Cost optimization for scheduling scientific workflows on clouds under deadline constraints

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

***Abstract*—Nowadays cloud service providers usually offer to users virtual machines with various combinations of configurations and prices. As this new service scheme emerges, the problem of choosing the cost-minimized combination under a deadline constraint is becoming more complex for users. The complexity of determining the cost-minimized combination may be resulted from different causes: the characteristics of user applications, and providers' setting on the configurations and pricing of virtual machine. In this paper, we proposed an algorithm with two variants to help the users to schedule their workflow applications on clouds so that the cost can be minimized and the deadline constraints can be satisfied. The proposed algorithm is evaluated by extensive simulation experiments with diverse experimental settings.**

I. Introduction

As cloud computing services prevail in commercial markets, cloud service providers now tend to offer to their users a variety of virtual machines with different configurations and prices. Normally, the price is charged based on the computation or storage resource used. That is to say it costs more when a user choose a more powerful and/or fast resource. For example, in the pricing model adopted by ElasticHosts [1] and CloudSigma [2], even a 1MHz frequency more of CPU customized by a user will still result in a small but still higher monetary cost. With such a pricing scheme, an interesting challenge arising to a user may be how to properly select virtual resources for his/her application so that the user's cost can be minimized while the QoS (Quality of Service) can still be satisfied. For simplicity, in this paper, we view makespan (the overall execution time of an application) as the only QoS metric we feel interested in, and consider CPU as the only virtual resource with different options on frequencies and associated prices.

One may argue that the cost-minimizing challenge can be easily addressed by iteratively lowering the chosen CPU's frequency so as to continuously reduce the overall monetary cost until the makespan reaches the deadline. However, this simple strategy can hardly work well for those application with multiple interdependent tasks such as scientific workflow. Such applications may consist of a collection of tasks with different characteristics. Some of the tasks may be CPU-bound, i.e., their execution times are very sensitive to the variation of CPU frequency, while some may be I/O-bound which are less sensitive. Therefore, changing CPU frequency for different types of tasks may result in different impacts to the makespan. It becomes a complicated question to decide for which tasks their CPU frequencies shall be lowered and to what extent their frequencies shall be changed. Moreover, the execution schedules for workflow applications usually contain gaps (idle time) as a result of task dependencies that need to be satisfied. This implies the opportunity to take advantage of such gaps by operating CPU at different frequencies for each task so that the cost is minimized, yet a reasonable makespan within a deadline is still achieved.

With the assumption that users are interested in completing a workflow within a specified deadline on clouds that provision CPU resources with different configurations and prices, this paper contributes an algorithm with two variants that can be used to generate cost-optimized and deadline-constrained schedules for workflows by choosing appropriate frequency of the allocated CPU to each task. Typically, a cost-optimized schedule is not produced by choosing lowest frequency for all allocated CPU resources. Instead, the cost optimization is normally acquired by a combination of different CPU frequencies with different tasks. The basic idea of the proposed algorithm is starting with all tasks tentatively scheduled at highest frequencies, and then iteratively trying to reduce CPU frequencies for tasks which are sorted by a certain priority related to cost saving. The two variants of the algorithm are extensively evaluated by simulation using two different scientific workflow applications and three different pricing models for CPU frequency, which manage to capture typical ways of varying the price of CPU based on frequency. By wisely choosing CPU frequency for each task, the algorithm allow users to complete their workflow applications within their deadlines at a low monetary cost.

The remainder of this paper is organized as follows. In Section II, we give a brief overview of related works. In Section III we identify the problem to be solved. In Section IV, we present the proposed algorithm aiming at minimizing cost for workflow applications while still meeting the deadline constraint. In Section V, the effectiveness of the proposed algorithm is evaluated. Finally, we conclude the paper and summarize future works in Section VI.

II. Related Work

Resource provisioning and job scheduling has become crucial topics of research on distributed system like clouds [3]. Different works have been done to deal with different optimization objectives.

