Decomposition Based Multi-objective Work□ow Scheduling for Cloud Environments

Emmanuel Bugingo, Wei Zheng, Dongzhan Zhang, Yingsheng Qin and Zhang Defu

December 12, 2019

Decomposition Based Multi-objective Workflow Scheduling for Cloud Environments.

Emmanuel Bugingo*, Wei Zheng*, Dongzhan Zhang*, Yingsheng Qin*, and Defu Zhang*
*Department of Computer Science
School of Information Science and Engineering
Xiamen University, China
Email:emmanuelbugingo@stu.xmu.edu.cn

***Abstract*—Workflow is a group of tasks that are processed in a particular order to complete an application. Also, it is a popular paradigm used to model complex applications. Executing complex application in a distributed system such as cloud computing implicates optimization of several conflicting objectives such as monetary cost, energy consumption, total execution time of the application (makespan), etc. Regardless of this trend, most of the workflow scheduling approaches focused on single or bi-objectives optimization problem. In this paper, we considered the problem of workflow scheduling in a cloud environment as a multi-objective optimization problem, and hence proposed a multi-objective workflow-scheduling algorithm based on decomposition (WSABD). The proposed algorithm is capable of finding optimal solutions with a single run. Our evaluation results show that, by a single run, the proposed approach manages to obtain the Pareto Front solutions which are at least as good as schedules produced by running a single-objective scheduling algorithm with contraint for multiple times.**

***Index Terms*—Multi-objective optimization, workflow scheduling algorithm, cloud computing.**

I. Introduction

A decade ago, workflow emerged as a popular paradigm to represent big data applications in distributed environments such as the cloud. One of the main objectives of workflow scheduling in a distributed environment is makespan optimization. It is well-known that workflow scheduling in a heterogeneous environment is NP-complete [1]. As cloud computing gain popularity use, makespan optimization is no longer the only one objective to be optimized during workflow scheduling.

There is a rise of other objectives of the same interest as makespan such as cost, energy, reliability, utilization, etc, that need to be taken into consideration during workflow scheduling. Therefore, modern workflow scheduling algorithms for cloud environment must be able to optimize more than one objective at the same time. Generally, cloud machines renting price is charged based on the computation capacity of the machines used. For example, the pricing models adopted by CloudSigma [2] and Elastichosts [3], are based on the users selected CPU frequency. To minimize the makespan, the user may need 1MHz CPU frequency additional,which results in a small but still higher monetary cost. With such pricing schema, an interesting challenge arising to the user may be how to properly select the machine and tune their CPU frequency so that makespan and cost of running his/her application are minimized.

More powerful or faster machines will cost more but produce shorter makespan schedules while slower machines may cost less and produce longer makespan schedules. In this context, cost and makespan are conflicting objectives. Selecting the machine capacity for the improvement of the total cost values cannot be possible without deteriorating the makespan values. Note that there is no single solution that can optimize both objectives at the same time. In this case, decision makers may have to select the final preferred solution from the Pareto optimal objective vectors. Therefore, approximating the set of all the Pareto optimal objective vectors is the appropriate way to deal with a multi-objective optimization problem.

It is well-known that solutions to a multi-objective optimization problem under trivial situations could be an optimal solution of a scalar optimization problem in which the objective is an aggregation of all the objectives [4]. Therefore, approximation of the PF can be decomposed into a number of scalar objective optimization sub-problems. To the best of our knowledge, none of the majority of the current state multi-objective workflow scheduling algorithms [5]–[10] considered decomposition.

With cost and makespan minimization in mind, in this paper, we propose a workflow-scheduling algorithm based on decomposition (WSABD). Given a scientific workflow with deterministic model of execution time and communication time, given also a set of resources with CPU frequencies and cost, WSABD starts with initialization step using CFMax [11], then updates the functional values, finally, checks for the satisfaction of stopping criteria if stopping criteria are satisfied, the algorithm returns Pareto Front solutions otherwise goes to the update step. In order to speed up the runtime, WSABD uses a search operation to get new solutions rather than overlapping mutation operations. The main contribution of this paper is proposing a novel workflow scheduling algorithm with three variants, which incorporate decomposition approaches in workflow scheduling and use search operation rather than mutation.

The rest of this paper is organized as follow: section II presents the related works, section III identifies the problem to be solved; Proposed algorithm is described in section IV; Evaluation settings and findings are presented in section V;

Finally, section VI concludes our work and summarizes future works .

