Data-Intensive Workflow Scheduling in Cloud on Budget and Deadline Constraints

Zhang Xin, Changze Wu(✉), and Kaigui Wu

College of Computer Science, Chongqing University, Chongqing 400044, China
zx06063068@163.com, {wuchangze,kaiguiwu}@cqu.edu.cn

Abstract. With the development of Cloud Computing, large-scale applications expressed as scientific workflows are often executed in cloud. The problems of workflow scheduling are vital for achieving high efficient and meeting the needs of users in clouds. In order to obtain more cost reduction as well as maintain the quality of service by meeting the deadlines, this paper proposed a novel heuristic, PWHEFT (Path-task Weight Heterogeneous Earliest Finish Time), based on Heterogeneous Earliest Finish Time (HEFT). The criticality of tasks in a workflow and data transmission between resources are considered in PWHEFT while ignored in some other algorithms. The heuristic is evaluated using simulation with five different real world workflow applications. The simulation results show that our proposed scheduling heuristic can significantly improve planning success rate.

Keywords: Workflow · HEFT · Bi-criteria · Data-intensive workflow scheduling

1 Introduction

Large-scale businesses and scientific applications which are usually comprised of big-data, multitasking and multidisciplinary sciences, have required more computing power beyond single machine capability [1]. An easy and popular way is to execute these applications which include scientific workflows, multi-tier web service workflows, and big data processing workflows on the cloud. In order to execute these workflows in reasonable amount of time and acceptable cost, the workflow scheduling problem has been studied extensively over past years.

Workflow scheduling is a process of mapping inter-dependent tasks on the available resources such that workflow application is able to complete its execution within the user's specified Quality of Service (QoS) constraints such as deadline and budget [2]. Workflow scheduling in the cloud faces some challenges. Typically, the non-dedicated nature of resources imposes more difficulties as the contention for shared resources on the cloud needs to be considered during planning. These suggest that the planner may have to somehow query resources for their runtime information (e.g., the existing load) to make informed decisions. And planning should be performed in short time, because users may require a real-time response, and the runtime information, on which a planning decision has been made, varies over time and, thus, a planning decision made using out of date information may not be valid. Moreover, a user may

S. Wang and A. Zhou (Eds.): CollaborateCom 2016, LNICST 201, pp. 262–272, 2017.
DOI: 10.1007/978-3-319-59288-6_24

require his/her workflow application to complete within a certain deadline and budget. However, minimizing makespan and minimizing execution cost are two conflicting objectives. There have been a few bi-criteria DAG planning heuristics in the literature [3–5], some of them have sophisticated designs, such as guided random research or local search, which usually require considerably high planning costs. Moreover, most of these heuristics do not take the data transmission between resources into account, which may lead to an invalid plan in Data-intensive workflow scheduling. Taking computation costs of scheduling into consideration, an efficient algorithm tries to compromise these values (makespan and execution cost) and still obtain approximate optimal solution. In this paper, we aim for bi-criteria scheduling of workflow with the cloud. We address a new heuristic aiming at seeking out a beneficial trade-off between execution time and execution cost under Budget and Deadline constraints. The proposed heuristic, namely, PWHEFT, is based on HEFT [6] algorithm, which maximizes efforts to reduce the overall execution time of a workflow. The HEFT algorithm selects the task with the highest upwards rank value at each step and assigns the selected task to the resource, which minimizes its earliest finish time with an insertion-based approach. While being effective at optimizing makespan, the HEFT algorithm does not consider the monetary cost and budget constraint when making scheduling decisions. Compared with HEFT, PWHEFT figures out appropriate schedule plan after considering Budget, Deadline, criticality of tasks and data transfer rates between processors.

The remaining paper is organized as follows: Sect. 2 presents the related work in the area of workflow scheduling. The problem description is presented in Sect. 3. The proposed heuristic, PWHEFT, is discussed with the help of an example in Sect. 4. The proposed PWHEFT algorithm is evaluated in Sect. 5 and Sect. 6 concludes the paper.

