# Fair multiple-workflow scheduling with different quality-of-service goals

Rezaeian, Amin; Naghibzadeh, Mahmoud; Epema, Dick H.J.

**DOI**
10.1007/s11227-018-2604-2

**Publication date**
2019

**Document Version**
Accepted author manuscript

**Published in**
Journal of Supercomputing

# Fair Multiple-Workflow Scheduling with Different Quality-of-Service Goals

Amin Rezaeian[a], Mahmoud Naghibzadeh[a,*], Dick H.J. Epema[b]

[a]*Department of Computer Engineering, Engineering Faculty, Ferdowsi University of Mashhad, Azadi Square, Mashhad, Iran*
[b]*Parallel and Distributed Systems Group, Faculty EEMCS, Delft University of Technology, Mekelweg 4, 2628 CD, Delft, The Netherlands*

**Abstract**

Cloud schedulers that allocate resources exclusively to single workflows are not work-conserving as they may be forced to leave gaps in their schedules because of the precedence constraints in the workflows. Thus, they may lead to a waste of financial resources. This problem can be mitigated by multiple-workflow schedulers that share the leased cloud resources among multiple workflows or users by filling the gaps left by one workflow with the tasks of other workflows. This solution may even work when users have different performance objectives for their workflows, such as budgets and deadlines. As an additional requirement, we want the scheduler to be fair to all workflows regardless of their performance objectives.

In this paper, we propose a multiple-workflow scheduler that is able to target different quality of service goals for different workflows and that considers fairness among different users. To this aim, we propose an unfairness metric and four workflow selection policies. We prove that the resource selection that decides based on a task's sub-budget, sub-deadline, finish time, and cost on different resources is selecting the best resource based on the given information, while using the smallest number of calculations. Simulations show that there is a trade off between overall cost, makespan, and fairness. We conclude that the best workflow selection policy to reduce unfairness is the direct policy, which explicitly selects the workflow that minimizes the value of the proposed unfairness metric in each round.

*Keywords:* Scheduling, multiple workflows, fairness, budget constrained workflow, deadline constrained workflow

## 1. Introduction

Today, many large-scale (scientific) collaborative distributed applications are designed as workflows. A suitable platform for the execution of these workflows is the cloud, which is utilizable through workflow scheduling methods. Whereas much research has been done on single-workflow scheduling methods, in order to reduce the monetary cost for users, when scheduling, it may be beneficial to consider multiple workflows. However, many users may encounter delays or even more costly schedule plans, because most of research done in this area focus on optimizing one criterion and the fairness is not considered. In this paper, we propose a fair multiple-workflow scheduler for workflows with known task runtimes, and evaluate it using simulation.

Scheduling workflows in distributed environments is an important topic in classical scheduling theory, and finding an optimal solution for it can be reduced to the multiprocessor scheduling problem, which has been classified as an NP-complete problem [1][2]. There are many methods which consider the problem of scheduling a single workflow on multiple resources [3][4][5]. Since workflows contain dependent tasks, task start times may be postponed until their preceding tasks are completed. Therefore, the schedule map of some resources rented in the cloud may contain idle time periods. Several researchers have used these idle times for scheduling more workflows on fewer resources, leading to multiple-workflow scheduling [6][7]. However, most of this research considers the optimization of only one metric such as makespan or cost for all workflows. In addition, only few researchers have considered fairness among the workflows [8][9]. In this paper, we consider both criteria of time and monetary cost as well as fairness.

The concept of resource sharing in multi-workflow scheduling on the cloud can be addressed from two different points of views. One point of view considers the perspective of optimizing the global quality of service (QoS) metrics, such as minimizing the overall execution cost. The second perspective has as its main objective the optimization of each individual user's QoS metrics. These two perspectives can have conflicting outcomes. For example, minimizing the overall execution cost of all the workflows can have a detrimental effect on some individual users' QoS metrics. In order to overcome this trade off, in this paper we propose a notion of unfairness, and a scheduling method for reducing it. It is considered that each workflow has an acceptable plan, and we want to merge these plans in order to find a better solution, i.e. improving their soft constraints. In this paper we assume workflows to have two constraints, viz., time and cost, each of which can be either hard or soft. The hard constraints are imposed as deadlines and budgets, while

*Corresponding author.
*Email addresses:* amin.rezaeian@mail.um.ac.ir (Amin Rezaeian), naghibzadeh@um.ac.ir (Mahmoud Naghibzadeh), D.H.J.Epema@tudelft.nl (Dick H.J. Epema)

the soft constraints must be optimized. Thus, for a deadline-constrained workflow, the execution cost must be minimized, while for a budget-constrained workflow, the makespan must be minimized.