Some scheduling researches have been focused on the

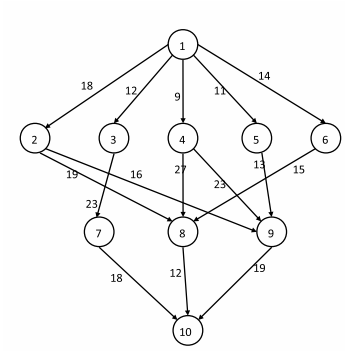
minimization of the execution time of the whole workflow [4], [5], while some others concentrated on optimization of the energy used by the machines during workflow execution [6], [7]. These works often leverage meta heuristics (e.g., evolutionary algorithm) [8], [9], which normally take high time cost. One of well-known light-weight heuristics for scheduling workflow on multiple heterogeneous computing resources is HEFT [4]. Given a set of resources, HEFT arranges jobs in a list according to their rank values, and then schedule one after another to a resource which will minimize its execution time by taking into consideration the communication cost. This algorithm has been extended in various ways [5] to address different scheduling problems. Since HEFT manages to generate a fair schedule with low complexity, it has been widely employed in many other meta algorithms [10], [11], [12]. In [10], HEFT is used to generate the initial schedule in conjunction with CSFS-Max and CSFS-Min methods which target at helping the user to choose the best configuration. In [11], HEFT is employed by LOSS and GAIN approaches to realize an overall time optimization under budget constraints. In [12] HEFT has played a role as a makespan-aware scheduler. This work has proposed an approach to help users in splitting a sum of frequency onto a fixed number of resources by giving each resource identical frequency configuration so that the makespan can be reduced.

Most of existing works have concentrated on the optimization of different constraints related to workflow scheduling by considering both at the same time or one of them, e.g., [13], [14], [15], [7], [16]. The work in [13] proposed a bi-criteria priority particle swarm optimization (BPSO) to schedule workflow application to cloud resources so as to optimize the execution cost for running the workflow and also minimize the total execution time under the given budget. The work presented in [14] adopted partial critical path to IAAS cloud and proposed two algorithms: a one-phase algorithm IC-PCP, and a two-phases PCP algorithm which both have polynomial time complexity suitable for large workflows. The objective of this work was to minimize the cost of workflow execution while still meeting the user's deadline.

Differentiating with the works described above, the algorithm presented in this paper assumes cloud resources with different configurations and prices, and considers choosing an appropriate CPU frequency for each workflow task so as to minimize the user's cost without violating the specified deadline. Similar to [10], [11], [12], our algorithm also employs HEFT to obtain an initial schedule at the beginning of scheduling.

III. Problem Description

In the problem we considered, the user submits a workflow and requires its completion before a specified deadline to the cloud providers possessing a number of cloud resources. The schedule specifies for each task the start time, the allocated resource and the CPU frequency used. The allocated resource is provisioned from the time the task execution starts until its completion. In line with the 'pay as you go' feature of cloud computing, the user is only charged for the task execution time in light to the price based on the CPU frequency used, while the cost of other resource characteristics, such as memory and disk is considered to be fixed.



(a) A DAG example with communication times

Tasks	Resource 1	Resource 2	Resource 3
Task1	30	12	16
Task2	27	21	72
Task3	4	36	6
Task4	21	6	20
Task5	20	16	81
Task6	28	6	48
Task7	35	14	7
Task8	5	48	30
Task9	21	3	36
Task10	48	2	64

(b) An example of execution times

Resources	f_{max}	f_{min}	f_{step}
Resource 1	3000 MHz	1000 MHz	100 MHz
Resource 2	2800 MHz	1400 MHz	200 MHz
Resource 3	2700 MHz	1800 MHz	300 MHz

(c) An example of frequency setting for 3 resources

Fig. 1. An DAG Example for Illustration

Each workflow considered in the paper can be modelled as a Directed Acyclic Graph (DAG), as illustrated in Fig. 1. The information on task execution times when resources run at maximum CPU frequency and data transfer times, as illustrated in Fig. 1(a) and 1(b), is assumed to be known. When a task executes at a frequency lower than the maximum, its execution time can be estimated by:

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

where $runtime_{(t,f)}$ is the task runtime when task t runs at the maximum frequency f_{max} and the parameter β indicates the impact of the CPU frequency on task runtime, ranging between 0 and 1 for tasks with I/O-bound or CPU-bound characteristics respectively. In this paper, we set $\beta = 0.4$ by default.