2 Related Work

Due to the NP-complete nature of the parallel task scheduling problem in general cases [7], many heuristics have been proposed in recent researches [8] to deal with this problem, and most of them achieve good performance in polynomial time. In the previous works, the heuristic-based algorithms can be classified into a variety of categories, such as list scheduling algorithms, clustering heuristics, and duplication-based algorithms.

Among them, the list scheduling algorithms are generally more practical, and their performances are better at a lower time complexity. There are different list based heuristic algorithms in literature like Dynamic Critical Path(DCP) [9], Dynamic Level Scheduling (DLS) [10], Critical Path on Processor (CPOP) [6], Heterogeneous Earliest Finish Time (HEFT) [6] etc. From all of these, HEFT outperforms in terms of makespan.

Only few works in the past considered bi-objective (time and cost mainly) criteria to schedule workflow tasks in cloud environment. Recently, Amandeep Verma and Sakshi Kaushal [11] proposed Cost-Time Efficient Scheduling Plan, BDHEFT, which is the extension of HEFT algorithm that schedule workflow tasks over the available cloud resources under Budget and Deadline Constrained. However, this heuristic generates a schedule plan only by considering the spare deadline along with spare budget which are calculated by simple average while selecting the suitable resource for

each workflow task without taking criticality of tasks and parallelism of executing workflow into consideration. To address all these gaps, we introduced in this paper, a novel heuristic that gains a Budget and Deadline Constrained tasks and resources mapper by considering high weights of critical task and the parallelism time efficiency.

3 Models

3.1 Workflow Application Model

A Directed Acyclic Graph (DAG), $G = (T, E)$, is used to model the aforementioned workflow application, where T is the set of n tasks and E is the set of e edges between the tasks. Each edge $(i, j) \in E$ denotes a dependency between two dependent tasks such that the execution of $t_j \in T$ cannot be started before $t_i \in T$ finishes its execution. A task with no parent represents an entry task and a task with no children represents exit task. If there is more than one exit (entry) task, they are connected to a zero-cost pseudo exit (entry) task with zero-cost edges. The task size ($amount_i$) is expressed in Million of Instructions (MI). Data is a n × n matrix of communication data, where $data_{i,j}$ is the amount of data required to be transmitted from task t_i to task t_j.

3.2 Cloud Resources Model

A cloud service provider which offers m computational resources, $R = (r_1 r_2 \cdots r_m)$ at different processing power and different prices, provides information which is needed to make planning decisions. Each r_i is represented by $r_i = (Msr_i, Psr_i)$, where Msr_i denotes Million of Instruction per Second (MIPS) which refers to processing power of a resource r_i and Psr_i denotes the price unit of using resource r_i for each time interval. The data transfer rates between resources are stored in matrix B with size p × p. The communication time between task t_i (scheduled on r_m) and t_j (scheduled on r_k) is defined as:

$$Tran_{(i,j)} = \frac{data_{(i,k)}}{B_{(m,k)}} \tag{1}$$

Before scheduling, average communication time is used to label the edges. The average communication time between task t_i and t_j is defined as

$$\overline{Tran_{(i,j)}} = \frac{data_{(i,k)}}{\bar{B}} \tag{2}$$

where B is the average transfer rate among the resources. Due to each task can be executed on different resources, the execution time, $ET_{(i,j)}$ of a task t_i on a resource r_j, is estimated by the following equation:

$$ET_{(i,j)} = \frac{amount_i}{Msr_j} * (1 + \alpha), \alpha \epsilon [0, 1] \tag{3}$$

where α is random number ranging from 0 to 1, and the execution cost $EC_{(i,j)}$ is given by:

$$EC_{(i,j)} = Psr_i * ET_{(i,j)} \tag{4}$$

Therefore, the average execution time of a task t_i which is defined as

$$\overline{ET_i} = \sum_{j=1}^{m} ET_{(i,j)} \tag{5}$$

and the average execution cost $\overline{EC_i}$ is given by:

$$\overline{EC_i} = \sum_{j=1}^{m} EC_{(i,j)} \tag{6}$$

Although most of the commercial clouds (like Amazon) transfer the internal data at free of cost, the data transfer time cannot be ignored while a large amount of data is needed to be transferred between tasks.

