, 2002.

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

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

[19] C. M. Fonseca and P. J. Fleming, "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.