A set of heterogeneous resources which can operates on variable CPU frequencies are assumed, which distribute between a minimum and maximum frequency (f_{min} and f_{max}) with a step f_{step}, as shown in Fig. 1(c). Each resource is charged according to the frequency allocated to each task. As different pricing models may be used by cloud providers according to their needs and goals, in this paper, we adopt three

types of models the same as presented in [10]. These models are linear, sublinear and superlinear pricing models. Suppose that $C_{(r,f)}$ denotes the price charged for per time unit's use of resource r with CPU frequency f_r, $C_{(r,f_{min})}$ denotes the price of resource r at minimum frequency f_{min}, and δ_r denotes the coefficient used for resource r to tune the changing rate of price according to frequency. Then in the case of linear pricing model, $C_{(r,f)}$ is computed by

$$C_{(r,f)} = C_{(r,f_{min})} + \delta_r \cdot (\frac{f_r - f_{min}}{f_{min}}) \tag{2}$$

and in the case of superlinear pricing model, $C_{(r,f)}$ is computed by:

$$C_{(r,f)} = C_{(r,f_{min})} + \delta_r \cdot ((1 + \frac{f_r - f_{min}}{f_{min}}) \cdot \log(\frac{f_r - f_{min}}{f_{min}})) \tag{3}$$

while in the case of sublinear pricing model, $C_{(r,f)}$ is computed by:

$$C_{(r,f)} = C_{(r,f_{min})} + \delta_r \cdot \log(1 + \frac{f_r - f_{min}}{f_{min}}) \tag{4}$$

So the overall cost for an executed workflow application is computed by

$$TotalCost = \sum_{\forall (t,r) \in S} runtime_{(t,f)} \cdot C_{(r,f)} \tag{5}$$

where S is the schedule which describes the mapping between tasks and resources as well as the operating frequency of each resource for each task.

Based on the aforementioned models and assumptions, the main objective of this paper is to generate a schedule and appropriately tune the CPU frequency for each task so that the total cost for the submitted workflow can be minimized without violating the deadline specified by the user.

IV. The Proposed Algorithm

In this section a cost-oriented algorithm to solve the scheduling problem identified in the previous section is described. The key idea of the algorithm proposed consists of two phases. In the first phase, a makespan-oriented scheduling algorithm (such as HEFT) is used to schedule tasks at maximum frequencies so that it can be guaranteed that the deadline is satisfied nevertheless the total cost is not considered. In the second phase, the algorithm iteratively change the task-resource assignment as well as the allocated CPU frequency in order to reduce the total cost, meanwhile the makespan is examined after each change to make sure the deadline is not exceeded.

In details, the algorithm starts with using HEFT to generate an initial assignment of the tasks onto resources with the maximum frequencies and computes for each reassignment of each task to a different resource and/or different frequency, a weight value associated with that particular change. These weight values are tabulated. Namely, a weight table is created for each task with a combination of each resource and each frequency. In such a table, easy weight value is computed as the cost reduction achieved by reassigning the currently scheduled task to the combination of resource and frequency

Input: A DAG G with known task execution time and communication time
A set of resources with CPU frequencies and associated prices
A DAG scheduling algorithm H
A specified deadline D
Output: A schedule specifying the start time, the allocated resource and the operating CPU frequency for each task

Algorithm: (two options: CFMax and CFMin)
1: Generate schedule S using algorithm H by using the maximum frequency for every resource.
2: Build an array $RW[n][\sum L_r]$, where n is the number of tasks, L_r is the number of frequency levels on resource r
3: **for** each Task t in G **do**
  for each pair of resource r and frequency f **do**
    if (according to Schedule S, task t is assigned to resource r with frequency f)
      then $RW[t][(r,f)] \leftarrow 0$
      else Compute $RW[t][(r,f)]$ according to Eq.(7)
  endfor
endfor
4: $condition \leftarrow (\exists RW[t][(r,f)] > 0)$
5: **while** (condition and not all possible reassignments have been tried)
  if (CFMax)
    then find the biggest non-zero value from RW, $RW[t'][(r',f')]$
    else find the smallest non-zero value from RW, $RW[t'][(r',f')]$
  Reassign task t' to resource r' with frequency f' and calculate new makespan of S, M
  if $(M > D)$
    then invalidate previous reassignment of task t' to resource r' with frequency f'.
    else update RW according to the new reassignment
  endwhile
6: **return** S