3.3 Scheduling Model

There are three entities in our workflow scheduling model: User, Planner and Cloud Service Provider (CSP). A CSP has a set of computational resources with different capabilities which include processing power and prices and responds to the queries from the planner about the availability of requested resources. The user submits a workflow application along with budget, B and deadline, D to the planner. The planner decides how to execute workflow tasks over available resources.

4 Time and Cost Efficient Scheduling Algorithm

4.1 The PWHEFT

The proposed heuristic, Path-task Weight Heterogeneous Earliest Finish Time (PWHEFT), is based on HEFT, which is a well-known DAG scheduling heuristic. It is an extension of HEFT and considers budget and deadline constraints while scheduling tasks over available resources. PWHEFT has two major phases: task attributes calculation and resource schedule.

First Phase: task attributes calculation

First phase in PWHEFT, the attributes of each task are calculated and all tasks are sorted by priority. The priorities of all tasks are computed using upward ranking. The upward rank of a task t_i is recursively defined by

$$rank_u(t_i) = \overline{ET_i} + \max_{t_j \epsilon succ(t_i)} \left\{ Tran_{(i,j)} + rank_u(t_j) \right\}. \tag{7}$$

where $succ(t_i)$ represents the set of all the children tasks of t_i. Taking into account the criticality level of the task, downward rank is also calculated by:

$$rank_d(t_i) = \max_{t_j \in pred(t_i)} \left\{ Tran_{(j,i)} + rank_d\left(t_j\right) + \overline{ET_j} \right\} \tag{8}$$

where $pred(t_i)$ represents the set of immediate predecessor of t_i. Thus, the criticality level of a task t_i is given by:

$$\mathrm{Clvl}(t_i) = \frac{\left(rank_u(t_i) + rank_d(t_i)\right) * 2}{\min_{t_i \in T}\left\{rank_u(t_i) + rank_d(t_i)\right\} + \max_{t_i \in T}\left\{rank_u(t_i) + rank_d(t_i)\right\}} \tag{9}$$

In order to facilitate the calculation, $EST_{\left(t_i, r_j\right)}$ and $EFT_{\left(t_i, r_j\right)}$ are used to denote the earliest start time and the earliest finish time of task t_i which been scheduled on processor r_j respectively. For the entry task t_{entry}, the EST can be calculated as:

$$EST_{\left(t_{entry}, r_j\right)} = 0 \tag{10}$$

For the other tasks in the graph, starting from the entry task, the EST and EFT values can be calculated as:

$$EST_{\left(t_i, r_j\right)} = \max\left\{ avail[j], \max_{t_m \in pred(t_i)} \left\{ AFT(t_m) + Tran_{(m,i)} \right\} \right\} \tag{11}$$

$$EFT_{\left(t_i, r_j\right)} = EST_{\left(t_i, r_j\right)} + EC_{(i,j)} \tag{12}$$

where $pred(t_i)$ is the set of immediate predecessor tasks of task t_i, $avail[j]$ is the time when the resource r_j is ready for task execution, and $AFT(t_m)$ represents the actual finish time of task t_i. Analogously, $\mathrm{LFT}(t_i)$ which denotes latest finish time of task t_i is calculated by:

$$\mathrm{LFT}(t_i) = \begin{cases} D, when\ t_i = t_{exit} \\ \min_{t_j \in succ(t_i)} \left\{ \mathrm{LFT}\left(t_j\right) - \overline{ET_j} - \overline{Tran_{(i,j)}} \right\}, others \end{cases} \tag{13}$$

where D is given deadline. The schedule length also called makespan is equal to the maximum of actual finish time of the exit task t_{exit}.