In order to reach a fair schedule, each workflow must be scheduled considering other workflows. Therefore, we add a workflow selection step. After each workflow selection, we select a task from the workflow. There are methods that are proposed to order the tasks of a workflow [3]. In this paper, we divide the multiple workflow scheduling problem into two steps of workflow selection and task scheduling and propose a fast task scheduling technique besides four workflow selection policies. We test the performance of the proposed method using simulations. The role of using each policy is investigated in the tests. In addition, the effect of workload size, user defined constraints, the rate of budget constrained workflows against deadline constrained workflows in the workload, and several other measures are also inspected. In the experimental results, the earned fairness, total cost, and the level of loyalty to users' schedule are reported. In the experimental results we answer the questions of which workflows are suitable for and can participate in resource sharing, and how well the soft constraints of workflows can be met in relation with their hard constraints.

The problem we address in this paper is minimizing the unfairness when scheduling multiple workflows while maintaining the hard constraints and optimizing the soft constraints. The main contributions of this paper are the following:

- We define a multi-criterion notion of unfairness (Section 2).
- We propose a multi-workflow scheduler that takes fairness into account (Section 4).

The other contributions are:

- We propose four workflow selection policies (Section 4).
- We provide the necessary mathematical model to prove the main idea of the proposed scheduling technique works (Section 4).
- The role of the gaps inside each workflow plan and its influence on the final multi-workflow scheduling is concentrated on (Section 6).

## 2. Problem Definition

In this section we define the main assumptions and an abstract model for scheduling multiple workflows with different QoS requirements on a cloud infrastructure.

### 2.1. Resource model

In our model, cloud resources consist of processing devices connected via a network that can be leased for specific periods of time, e.g., multiples of one hour. Resources are assumed to be heterogeneous in that they may have different speeds, which are measured by the ratio of the processing capacity of each resource and that of the resource with lowest processing capacity.

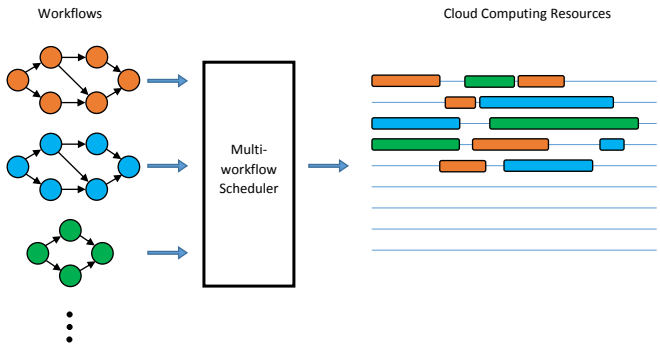



Figure 1: The system model with a scheduler scheduling multiple workflows on resources of a cloud.

The bandwidth of the connecting network is limited which causes communication latencies, though the bandwidth for tasks on the same resource is assumed infinite. The system is depicted in Figure 1.

### 2.2. Application model

In this paper we consider workflow applications that can be modeled as directed acyclic graphs (DAG), with nodes representing computational tasks and edges representing data dependencies. A task can only start executing after it has received all data from its predecessors. Tasks are considered rigid, and their runtimes vary in proportion to the speeds of the resources they run on. Every DAG has a single entry and a single exit node, which can be guaranteed by possibly adding dummy tasks with zero runtime.

#### 2.2.1. Hard and soft constraints of workflows

It is assumed that each workflow has either a budget or a deadline, set by the user, as a *hard constraint*; a workflow is called a budget or a deadline workflow accordingly. The makespan of a workflow is length of the time interval between the start time of its entry task and the finish time of its exit task. Similarly, the execution cost of a workflow is the overall billing cost of the resources over the rented time periods. For *budget workflows*, the scheduler must minimize the makespan while spending at most its budget. For *deadline workflows*, the cost of the schedule plan must be minimized while meeting the given deadline. We call the metric to be minimized for each workflow its *soft constraint*.