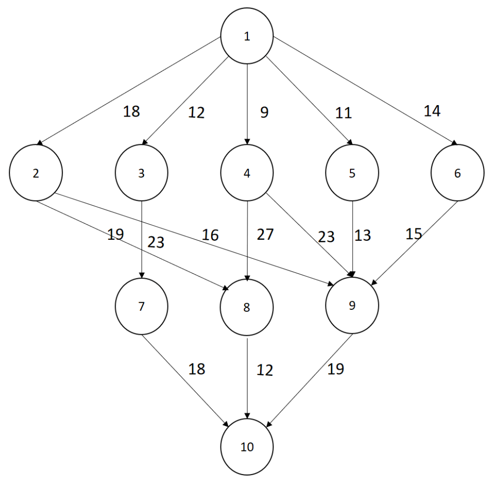


Fig. 1: DAG example with communication time

Tasks Id	$Machine1$	$Machine2$	$Machine3$
1	30	12	16
2	27	21	72
3	4	36	6
4	21	6	20
5	20	16	81
6	28	6	48
7	35	14	7
8	5	48	30
9	21	3	36
10	48	2	64

TABLE I: An example of execution time

Machines	$MaxCPU_{fr}$	$MinCPU_{fr}$	$StepCPU_{fr}$
1	4200	2100	300
2	3600	2400	300
3	3000	2000	200

TABLE II: An example of Frequency settings for 3 machines

II. Related works

Workflow scheduling and resource provisioning have become the fundamental research topic on the cloud-computing platform. A remarkable number of works have been done to deal with optimization problem, either as single objective, bi-objective or as multi-objective optimization problem. Among those focused on optimization of makespan as a single objective, HEFT [12] is the lightweight workflow scheduling heuristic for a heterogeneous environment like a cloud. Given a set of machines, HEFT ranks tasks according to their values, and then schedule them one after another to the machine that minimizes its execution time by taking into consideration the tasks communication time. Because of its low complexity, HEFT has been employed by other researchers to provide new workflow scheduling algorithms [11], [13]–[16]. With the objective of mapping all the workflow tasks to the available VMs so that makespan and cost are minimized, [17] proposed a bi-objective algorithm which is a hybrid of HEFT and GSA(Gravitation Search Algorithm). As cloud computing emerges, modern workflow scheduling algorithms have to be able to optimize more than one objective. Different researches have been carried out to respond to this trend [6]–[10]. Mostly, multi-objective workflow scheduling algorithms rely on finding the Pareto set solutions and then find non-dominated solutions from the Pareto set. Different from another usual objective, the work presented in [7] designed a new systematic method that considers both tasks security demand and interaction in secure tasks placement in the cloud. This work proposed a heuristic algorithm that is based on tasks completion time and security requirements. Most of the multi-objective workflow-scheduling algorithm considered two to three objective at once, [6] proposed a generic multi-objective optimization framework to evaluate list scheduling heuristic over four scheduling objectives(makespan, cost, energy consumption and reliability). The proposed algorithm approximates the optimal solution by considering user-specified constraints on objectives in a dual strategy: maximize the distance to the users constraints for dominant solutions and minimize it. Evolutionary algorithms are an excellent way to solve the multi-objective optimization problem, however, there are designed for non-constrained problems. With the aim of investigating the proper task-machine mapping plan to minimize the total financial cost and degree of imbalance under deadline constraints, The algorithm proposed in [18] modified NSGA-II (Non-dominated Sorting Genetic Algorithm-II) proposed in [19] and make it accept constraints, Then use modified version to to solve the considered optimization problem. [20] proposed a scheduler that is capable of computing a set of trade-off solutions in a single run. The main objective of this work is to rely on the empirical models for the execution time of workflow tasks, the involved energy consumption and then overcome the drawback of deterministic facts(execution time and communication time are known in advance). Decomposition is a traditional multi-objective optimization strategy that decomposes a multi-objective optimization problem into a number of scalar optimization problems and optimizes them simultaneously. [4] presented a multi-objective evolutionary algorithm that is based on the decomposition techniques. However, this work is not designed for workflow scheduling purposes.

Differentiating from the work presented above, this paper proposes a workflow-scheduling algorithm based on decomposition. Our algorithm uses a search operation to get a new solution rather than overlapping mutation. To generate the initial population our algorithm employs CFMax [11].