$$\mathrm{makespan} = AFT(t_{exit}) \tag{14}$$

Second Phase: resource schedule

In the resource schedule phase, candidate resources are generated and the best resource is selected. For each task which is ordered by $rank_u$, the set of candidate resources is constructed using the six variables: Workflow Prediction Budget (WPB), Prediction Task Budget (PTB), Prediction Budget Factor (PBF), Weight Deadline Factor (WDF), Prediction Deadline Factor (PDF) and Prediction Task Deadline (PTD). For a task t_i, the value of these variables is given by (15) to (20), as follows:

$$\mathrm{WPB} = \mathrm{B} - \sum_{t_i \in allocatedTasks} EC_{(i)} - \sum_{t_i \in unallocatedTasks} \mathrm{Clvl}(t_j) * \overline{EC_j} \tag{15}$$

$$\mathrm{PBF}(t_i) = \begin{cases} 0, when\ WPB < 0 \\ \mathrm{Clvl}(t_i) * \overline{EC_i} / \sum_{t_j \in unallocatedTasks} \mathrm{Clvl}(t_j) * \overline{EC_j}, \text{others} \end{cases} \tag{16}$$

$$\mathrm{PTB}(t_i) = \mathrm{Clvl}(t_i) * \overline{EC_i} + \mathrm{PBF}(t_i) * \mathrm{WPB} \tag{17}$$

$$\mathrm{WDF}(t_i) = \begin{cases} 1, when t_i = t_{exit} \\ \left(\mathrm{Clvl}(t_i) * \max_{t_k \in succ(t_i)} \left\{ Tran_{(i,k)} \right\}\right)^{-1} * \overline{EC_i}, others \end{cases} \tag{18}$$

$$\mathrm{PDF}(t_i) = \frac{\mathrm{WDF}(t_i)}{\sum_{t_k \in unallocatedTasks} \mathrm{WDF}(t_k)} \tag{19}$$

$$\mathrm{PTD}(t_i) = \begin{cases} 0, when\ LFT(t_i) - \min_{r_j \in R} \left\{ EFT_{(t_i, r_j)} \right\} < 0 \\ \min_{r_j \in R} \left\{ EFT_{(t_i, r_j)} \right\} + PDF(t_i) * \left(\mathrm{LFT}(t_i) - \min_{r_j \in R} \left\{ EFT_{(t_i, r_j)} \right\} \right), others \end{cases} \tag{20}$$

where B is given budget, $EC_{(i)}$ is the execution cost of allocated task t_i, $\overline{EC_j}$ *and* $\overline{ET_j}$ are average execution cost and average execution time of un-allocated task t_j. PBF or PDF is a value intended to act as a weight that tunes the impact on PTB or PTD Such prediction function is designed to determine which resources the task t_j PBF or PDF is a value intended to act as a weight that tunes the impact on PTB or PTD Such prediction function is designed to determine which resources the task t_i is predicted to finish on.

Based on the allocated deadline and budget to a task t_i, a candidate set CS_i is calculated by considering possible resources for a task t_i, by:

$$CS_i = \left\{ S_{(i,j)} | \exists S_{(i,j)}, EC_{(i,j)} \leq \mathrm{PTB}(t_i), \mathrm{EFT}_{(t_i, r_j)} \leq \mathrm{PTD}(t_i) \right\} \tag{21}$$

where $S_{(i,j)}$ represents the possible resource, from the given R, which satisfies the inequality .Then the best possible resource is selected by the selection rules as follows:

- I. If $CS_i \neq \emptyset$ then the best resource is selected from this set that minimizes the following expression: $\theta * \mathrm{EFT}_{(t_i, r_j)} + (1 - \theta) * EC_{(i,j)}$ *for all* $j \in CS_i$ where $\mathrm{EFT}_{(t_i, r_j)}$ is the earliest finish time and $EC_{(i,j)}$ is the execution cost of a task t_i over all possible j resources in CS_i respectively and θ is the cost-time balance factor in a range of [0,1] which represents the user preference for execution time and execution cost.
- II.If $CS_i = \emptyset$ and $\mathrm{WPB} \geq 0$, then the resource from all the available resources that minimize the above equation is chosen.

Input: DAG G with Budget B and Deadline D
Output: Schedule Plan

1. Compute the $rank_u$, $rank_d$, Clvl and LFT using equation (7), (8), (9) and (13) for all the tasks.
2. Sort all the tasks in an unallocated list in descending order of upward rank.
3. While there are tasks in unallocated list do
4. Select the first tasks t_i, from the unallocated list.
5. Compute WPB, PTB and PTD for a task t_i using equation (15) to (20).
6. Calculate the candidate set CS_i using equation(21)
7. Select a resource for a task t_i using the defined selection rules.
8. End while

Fig. 1. The PWHEFT heuristic

– III.If $CS_i = \emptyset$ and WPB < 0, the cheapest resource is selected from all the available resources.