#### 2.2.2. Preliminary scheduling

Prior to scheduling the workflows collectively, each workflow's schedule plan without resource sharing is developed. This process is called *preliminary scheduling*, and its result is called the *reference plan*, which will be used during the final scheduling. In the reference plan, it is assumed that the cloud resources are enough to schedule each workflow. In other words, there is a scheduling plan to meet the deadline or budget of a workflow. The number and type of resources used by the

plan of each workflow may vary based on the workflow properties such as number of tasks, their interrelationship, the budget or deadline, etc. In order to determine the number of resources in the shared system used for scheduling all workflows, all the resources used in the individual reference plans are combined.

For a budget workflow ($w$), we indicate its budget by $B_w$, and in case of a deadline workflow, its deadline is indicated by $D_w$. The makespan and cost of the reference plan for $w$ are indicated by $R_w$ and $C_w$, respectively, while its makespan and cost in the shared system are denoted by $r_w$ and $c_w$.

#### 2.2.3. Cost calculation

A timeslot is the shortest period of time that a resource can be leased. In our model, the scheduler may map tasks of different workflows to the same timeslot, which will be called a shared. In that case, all participating workflows must pay for that timeslot. Since the amount of a timeslot workflows are using may vary from close to zero percent to the whole timeslot, we assume that the share of the cost of each workflow for a shared timeslot is proportional to the cost of the timeslot multiplied by the usage ratio of the tasks of that workflow in the timeslot. So, if $t_{i,c}$ is the amount of time used by tasks of workflow $w_i$ in timeslot $c$ with price $p_c$, the cost share for workflow $w_i$ is $p_c t_{i,c}/(\sum t_{j,c})$, where $\sum t_{j,c}$ denotes total usage of timeslot $c$.

### 2.3. Unfairness definition

In this paper, we also consider fairness. Since there are workflows with different quality of service requirements, and resource sharing is arbitrary (i.e. users are able to participate or not), the definition of fairness is different from that of Zhao et al. [8].

We calculate the unfairness using the speedup of the budget workflows and the savings of the deadline workflows. The *speedup* and the *savings* of a workflow $w$ are defined as $S_w = R_w/r_w$ and as $E_w = C_w/c_w$, respectively. The larger these values are, the more advantage the corresponding workflows benefit from resource sharing.

In order to maintain fairness, all speedups and savings must be relatively the same. Let $W_B$ be the set of budget workflows, let $W_D$ be the set of deadline workflows, and let $W$ be the set of all workflows. Then, defining

$$\overline{A} = \frac{\sum_{w \in W_B} S_w + \sum_{w \in W_D} E_w}{|W|} \tag{1}$$

as the average resource sharing benefit, the *unfairness* $U$ is defined as

$$U = \frac{\sum_{w \in W_B} |S_w - \overline{A}| + \sum_{w \in W_D} |E_w - \overline{A}|}{|W|}, \tag{2}$$

which is equal to zero when all the $S_w$ and $E_w$ are equal.

The problem we want to address in this paper is minimizing the unfairness $U$ while satisfying the hard constraints and optimizing the soft constraints of the workflows.

## 3. Background and Related Work

Scientific workflows are usually represented by Directed Acyclic Graphs (DAG) which consist of tasks as vertices. The edges in the workflow reveal the data dependencies and the precedence constraints between the interrelated tasks. Some researchers have also modeled scientific workflows as Hybrid DAGs in which super-tasks can be composed of tasks that may have data interactions during executions [10].

The vast majority of researchers in Cloud computing area have focused on single workflow scheduling algorithms in order to satisfy the users QoS constraints, namely deadline and budget [11][4][12]. Furthermore, there are algorithms that have been proposed to schedule multiple workflows [8][9].

One of these algorithms is BHEFT which is based on budget distribution. This is a list-based scheduling approach to schedule single workflow, and it is developed by Zheng et al. [11] with the aims of minimizing makespan so that the users' budget and deadline constraints are met. In each step, BHEFT estimates a tasks sub-budget. This estimation is done based on the remaining budget after the assignment of the previous tasks and the average budget required for scheduling the unassigned remaining tasks. Then, the algorithm minimizes the execution time of the task under the tasks sub-budget constraint (based on the HEFT method). Finally, if the budget is sufficient to schedule all of the tasks, the workflow application is admitted, otherwise the scheduler rejects it.