III. System Model

We assume the presence of cloud computing machines that are charged based on the pay-as-you-go basis of the CPU frequency used to execute each task in the workflow. Each allocated machine is provisioned from the beginning of the execution time of the task until its completion time.

Information about data transfer between tasks and execution time of tasks when the machines run at their maximum CPU frequency are known in advance as illustrated in figure 1 and table I respectively. We also consider a workflow application modeled as Directed Acyclic Graph(DAG) $G = (T, D)$, where T represent a set of interdependent tasks $T = \{t_1, t_2, .., t_n\}$ and D represent a set of intermediate data between two adjacent tasks $D = \{d_{ij}\}$ (Figure 1 for illustration). We use $pred(t_i)$ to determine a set of predecessors of task t_i, and $Succ(t_i)$ to determine a set of successors of task t_i. If task t_i is adjacent to task t_j, task t_i is a parent of task t_j, t_j is a child of task t_i. Task t_j can not start its execution before all its parents are completed and transmitted all required data d_{ij} to it. If a task is executed on the machine using a CPU frequency lower than the maximum, its execution time can be calculated by:

$$ET_{(t,f)} = (\beta \cdot (\frac{f_{max}}{f} - 1) + 1) \cdot ET_{(t,f_{max})} \tag{1}$$

where $ET_{(t,f_{max})}$ is the execution time when task t_i runs at the maximum CPU frequency and the parameter β indicates the impact of the CPU frequency on task execution time in the range of 0 to 1. In this paper, we set β=0.4 by default.

The heterogeneous machines operates on variant CPU frequencies in between maximum and minimum frequency (f_{min}, f_{max}) with a step frequency f_{step} that determine the variability level as illustrated in table II. Each machine is charged according to the CPU frequency allocated to each task. We adopted one of the pricing model presented in [15]. Let $C_{(m,f)}$ represent the price charge per time unity use of a machine m with CPU frequency f, $C_{(m,f_{min})}$ represent the price charge per time unity use of a machine m running at minimum CPU frequency f_{min}, and δ represent the coefficient to tune the charging rate of the price according to f. Then, with linear pricing model $C_{(m,f)}$ can be calculated as bellow:

$$C_{(m,f)} = C_{m,f} + \delta_m \cdot \frac{f_i - f_{min}}{f_{min}} \tag{2}$$

Let also $EC_{(t,m,f)}$ denote the task execution cost on the machine running at a frequency f. $EC_{(t,m,f)}$ is calculated as:

$$EC_{(t,m,f)} = ET_{(t,f)} \cdot C_{(m,f)} \tag{3}$$

The total cost to execute the whole workflow tasks is calculated as:

$$TC = \sum_{\forall(t,m)\in S} EC_{(t,m,f)} \tag{4}$$

where S is the schedule which describes the tasks-machines mapping and the operating CPU frequency of each machine. The aim of multi-objective optimization algorithms is to find the trade-off between contradicting objectives. During multi-objective workflow scheduling, there may be a big or even infinite number of solution. However, only non-dominated solutions can be taken by decision-makers for the selection of the final preferred solution. A solution S_a is said to dominate a solution S_b if and only if S_a is better than S_b in both objectives. $F(x^{'})$ is said to be Pareto Optimal if there is no solution x such that $F(x)$ dominates $F(x^{'})$. This means that any change in Pareto optimal values for the satisfaction of one objective must lead to the change in at least another objective. A set of all Pareto optimal solutions is called PS, and a set of all Pareto optimal objective vectors is called PF. Some mathematical models have been developed to approximate PF, however, it is well-known that Pareto Optimal solutions for a multi-objective problem under slight condition can be optimal solution of a scalar optimization problem in which objective is an combination of both the weight vectors [4]. Hence, approximation of the PF can be decomposed into a number of scalar objective optimization sub-problems. In this paper, we adopted three decomposition approaches: Weighted sum, Tchebycheff, and Penalty Boundary intersection approaches, to decompose the problem of approximation of the PF into number of scalar optimization problems. Let g^{ws} be decomposition method using weighted sum approach, g^{te} be decomposition method using Tchebycheff approach, and g^{pbi} be decomposition method using penalty boundary intersection. The decomposition values can be computed as bellow:

$$minimize \quad g^{ws}(x|\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x) \tag{5}$$

$$minimize \quad g^{te}(x|\lambda, z) = \max_{1\leq i\leq m} \{\lambda_i |f_i(x) - z_i|\} \tag{6}$$

$$\begin{aligned} minimize \quad & g^{pbi}(x|\lambda, z) = d_1 + \theta d_2 \\ \text{where} \quad & d_1 = \frac{\|(F(x) - z)^T \lambda\|}{\|\lambda\|} \\ \text{and} \quad & d_2 = \|F(x) - (z + d_1\lambda)\|. \end{aligned} \tag{7}$$