The algorithm terminates when all tasks ordered by their rank are considered. The proposed heuristic, PWHEFT is shown in Fig. 1.

4.2 Time Complexity Analysis

To find out the time complexity of PWHEFT algorithm, suppose that the scheduler receives a Workflow, G(T, E) as an input with n tasks and e dependencies. As G is directed graph, so maximum number of dependencies is $(n-1)(n-1)/2 \approx O(n^2)$. The first step of the algorithm is to find two type ranks of all the tasks which requires processing of all workflow tasks and edges, so its time complexity equals $O(n+e) \approx O(n^2)$. Similarly, the step 3, consists of two nested loops. The outer loop is for n tasks and inner loop which refer to calculating the candidate set CS_i is for all the possible m resources. Therefore time complexity of scheduling all tasks is $O(n \cdot m)$. The overall time complexity of PWHEFT algorithm equals $\max(O(n^2), O(n \cdot m))$.

5 Experiments and Analysis

5.1 Experimental Settings

To evaluate the heuristic, the core framework of CloudSim simulator [12] was extended by adding those variables and the heuristic, PWHEFT. And five types of DAGs based on realistic workflows from diverse scientific applications were considered in the experiments, which are:

- LIGO: Gravitational physics
- SIPHT: Biology
- Epigenomics: Biology
- Montage: Astronomy
- CyberShake: Earthquake

The detailed characterization for each workflow including their structure, data and computational requirements can be found in [13]. The Directed Acyclic Graph in XML (DAX) format for all these workflows are available at website (https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator). The cloud environment consisting of a resource provider, which offers 20 different computation resources with different processing speed and hence with different prices, was assumed for the simulation.

The communication computation ratio CCR, which is the ratio of the average of $\overline{Tran_{(i,j)}}$ to the average of $\overline{ET_i}$, is about 2. And the data amount transmitted between tasks is randomly generated according to the CCR. Accordingly, the DAG can be considered as a data-intensive workflow. Given a DAG, constraints for reasonable values for deadline and budget are generated as follows:

$$\boldsymbol{Deadline\,D} = \boldsymbol{Dbound_d} + \boldsymbol{k}_1 * (\boldsymbol{Dbound_u} - \boldsymbol{Dbound_l}),$$

where $\boldsymbol{Dbound_l} = \boldsymbol{M_{HEFT}}$ (makespan of HEFT), $\boldsymbol{Dbound_u} = 3 * \boldsymbol{M_{HEFT}}$ and k_1. is a deadline ratio in range from 0 to 1.

$\boldsymbol{BudgetB} = \boldsymbol{Bbound_d} + \boldsymbol{k}_2 * (\boldsymbol{Bbound_u} - \boldsymbol{Bbound_l})$, where $\boldsymbol{Bbound_l}$ is the lowest cost obtained by mapping each task to the cheapest service and $\boldsymbol{Bbound_u}$ is the highest cost obtained conversely and k_2 is a budget ratio in range from 0 to 1.

After the heuristic was run, if the schedule length and cost of a plan were in conformance with Deadline and Budget, the planning succeeded. To analyze the performance of a heuristic, the experiment was repeated multiple times and the metric Planning Success Rate (PSR), Normalized Schedule Cost (NSC) and Normalized Schedule Length (NSL) were used, as defined below:

$$\mathrm{PSR} = \frac{number\ of\ successful\ plan}{number\ of\ total\ repeated\ times\ of\ experiment} \tag{22}$$

$$\mathrm{NSC} = \frac{total\ cost}{\sum_{t_i \in T} \min_{r_j \in R}\left\{EC_{(i,j)}\right\}} \tag{23}$$

$$\mathrm{NSL} = \frac{Total\ Execution\ Time}{\sum_{t_i \in T} \overline{ET_i}} \tag{24}$$