In contrast, in [4], a cluster-based scheduling method to minimize monetary cost under deadline constraint is proposed. Their solution has used Partial Critical Path (PCP) length to cluster tasks and assigned it to a single instance which can schedule all of the cluster's tasks before their calculated latest finish time (LFT). The main idea of PCP is to reduce communication cost by grouping the tasks of a critical path as a labeled cluster.

Bittencourt and Madeira used horizontal clustering to address the workflow scheduling problem [9]. In their approach, they have used Path Clustering Heuristic (PCH) [12] to schedule more than one workflow at the same time on grid environment. Authors introduced four strategies of sequential scheduling, gap search scheduling, interleave, and group DAGs. The interleaving algorithm tries to interleave pieces of each of DAGs in a round robin manner. They illustrated that this strategy causes to minimize the workflow execution time and achieve better fairness among four strategies for scheduling multiple workflows. However, they have not considered the fairness as an optimization criterion in their scheduling algorithms.

In [8], Zhao and Sakellariou presented two fairness-based algorithms for static scheduling of multiple DAGs. In the first step, they defined the slowdown parameter as the ratio between the workflow execution time, when scheduled alone and when scheduled with other DAGs. In this strategy, the fairness criterion is determined based on the slowdown parameter. Next DAGs are sorted in ascending order of their slowdown parameter and select unscheduled task from DAG with minimum slowdown value. For mapping phase, they have used list schedul-

ing algorithms like HEFT. At each scheduling round, two fairness policies are considered for optimizing makespan and fairness: re-computing the slowdown parameter based on finish time (FPFT) for the selected DAG, and based on current time (FPCT) for all DAGs.

With the illusion of unlimited resources, each user may reach the desired resources for a reasonable workflow size [13][14]. However, it might be either too expensive or too time consuming to reach a predefined constraint, and there may be lots of idle periods in the schedule plan. Therefore, a multi-workflow scheduler which is able to utilize the idle periods of different plans and make more affordable schedules gains higher importance in this context.

These methods are proposed for the utility grid environment and none of them takes into account the cloud features such as monetary costs based on pay-as-you-go model, timeslot pricing policy and VM heterogeneity. In addition, fairness of sharing resources among multiple workflow applications has not received the deserved attention.

## 4. Creating a Schedule

In order to address the problem of scheduling multiple workflows in a way that meets the hard constraints of all workflows while considering fairness, we propose a method that consists of four steps. First, we determine the sub-deadline and the sub-budget of all tasks of all workflows, based on their preliminary schedule plans. Then we repeatedly select a workflow and a task from that workflow, and finally we select a cloud resource for that task. The first step is done only once at the beginning, while the remaining three steps are repeated until all tasks of all workflows have been scheduled. The scheduling process is shown in Figure 2.

### *4.1. Determining sub-deadlines and sub-budgets*

In the first step, preliminary scheduling on the empty reference system is done for every workflow in order to obtain its reference plan.

For budget workflows, we apply a modified version of the BHEFT scheduling algorithm [11], which is able to deal with timeslot-based resource costs. In order to assign a sub-budget to Task $t$ scheduled on the Resource $p$, we use the ratio $r$ of the task's weight $h_t$ to the sum of all tasks' weights on $p$, which can be written as

$$r = h_t / \sum_{h_p \in tasks_p} h_p, \tag{3}$$

where $tasks_p$ denotes the set of tasks scheduled on Resource $p$. Then the sub-budget of $t$ is assigned by $(rO_p/C_w)B_w$, where $O_p$ is the total cost of Resource $p$, $B_w$ is the budget of workflow $w$, and $C_w$ is the cost of the schedule of the BHEFT algorithm. As resource sharing is used to improve the soft constraints, the finish times of the tasks in the reference plan are stored as their sub-deadlines.

For deadline workflows, we use the IC-PCP algorithm [15]. This algorithm computes *latest finish time* of each task ($LFT$) based on the user-defined deadline. We use these values as sub-deadlines of tasks. Users expect to reach a better plan via resource sharing, thus sub-budgets are also calculated. Since there is no budget, $rO_p$ is considered as the sub-budget of Task $t$.