Where x is a vector containing the variables of both objectives to be optimized(Cost and Makespan) λ is weight vector, z represents the reference point which is equal to the minimal values for both objective considered(Cost, Makespan),θ is a penalty parameter that has to be greater than 0, d_1 is the distance between z^* and y, d_2 is the distance between $F(x)$ and line L. More details about those three approaches can be found in [4].

Based on the model and assumptions above, we presented the multi-objective workflow scheduling algorithm, which generates schedules and properly tunes the CPU frequency for each task so that the makespan and total cost for the submitted workflow are minimized.

IV. The proposed algorithm

This section describes the Workflow Scheduling Algorithm Based on Decomposition(WSABD), a multi-objective algorithm proposed to solve the problem described in section III. As presented in algorithm 2, WSABD takes seven elements(W, R, SC, N, WV, T, A) as input. Those input elements are described as follow: W is a set of tasks with known execution time and communication time, R is a set of resources with CPU frequencies and associated prices, SC is a stopping criteria (a fixed number of iteration), N is number of sub-problem considered, WV is a uniform spread of N Weight Vectors: $\lambda^1 ... \lambda^N$ (N=2), T is the number of the weighted vectors in the neighborhood of each weight vector, A is a set of decomposition approaches that are used to compute and compare new solutions. In our case, A consists of three approaches: Weighted Sum (WS), Tchebcheff (TE) and Penalty Based Boundary (PBI). WSABD returns EP as the output, where EP is a set of non-dominated solutions. The proposed algorithm consists of three main steps: Initialization, Update and Stopping criteria.

Algorithm 1 Using a Specific Decomposition Approach

```
    switch ($A$)
 2: case (WSABD-WS):
        Calculate $g^{ws}(x^j \| \lambda^j)$, $g^{ws}(y^{'} \| \lambda^j)$ using Eq.(5), break
 4: case (WSABD-TE):
        Calculate $g^{te}(x^j \| \lambda^j, z)$, $g^{te}(y^{'} \| \lambda^j, z)$ using Eq.(6), break
 6: case (WSABD-PBI):
        Calculate $g^{pbi}(x^j \| \lambda^j, z)$, $g^{pbi}(y^{'} \| \lambda^j, z)$ using Eq.(7), break
 8: end switch
    if ($g^*(y^{'} \| \lambda^j, z) \leq g^*(x^j \| \lambda^j, z)$) then
10:     Set $x^j = y^{'}$ and $FV^j = F(y^{'})$
    end if
```

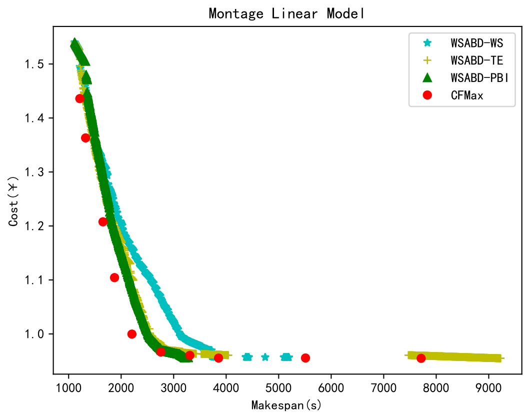
In our previous study [21], we proposed workflow scheduling algorithm with two variant(CFMax, CFMin). The purpose of our study [21] is to minimize the users' monetary expenditure for the submitted workflow application under a given deadline. The experimental results of our proposal show that CFMax performs better than CFMin. To satisfy the user's deadline regardless of the total cost, CFMax starts with makespan-aware scheduling algorithm (HEFT but other can be used like in [11]) and schedule each task to the appropriate machine using the maximum CPU frequency. To guarantee the cost reduction, a Reduction weight(RW) table is created to measure the cost reduction impacted by the task reassignment and CPU frequency re-allocation.The values are inserted in RW according to Eq.(8), where $TC_{(t,m',f')}$ represents new task's cost after changing CPU frequency, and $TC_{(t,m,f)}$ represents the cost of executing the tasks on the machine using current CPU frequency. To take re-assignment decision, the combination of machine and CPU frequency that produces the maximum value in RW are selected as the winner. In the first step of WSABD, we initialize the inputs, CFMax is used to generate the initial population (line 4 of Algorithm 2). In the second step, we update the initialized variables by iteratively changing vectors variables and iteration settings. We compute new solutions according to the updated settings and update new solutions according to the decomposition approaches decision. In step three, we check if stopping criteria is satisfied to return the Non-Dominated solutions (namely, EP), otherwise go to step two.