Fig. 2. The Key Steps of the Proposed Algorithm

that this value is associated with. Let $TC_{(t,r,f)}$ denote the cost for running task t on resource r with frequency f, we have:

$$TC_{(t,r,f)} = runtime_{(t,f)} \cdot C_{(r,f)} \tag{6}$$

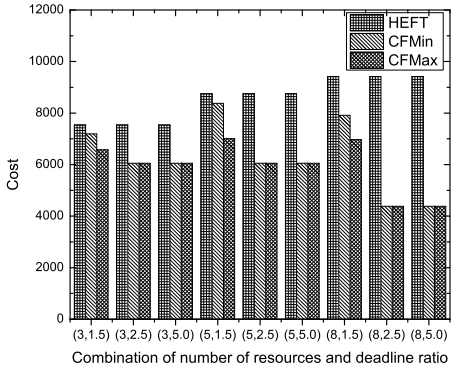
Then the reduction weight (RW, for short) for task t on resource r with frequency f can be computed by:

$$RW_{(t,r,f)} = TC_{(t,r^*,f^*)} - TC_{(t,r,f)} \tag{7}$$

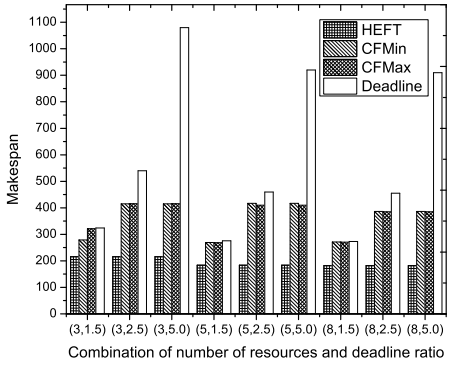
where r^* and f^* are respectively the resource and frequency that are currently assigned to task t. Once all weight values in the weight table are computed, tasks are considered for possible reassignment to a new resource or a new frequency in turn, as long as the total cost of the workflow can be reduced and the deadline is not exceeded, until no such a possible reassignment exists. Note the the algorithm will try to reassign a task to a specific pair of resource and frequency only once, so when there is no untried resource-frequency pair for possible reassignment, the algorithm will terminate. The key steps of the algorithm are illustrated in Fig. 2. Note that there are two opposite orders by which the weight values can be selected in turn, we consider two options for the algorithm: CFMax (Change From the Maximum) and CFMin (Change From the Minimum)

V. Evaluation

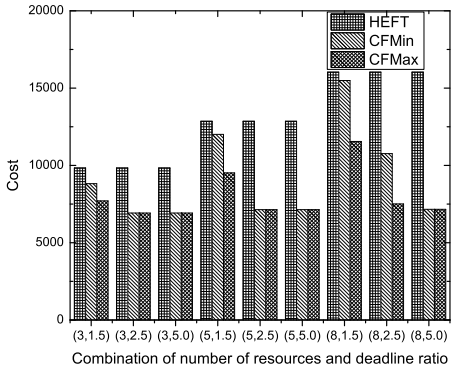
The performance of the two variants of the proposed algorithm is evaluated and compared using the aforementioned three different pricing models. The characteristics of the three workflows used in our experiments and the underlying system are described below.