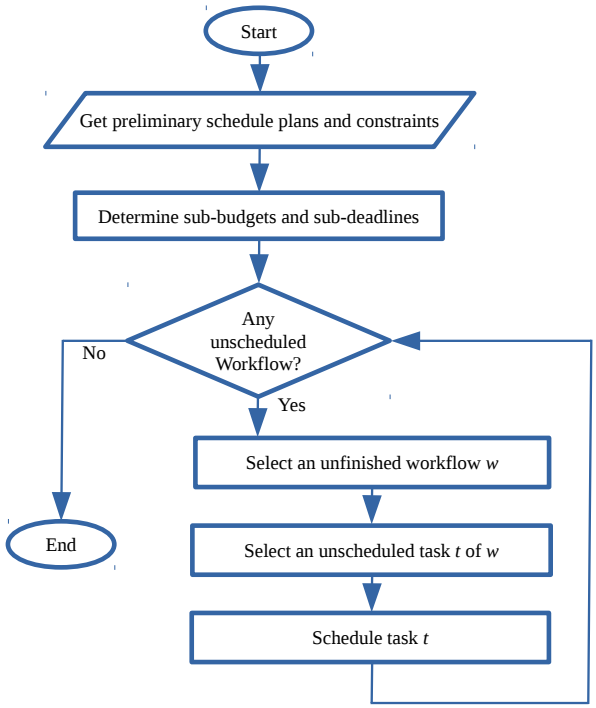



Figure 2: The proposed scheduling process

### *4.2. Workflow selection*

After the calculation of sub-budgets and sub-deadlines, a loop of scheduling tasks is started. At the beginning of each iteration, a workflow is selected. We propose the following four workflow selection policies.

#### *4.2.1. Ordered workflows*

In order to apply this policy, a unique number is assigned to each workflow. Then, in each iteration, the unfinished workflow with smallest number is selected. Effectively, this means that a workflow is selected and all of its tasks are scheduled before the scheduler goes to the next workflow.

#### *4.2.2. Round-Robin*

In this policy, a queue of unfinished workflows is maintained by the scheduler. Iterating over workflows in the queue, workflows are selected one by one and for each of them one task is scheduled. In order to reduce the unfairness, the Round-Robin (RR) policy interleaves the tasks, not parts of a DAG as proposed in [9].

#### *4.2.3. Weighted Round-Robin*

In order to compare workflows according to their resource utilization, the fraction of idle time in a resource is defined as its gap ratio. The gap ratio of a workflow schedule plan is defined as the weighted average of the gap ratios of the resources

in this plan, with the weights of the resources equal to their relative speeds. Presumably, workflows with higher gap ratios in their reference plans can take more advantage of resource sharing, and workflows with lower gap ratios tend to be scheduled without resource sharing. In addition, a workflow which rents an instance for the duration of a timeslot and pays for it, tends to keep it and schedules its own tasks. Workflows take advantage of resource sharing only when a gap appears.

We now propose a weighted round robin (WRR) workload selection policy which considers both gap ratios and timeslots. To this aim, for each workflow $w$, its gap ratio $g_w$ is defined as the average gap ratio among its rented reference resources. The weight of workflow $w$ is defined as $s_w = 1/g_w$. In addition, let $\min_s$ be equal to the smallest value of the $s_w$. In every iteration, the workflow with the largest value of $s_w$ is selected. Then, its $s_w$ is decreased by $\min_s$. When all $s$ values become negative, they are increased by their $1/g_w$.

In addition, if during a workflow task assignment a new timeslot is rented, its next tasks can be scheduled with high priority while they are scheduled in current timeslot of the current resource.

#### 4.2.4. Direct Heuristic

This policy is guided by the definition of unfairness (2). In fact, the policy selects the workflow that maximizes the value of $U$. However, that equation requires the speedups and savings of the current plan, which are not available until the scheduling finishes. So we propose heuristics to estimate the partial speedups and the partial saving s of the current plan.

To calculate the partial savings of a workflow, its current cost $c_w$ is computed as the sum of the shares of its tasks on each resource multiplied by their prices, and the sum of the sub-budgets of its scheduled tasks is taken as $C_w$. Now, $E_w$ is computed as $C_w/c_w$.