$$RW_{(t,m,f)} = TC_{(t,m',f')} - TC_{(t,m,f)} \tag{8}$$

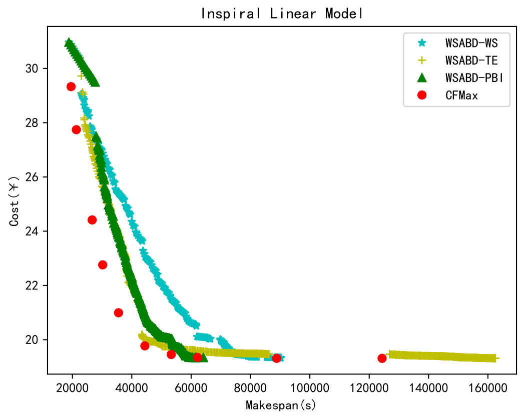
Algorithm 2 WSABD (Workflow Scheduling Algorithm Based On Decomposition)

```
Input: $W, R, SC, N, WV, T, A$
Output: $EP$: External Population(Non Dominated Solutions)
    Step1: Initialization :
 1: Set Iteration = 0, $EP$ = $\emptyset$
 2: Calculate Euclidean distance between any two weight vectors then work out the $T$ closest weight vectors to each weight vector store in $B$.
 3: Initialize $W$ and $R$
 4: Initialize the population using CFMax [11]
 5: Calculate the individual cost and makespan of the population as the objective function value ($FV$)
 6: Initialize the reference point $z$, $z$=($minimized_{cost}$, $minimized_{makespan}$)
    Step 2: Update
 7: for (individual: population) do
 8:     if ($index[individual]$ is even) then
 9:         Get new individual $y^{'} \leftarrow$ through minimized-cost
10:     else
11:         Get new individual $y^{'} \leftarrow$ through minimized-makespan
12:     end if
13:     Calculate the cost and makespan of $y^{'}$, rename them as $FV(y^{'})$
14:     Update reference point $z$ using FV($y^{'}$)
        Update the solutions of this individual's neighbors
15:     for (j ∈ B (individual)) do
16:         Call Algorithm 1 with $A$: WSABD-[$SW$, $TE$, $PBI$]
17:     end for
        Update $EP$
18:     Remove from $EP$ all the vectors dominated by $FV(y^{'})$
19:     add $F(y^{'})$ to the $EP$ if no Vectors in $EP$ dominate $FV(y^{'})$
20: end for
21: iteration +=1
    Step 3: Stopping criteria
22: if (iteration==SC) then
23:     return EP
24: else
25:     Go to step 2.
26: end if
```

In detail, step one (from line 1 to 6) consists of initialization of the input variables such as number of weight vectors in the neighborhood (T), weight vector indexes B, machines information, DAG information. After initializing the initial population using CFMax, the individual cost and makespan are calculated as objective function values. Among the objective function values (F), the minimum one is selected as the initial reference point z. Note that T plays an important role by limiting the search operation to a certain extend. The second step (from line 7 to 26) consists of two sub-steps. In the first sub-step (from line 7 to 15), we update the individual