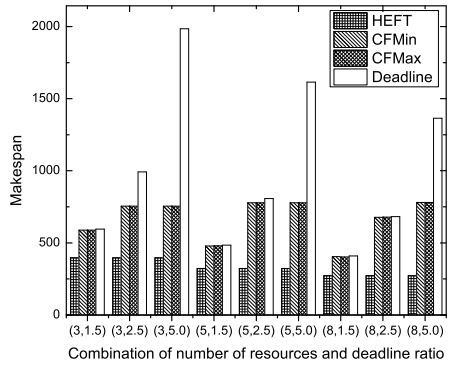
(a) Cost for AIRSN



(b) Makespan for AIRSN



(c) Cost for Montage



(d) Makespan for Montage

Fig. 3. Results under Linear Pricing

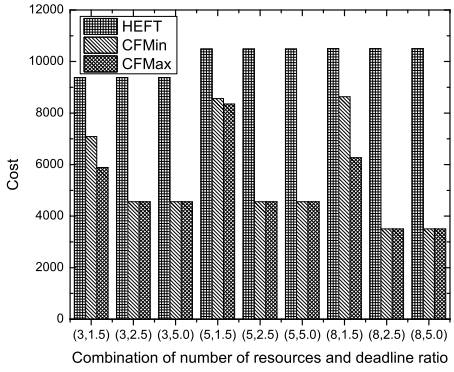
TABLE I. PARAMETERS USED IN PRICING MODELS

Pricing Model	C_{min}	δ
Linear	9.24	3.33
Superlinear	9.24	4.44
Sublinear	2.78	12

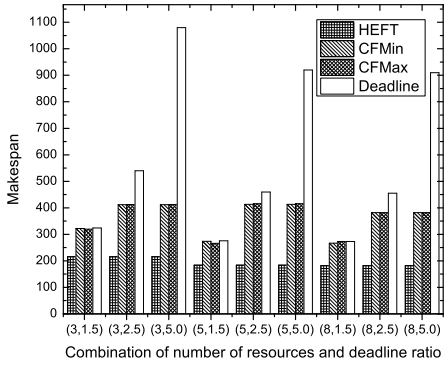
The algorithm described in the previous section and the simulation tool are implemented by Java. Each simulated resource operates at a range of frequencies between a minimum frequency and a maximum, with a frequency step randomly selected from the settings listed in Fig. 1(c). For each task, its execution time on each resource when running at the maximum frequency, as well as its communication time to the dependent tasks, is considered to be known, and randomly generated from the range between 1 and 100. The parameters C_{min} and δ for each of the three pricing models are the same as those used in [10], as listed in Table I. According to [10], these values were chosen to approximate roughly the monthly charges of ElasticHosts for the provisioning of VMs.

We consider two workflows, AIRSN [17] with 53 nodes, and Montage with 102 nodes [18] which are derived from real-world applications and have distinctive structures and sizes. The number of resources used to run these DAGs are considered from 3, 5 and 8. The proposed algorithm is applied to each workflow with each number of resources. This combination covers a wide spectrum of the ratio of workflow size to the number of available resources.

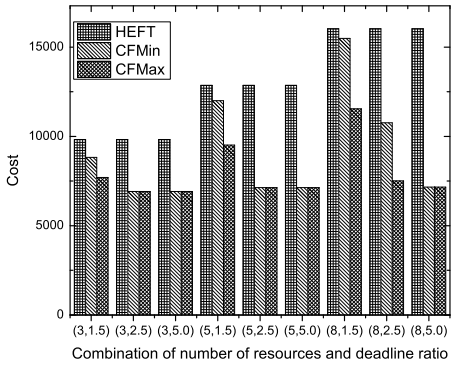
The metrics measured in our experiments are makespan and the total cost. Given a setting tuple which comprises of ($type\_of\_workflow$, $number\_of\_resources$, $pricing\_model$, $task\_runtimes$, $task\_communication\_times$), the naive HEFT is firstly executed at the maximum frequency to generate a schedule for mere makespan optimization while the cost optimization is not considered. Based on this result, we use a coefficient 'deadline ratio' γ which means the user's deadline is specified as γ times of the makespan obtained by HEFT. We consider three possible values of γ: 1.5, 2.5 and 5, which reflect various degrees of tightness of the deadline respectively. Then, we evaluate the performance of two variants of the proposed algorithm, i.e., CFMax and CFMin, with the specified deadline and compare the performance results with HEFT. As for each workflow instance, the task execution times and communication times are randomly generated, we repeat the experiments 100 times and the average results are collected and shown in Fig. 3, 4 and 5 for three different pricing models. In each figure, the cost results and makespan results are shown in the sub-figures on the left column and right column respectively. Note that the makespan and cost result diagrams on the same line are highly corelative. We display the results of HEFT, CFMin and CFMax in the cost