In order to compute the partial speedup of a workflow, let $h$ be the set of its tasks scheduled in the current plan that do not have a successor scheduled in that plan. Now instead of $R_w/r_w$, the partial speedup of workflow $w$ is computed as the average ratio of the task sub-deadline $sd(t)$ in the reference plan and the task finish time $ft(t)$ in the current plan across the tasks $t$ in the set $h$:

$$S_w = \frac{\sum_{t \in h} \frac{sd(t)}{ft(t)}}{|h|}. \quad (4)$$

Prior to starting the scheduling process and using the above estimations, one task of each workflow is scheduled to prevent $h$ to be empty.

### 4.3. Task selection

In the third step, the selection of a task is done. Task selection is done using upward rank [3]. A task with a longer critical path (in terms of both execution time and communication time) to the end of workflow gets a higher upward rank than a task with the shorter one. The end of a workflow is reached when the last task with no child is completed.

### 4.4. Task scheduling

In the final step, the selected task is scheduled on the cloud's shared resources. For selecting a resource for the task, we use different policies for budget workflows and deadline workflows.

For budget workflows, the affordable cloud resource (which does not violate sub-budget) with the earliest finish time is selected. This selection causes to have a schedule with the same (or lower) budget which finishes earlier. If an affordable resource is not found, the cheapest resource that meets the deadline is selected.

For deadline workflows, the cheapest resource that meets the task's sub-deadline is selected. Here again, if a resource is not found that meets the deadline, the resource which finishes the task closer to the deadline is selected.

It is possible that no resource meeting the above mentioned conditions is found. In order to select then the most appropriate resource, we test the scheduling of the selected task on each of the resources that have already been leased along with one resource from each resource type. This gives the scheduler the opportunity to easily select between using an already leased resource and a new resource.

When comparing two resources as candidates for a task, either one of them is better in finish time and the other in cost, or one of them is better (or at least equal) in both criteria of finish time and cost, in which case the decision is clear. In order to distinguish all situations when comparing two resources, which are listed in Table 1, we denote the sub-deadline and the sub-budget of a Task $t$ and its finish time and cost on Resource $r_i$ by $sd(t)$, $bt(t)$, $ft_i(t)$ and $ct_i(t)$, respectively. The last column of the table shows the decision, which is the index of the selected resource.

Row 5 of the table is the "easy" case when Resource $r_i$ is better in both criteria than Resource $r_j$. Rows 1–4 cover all cases when Resource $r_i$ ($r_j$) is better than Resource $r_j$ ($r_i$) in finish time (cost). In the situations in Rows 1 and 2, both resources meet both constraints, and the decision then depends on the type of the workflow. In Row 1, In case of deadline workflow, since the deadline is met, the less cost effective resource is selected. The same is also done for budget workflows in Row 2, whereas the resource with the earlier finish time is selected. In the situation in Row 3, both resources still satisfy the sub-deadline, but at least the most costly Resource $r_i$ exceeds the sub-budget. Then the cheapest resource is preferred. Row 4 covers the remaining situations in which the slowest Resource $r_j$ exceeds the sub-deadline, while the most costly Resource $r_i$ may or may not exceed the sub-budget. We assume that the sub-deadline of a task is more important than its sub-budget, and so, violating the sub-budget of a task is allowed when this is needed to meet its sub-deadline, or to finish at the earliest after its sub-deadline. Therefore, in the situations of Row 4, Resource $r_i$, which finishes the task earlier, is selected.

In order to avoid mutually comparing all resources, which would lead to a complexity of order $O(n^2)$, we use the selection process to create a transitive partial order on the set of candidate resources to schedule a task. In many methods such as HEFT [3], PCH [16] and IC-PCP [15], only one criterion (finish time)

Table 1: Resource selection decision for a task based on its constraints and its cost and finish time on two resources.

| Row | Situation | Decision |
|---|---|---|
| 1 | $ft_i(t) < ft_j(t) < sd(t)$ and $ct_j(t) < ct_i(t) < bt(t)$ | $i$ (budget wf) |
| 2 | $ft_i(t) < ft_j(t) < sd(t)$ and $ct_j(t) < ct_i(t) < bt(t)$ | $j$ (deadline wf) |
| 3 | $ft_i(t) < ft_j(t) < sd(t)$ and $ct_j(t) < ct_i(t)$ and $bt(t) < ct_i(t)$ | $j$ |
| 4 | $ft_i(t) < ft_j(t)$ and $sd(t) < ft_j(t)$ | $i$ |
| 5 | $ft_i(t) \leq ft_j(t)$ and $ct_i(t) \leq ct_j(t)$ | $i$ |

is used to select the best resource, while the decision making in the present paper is based on both time and cost.