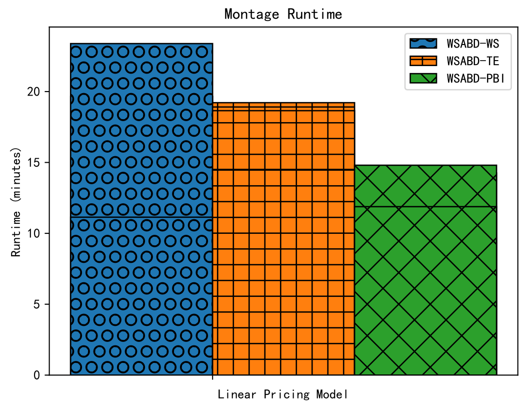


(a) DAG: Montage

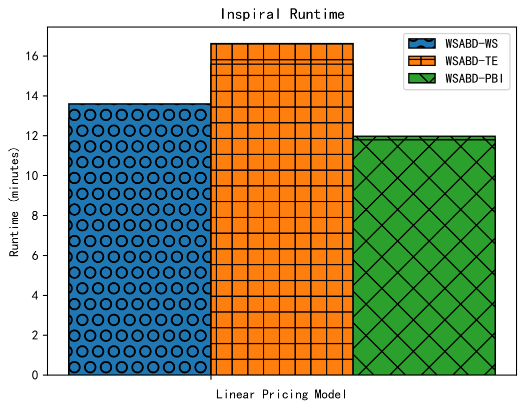


(b) DAG: Inspiral

Fig. 2: Optimization results



(a) DAG: Montage



(b) DAG: Inspiral

Fig. 3: Runtime results

by searching the minimized cost (y) or makespan (y) based on the position index of the individual in the population. If the index of the individual in the population is even, we get the minimized cost otherwise we get the minimized makespan. Then the new cost and the new makespan are calculated according to the new individual values and renamed as F(y). Finally, the reference point is updated according to F(y). In the second sub-step(from line 16 to 26): The solutions of the individuals neighbors are updated. As shown in the Algorithm 1, in this stage we use either of the approaches defined in Eq.(5), (6) and (7). For each selected approach, we calculate the $g^*(y'\|\lambda^j, z)$ and $g^*(x^j\|\lambda^j, z)$. Before re-setting the current cost-makespan(x^j) value to be equal to the new individual y the two values are compared first. If the new values ($g^*(y'\|\lambda^j, z)$) are less or equal to the current values ($g^*(x^j\|\lambda^j, z)$), update is allowed otherwise the current values are keept and the algorithm continues. EP is also updated according to the new values of $FV(y)$. This is repeated until the stopping criteria are satisfied then the final EP is returned.

V. Evaluation Results

The algorithm presented in Section 2 and its simulation tool are implemented in Java. A 4 core Intel i5-7300HQ @2.5GHz CPU with 8GB-RAM PC is used as an experimental environment. Each simulated machine run at a CPU frequency in the range between maximum and minimum, with a variation step, selected randomly as shown in table II. We let pricing model be linear and its parameters c_{min} and δ to be equal 9.24 and 3.33 respectively as in our previous work [11].

We consider two types of workflows: Montage and Inspiral with 1000 tasks per each workflow. The input parameters such as population size, number of neighbors and number of the iteration are set to be 20, 5 and 2000 respectively. WSABD is applied to each DAG with each of the other parameters defined above.

Given the above parameters and settings, CFMax is firstly executed. However, original CFMax is designed for single objective optimization problem. With the help of weight vector that is assigned to each objective, CFMax gets the ability to respond to the multi-objective optimization problem. The results generated by CFMax are also the initial population for WSABD. WSABD can approximate optimal solutions (set of makespans and costs) at the end of a single run. Note that WSABD is made of three decomposition approaches. As long as the iteration number is not yet achieved and all possibility not tried yet the algorithm will continue to generate new solutions. The optimization results for both considered workflows are presented in Figure 2. Users can determine the solutions that suit their needs based on some constraints such as users' budget and/or deadline. To evaluate the runtime of WSABD, the aforementioned environment and settings are used to run each algorithm 100 times and return the average as their runtime results as shown in Figure 3.

Based on the result presented in Figures 2 and 3, the key findings of this paper can be summarized as follows:

- WSABD-WS and WSABD-TE can generate almost the same optimal results for both objectives and these results are fairly close to the schedules generated by running CFMax for multiple times.
- In most of the cases, WSABD-WS and WSABD-TE produce better scheduling results than WSABD-PBI.
- WSABD-PBI can be completed within a shorter time compared to other approaches.
- The comparison result between the runtime of WSABD-WS and that of WSABD-TE varies when different DAGs are used.

VI. Conclusion

In this paper, the problem of minimizing makespan and monetary cost of a submitted workflow is considered and modeled as a multi-objective optimization problem. A novel workflow scheduling algorithm based on decomposition is proposed to assist in the tunning of the CPU frequency for each task so that both makespan and cost can be minimized. The evaluation results on optimization show that in different conditions, the performance of running WSABD-WS and WSABD-TE for just one time, which generate a set of solutions, is stable and almost as good as running CFMax for multiple times and generating one solution per time. However, the evaluation results on runtime show that WSABD-PBI takes shorter runtime than WSABD-WS and WSABD-TE for both considered workflows. Future works could consider different pricing models, study the impact of different settings like number of machines, number of iteration to the evaluation result. Additionally, the efficiency of the proposed algorithm could be tested through a real cloud platform.