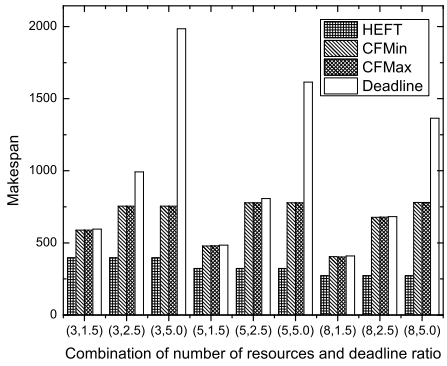
(a) Cost for AIRSN



(b) Makespan for AIRSN



(c) Cost for Montage



(d) Makespan for Montage

Fig. 4. Results under Superlinear Pricing

diagrams, while in each makespan diagram a white bar is added to demonstrate how close the makespans of CFMax and CFMin are to the deadline.

Based on these results, we summarize a list of key findings as follows:

- In all cases of experimental settings, CFMax and CFMin manage to reduce the total cost for workflow execution without violating the deadline constraint. Especially when the deadline ratio is high, e.g., $\gamma = 5$ for Montage, the cost reduction percentage is up to 55% compared to the total cost of HEFT for which the cost optimization is not considered.
- In most cases, CFMax outperforms CFMin by achieving a more significant cost reduction. When the deadline constraint fall in a reasonable range, the advantage of CFMax over CFMin is significant. However, when the deadline is too relaxed, the schedule that allocates all tasks at minimum frequency over the resources can be found by both CFMax and CFMin. At the other end, when the deadline is too tight, little cost reduction can be achieved no matter what algorithm is used.
- The pricing model's impact on the comparion of CFMax and CFMin is non-trivial. For example, when sublinear pricing model is used with a tight deadline (when $\gamma = 1.5$), CFMin may outperform CFMax.

VI. Conclusion and Future Works

In this paper the problem of minimizing monetary cost for the execution of deadline-constrained workflows is considered, and an algorithm that select a separate CPU frequency for each task to reduce the overall user cost is proposed. The evaluation results suggest that the proposed algorithm can significantly benefit users under different circumstances. Future work could consider different computation schemes for the weight used in our algorithm, and the evaluation may be extended with more extensive settings. Moreover, the effectiveness of the proposed algorithm could be evaluated through real cloud platforms.

Acknowledgment

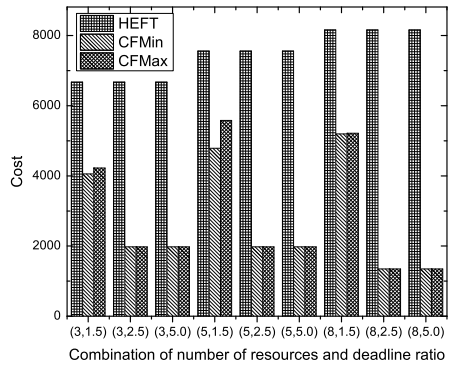
The work is supported by National Natural Science Foundation of China (NSFC, Grant No. 61671397, 61672439).

References

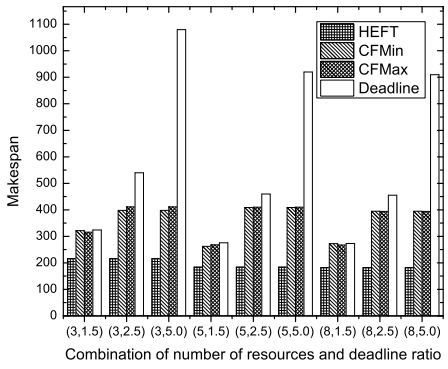
[1] "Elastichosts," https://www.elastichosts.com/, accessed April 28, 2017.

[2] "Cloudsigma," https://www.cloudsigma.com/us/, accessed April 28, 2017.

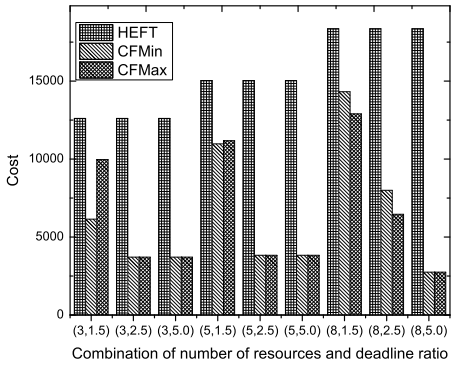
[3] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.