5.2 Results and Analysis

For comparison purpose, the BDHEFT which is aforementioned and the PWHEFT have been compared on the basis of PSR, makespan and monetary cost. An average value of PSR was captured through 1500 runs of simulations by selecting different values of cost-time balance factor θ, i.e., θ = 0.3, 0.5, and 0.7, each consisting of 500 simulations. Figure 2 shows the PSR, respectively, of scheduling different workflows with BDHEFT and PWHEFT for three different values of deadline ration $\boldsymbol{k}_1$., i.e., $\boldsymbol{k}_1$ = 0.2, 0.4, and 0.8 and three different values of budget ration $\boldsymbol{k}_2$, i.e., $\boldsymbol{k}_2$ = 0.2, 0.4,

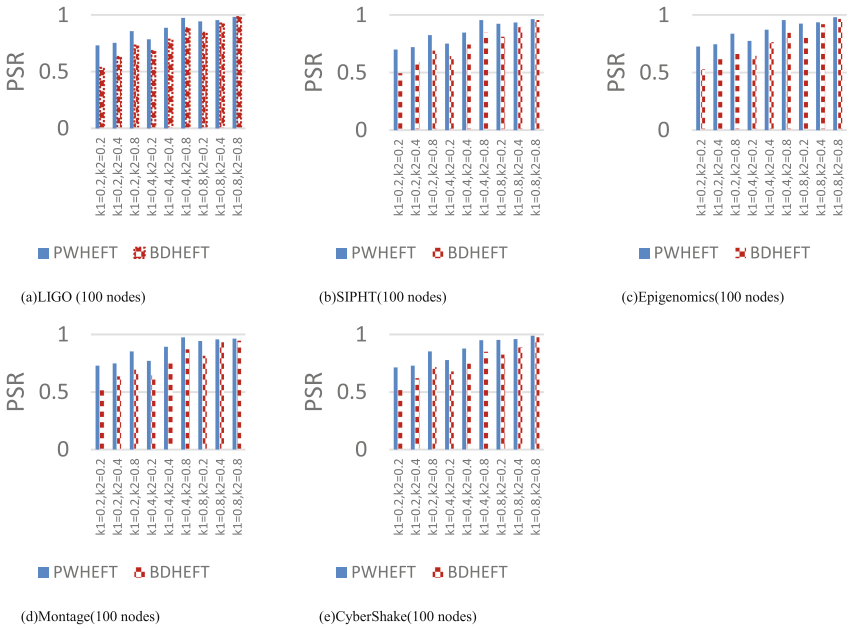

Fig. 2. Average PSR of different workflows

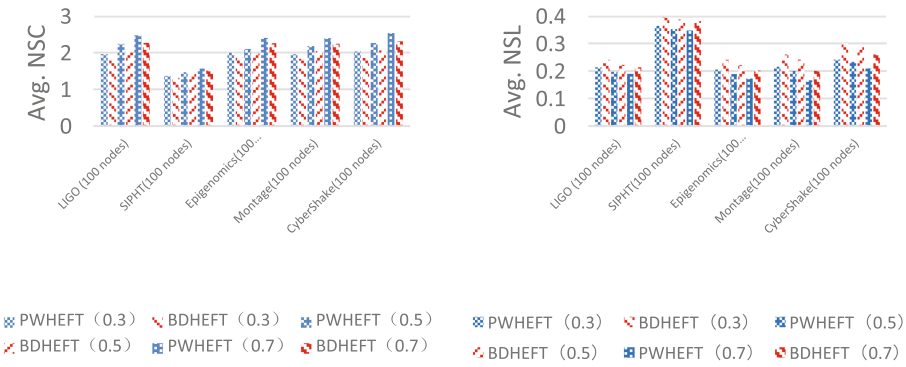

Fig. 3. Average NSC and NSL of different workflows

and 0.8, in total 9 combinations. Even in harsh situations where deadline ration $\boldsymbol{k}_1 = 0.2$ and budget ration $\boldsymbol{k}_2 = 0.2$ as shown in Fig. 2, the average PSR of PWHEFT is significantly higher than BDHEFT.