Acknowledgment

The work was supported by the National Science Foundation of Fujian Province of China (No. 2018J01107), and was also jointly supported by National Natural Science Foundation of China (NSFC, Grant No. 61671397, 61672439).

References

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[2] "Cloudsigma," https://www.cloudsigma.com/.

[3] "Elastichosts," https://www.elastichosts.com/.

[4] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, Dec 2007.

[5] B. Keshanchi, A. Souri, and N. J. Navimipour, "An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing," *Journal of Systems and Software*, vol. 124, pp. 1 – 21, 2017.

[6] R. P. Hamid Mohammadi Fard and T. Fahringer, "Multi-objective list scheduling of workflow applications in distributed computing infrastructures," *Journal of Parallel and Distributed Computing*, vol. 74, no. 3, pp. 2152 – 2165, 2014.

[7] F. Abazari, M. Analoui, H. Takabi, and S. Fu, "Mows: Multi-objective workflow scheduling in cloud computing based on heuristic algorithm," *Simulation Modelling Practice and Theory*, vol. 93, pp. 119 – 132, 2019, modeling and Simulation of Cloud Computing and Big Data.

[8] H. Hu, Z. Li, H. Hu, J. Chen, J. Ge, C. Li, and V. Chang, "Multi-objective scheduling for scientific workflow in multicloud environment," *Journal of Network and Computer Applications*, vol. 114, pp. 108 – 122, 2018.

[9] G. L. Min Zhang, "Multi-objective optimization algorithm based on improved particle swarm in cloud computing environment," p. 1413, 2019.

[10] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, and S. Hu, "Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based heft," *Future Generation Computer Systems*, vol. 93, pp. 278 – 289, 2019.

[11] B. Emmanuel, Y. Qin, J. Wang, D. Zhang, and W. Zheng, "Cost optimization heuristics for deadline constrained workflow scheduling on clouds and their comparative evaluation," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 20, p. e4762, 2018.

[12] H. Topcuoglu, S. Hariri, and Min-You Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, March 2002.

[13] J. J. Durillo, H. M. Fard, and R. Prodan, "Moheft: A multi-objective list-based method for workflow scheduling," in *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, Dec 2012, pp. 185–192.

[14] W. Zheng and R. Sakellariou, "Budget-deadline constrained workflow planning for admission control," *Journal of Grid Computing*, vol. 11, no. 4, pp. 633–651, Dec 2013.

[15] W. Zheng, Y. Qin, E. Bugingo, D. Zhang, and J. Chen, "Cost optimization for deadline-aware scheduling of big-data processing jobs on clouds," *Future Generation Computer Systems*, vol. 82, pp. 244 – 255, 2018.

[16] I. Pietri and R. Sakellariou, "Cost-efficient cpu provisioning for scientific workflows on clouds," in *Economics of Grids, Clouds, Systems, and Services*, J. Altmann, G. C. Silaghi, and O. F. Rana, Eds. Cham: Springer International Publishing, 2016, pp. 49–64.

[17] A. Choudhary, I. Gupta, V. Singh, and P. K. Jana, "A gsa based hybrid algorithm for bi-objective workflow scheduling in cloud computing," *Future Generation Computer Systems*, vol. 83, pp. 14 – 26, 2018.

[18] M. Zhang, H. Li, L. Liu, and R. Buyya, "An adaptive multi-objective evolutionary algorithm for constrained workflow scheduling in clouds," *Distributed and Parallel Databases*, vol. 36, no. 2, pp. 339–368, Jun 2018.

[19] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, April 2002.

[20] J. J. Durillo, V. Nae, and R. Prodan, "Multi-objective energy-efficient workflow scheduling using list-based heuristics," *Future Generation Computer Systems*, vol. 36, pp. 221 – 236, 2014.

[21] W. Zheng, B. Emmanuel, C. Wang, Y. Qin, and D. Zhang, "Cost optimization for scheduling scientific workflows on clouds under deadline constraints," in *2017 Fifth International Conference on Advanced Cloud and Big Data (CBD)*, Aug 2017, pp. 51–56.