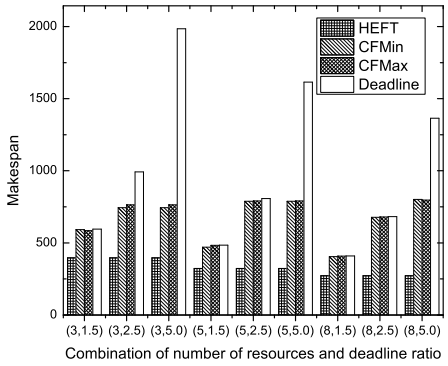
(a) Cost for AIRSN



(b) Makespan for AIRSN



(c) Cost for Montage



(d) Makespan for Montage

Fig. 5. Results under Sublinear Pricing

[4] H. Topcuoglu, S. Hariri, and M.-y. 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, 2002.

[5] W. Zheng, B. Emmanuel, and C. Wang, "A randomized heuristic for stochastic workflow scheduling on heterogeneous systems," in *Advanced Cloud and Big Data, 2015 Third International Conference on*. IEEE, 2015, pp. 88–95.

[6] D. Sun, G. Zhang, S. Yang, W. Zheng, S. U. Khan, and K. Li, "Re-stream: Real-time and energy-efficient resource scheduling in big data stream computing environments," *Information Sciences*, vol. 319, pp. 92–112, 2015.

[7] S. Yassa, R. Chelouah, H. Kadima, and B. Granado, "Multi-objective approach for energy-aware workflow scheduling in cloud computing environments," *The Scientific World Journal*, vol. 2013, 2013.

[8] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in *Advanced information networking and applications (AINA), 2010 24th IEEE international conference on*. IEEE, 2010, pp. 400–407.

[9] J. Yu and R. Buyya, "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms," *Scientific Programming*, vol. 14, no. 3-4, pp. 217–230, 2006.

[10] I. Pietri and R. Sakellariou, "Cost-efficient cpu provisioning for scientific workflows on clouds," in *International Conference on Grid Economics and Business Models*. Springer, 2015, pp. 49–64.

[11] R. Sakellariou, H. Zhao, E. Tsiakkouri, and M. D. Dikaiakos, "Scheduling workflows with budget constraints," in *Integrated research in GRID computing*. Springer, 2007, pp. 189–202.

[12] T. A. Genez, I. Pietri, R. Sakellariou, L. F. Bittencourt, and E. R. Madeira, "A particle swarm optimization approach for workflow scheduling on cloud resources priced by cpu frequency," in *Utility and Cloud Computing (UCC), 2015 IEEE/ACM 8th International Conference on*. IEEE, 2015, pp. 237–241.

[13] A. Verma and S. Kaushal, "Bi-criteria priority based particle swarm optimization workflow scheduling algorithm for cloud," in *Engineering and Computational Sciences (RAECS), 2014 Recent Advances in*. IEEE, 2014, pp. 1–6.

[14] S. Abrishami, M. Naghibzadeh, and D. H. Epema, "Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 158–169, 2013.

[15] E.-K. Byun, Y.-S. Kee, J.-S. Kim, E. Deelman, and S. Maeng, "Bts: Resource capacity estimate for time-targeted science workflows," *Journal of Parallel and Distributed Computing*, vol. 71, no. 6, pp. 848–862, 2011.

[16] E.-K. Byun, Y.-S. Kee, J.-S. Kim, and S. Maeng, "Cost optimized provisioning of elastic resources for application workflows," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1011–1026, 2011.

[17] Y. Zhao, J. Dobson, I. Foster, L. Moreau, and M. Wilde, "A notation and system for expressing and executing cleanly typed workflows on messy scientific data," *SIGMOD Record*, vol. 3, no. 34, 2005.

[18] G. B. Berriman, J. C. Good, A. C. Laity, A. Bergou, J. Jacob, D. S. Katz, E. Deelman, C. Kesselman, G. Singh, M.-H. Su, and R. Williams, "Montage: A grid enabled image mosaic service for the national virtual observatory," *Astronomical Society of the Pacific Conference Series*, vol. 314, pp. 593–596, 2004.