Figure 3 shows the comparison of execution cost and makespan of schedule plan created by PWHEFT (0.3), PWHEFT (0.5) and PWHEFT (0.7) over schedule plan created by BDHEFT (0.3), BDHEFT (0.5) and BDHEFT (0.7) under the same deadline ratio and budget ratio. The results shows that PWHEFT outperform BDHEFT

Table 1. Comparative results of PWHEFT vs. BDHEFT

Average of PWHEFT over BDHEFT			
Workflow Structure	Cost	Makespan	PSR
LIGO (100 nodes)	+9.08%	−11.39%	+12.95%
SIPHT(100 nodes)	+3.69%	−9.04%	+14.65%
Epigenomics(100 nodes)	+6.29%	−14.65%	+14.07%
Montage(100 nodes)	+6.69%	−17.12%	+15.33%
CyberShake(100 nodes)	+9.71%	−19.56%	+14.24%

algorithm significantly by reducing the makespan while increasing execution cost of schedule slightly under the same cost-time balance factor.

Table 1 shows the overall comparison of execution cost, makespan and PSR of schedule plan created by PWHEFT over schedule plan created by BDHEFT. The overall results shows that PWHEFT algorithm is able to achieve a significant improvement on PSR. Furthermore, PWHEFT has a lower Makespan, while making the execution cost as good as given by BDHEFT under the same deadline and budget constraint and using same pricing model in all cases.

6 Conclusion and Future Works

This paper proposed PWHEFT, a new workflow scheduling algorithm for cloud environment, which is an extension of HEFT algorithm. The proposed heuristic is evaluated with synthetic workflows that are based on real world workflows with different structures and different sizes. The comparison of proposed algorithm is done with BDHEFT heuristic, under same deadline and budget constraint and pricing. The simulation results show that our proposed algorithm outperforms BDHEFT algorithm in terms of PSR and makespan while producing the monetary cost as good as produced by BDHEFT algorithm. Based on the work in this paper, further work could try to examine the performance of PWHEFT using different settings such as diverse CCRs, storage and file transfer model such as a global storage model.

References

1. Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., Vahi, K.: Characterizing and profiling scientific workflows. Futur. Gener. Comput. Syst. **29**(3), 682–692 (2013)
2. Juve, G., Deelman, E., Berriman, G.B., Berman, B.P., Maechling, P.: An evaluation of the cost and performance of scientific workflows on amazon ec2. J. Grid Comput. **10**(1), 5–21 (2012)
3. Prodan, R., Wieczorek, M.: Bi-criteria scheduling of scientific Grid workflows. IEEE Trans. Autom. Sci. Eng. **7**, 364–376 (2010)
4. Talukder, A.K.M., Kirley, M., Buyya, R.: Multiobjective differential evolution for scheduling workflow applications on global Grids. Concurr. Comput. Pract. Exp. **21**(13), 1742–1756 (2009)

5. Yu, J., Buyya, R.: Multi-objective planning for workflow execution on Grids. In: Proceedings of the 8th IEEE/ACM International Conference on Grid Computing, pp. 10–17 (2007)
6. Topcuoglu, H., Hariri, S., Wu, M.: Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. **13**(3), 260–274 (2002)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York (1979)
8. Wu, F., Wu, Q., Tan, Y.: Workflow scheduling in cloud: a survey. J. Supercomput., **71**(9), 3373–3418
9. Kwok, Y.K., Ahmad, I.: Dynamic critical-path scheduling: an effective technique for allocating task graphs to multiprocessors. IEEE Trans. Parallel Distrib. Syst. **7**(5), 506–521 (1996)
10. Sih, G.C., Lee, E.A.: A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures. IEEE Trans. Parallel Distrib. Syst. **4**(2), 175–187 (1993)
11. Verma, A., Kaushal, S.: Cost-Time efficient scheduling plan for executing workflows in the cloud. J. Grid Comput. **13**(4), 1–12 (2015)
12. Rodrigo, N.C., Ranjan, R., Anton, B., Cesar, A.F.D.R., Buyya, R.: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. J. Softw. Pract. Exp. (SPE) **41**(1), 23–50 (2011)
13. Bharathi, S., Lanitchi, A., Deelman, E., Mehta, G., Su, M.H., Vahi, K.: Characterization of scientific workflows. In: Workshop on Workflows in Support of Large Scale Science, CA, USA, pp. 1–10 (2008)