Using a transitive relation, a Resource $r_i$ that is found in a linear search is better than all other resources, say for any Resource $r_x$, we have $r_i$ is preferred over $r_x$. In other words, it is enough to compare the candidate Resource $r_i$ with a resource that is preferred over some resources, to be sure that we can prefer $r_i$ over all of them or not. Below we define the preferred relation, and next we prove that it is transitive.

**Definition 1.** *The preferred relation $\preceq_t$ on the set of resources on which Task t can be scheduled, written $r_i \preceq_t r_j$, is defined by Resource $r_i$ being preferred over Resource $r_j$ according to Table 1.*

We now show that the preferred relation is transitive.

**Theorem 1.** *The preferred relation is transitive.*

*Proof.* Assume that for Task $t$ of a workflow $r_i \preceq_t r_j$ and $r_j \preceq_t r_k$ are decided based on Rows $p$ and $q$ of Table 1, respectively. If $q$ is one of Rows 1 to 4, then $r_i \preceq_t r_k$ is decided based on either Row 5 or Row $q$. If $q$ is Row 5, then $r_i \preceq_t r_k$ is decided based on either Row 5 or Row $p$. The only exception is for $p = 3$, which the decision can also be made on Row 4. □

Since the task scheduling policy defined above is proven to be efficient, we fix it for the rest of the paper, and we evaluate the different workflow selection policies to see how they affect fairness.

## 5. Experimental Setup

In this section, we describe the simulation environment and applied parameters to workloads and the scheduler.

In the reference system, we consider two types of resources $a$ and $b$. The speed of Type $a$ is half of the speed of Type $b$ and the price of Type $b$ is three times the price of Type $a$ [11]. We assume that the reference system includes 100 resources of each type. Timeslot lengths of the resources are selected to be 5 or 60.

The speed, price, and timeslot of resources in the shared system is set to the same values as in the reference system. In order to set the amount of resources in the shared system, the reference resources used for scheduling the examined workflows are combined. In order to set a scale on using combined resources, we defined a parameter called *resource factor*. It is defined as the fraction of combined resources that is used in the shared system. The number of combined resources is multiplied by *resource factor*, to determine the number of resources. The *resource factor* ranges between 0.4 and 1.2.

We use different workload sizes to examine how the workload size affects the scheduler performance. Each workload contains $l$ workflows, where $l = 10, 20, 30, ..., 160$. All workflows of the workload are present at the start time. The *budget-ratio* is defined in each workload as the ratio of budget workflows to the total number of workflows. This factor is assumed to be 0, 0.5 or 1.

In order to get realistic results, we use scientific workflows of different domains. We select bioinformatics (Epigenomics and SIPHT workflows [17], [18], [19]), astronomy (Montage workflow), gravitational physics (LIGO workflow) and earthquake science (CyberShake workflow). The Epigenomics workflow is a data processing pipeline which automates various genome sequencing operations. SIPHT workflow automates the search for sRNAs. Montage [20] workflow is used to process several images of the sky in order to make a final mosaic, and calculate the geometry of the output image from the input images. LIGO Inspiral workflow is used to detect gravitational waves [21][22]. CyberShake workflow is used to apply the Probabilistic Seismic Hazard Analysis (PSHA) technique, in order to characterize earthquake hazards [23].

In order to generate random workflows, we use the WF generator [24]. The processing load consists of scientific workflows of Montage, Epigenomics, Cybershake and LIGO Inspiral with equal chances of appearance. We also have done experiments with the SIPHT workflow.

For the total execution time of workflows, we use the two-stage hyper-Gamma distribution that is also used in [25]. The shape and scale parameters $(\alpha, \beta)$ of two component Gamma distributions are set to (5.0, 501.266) for 30% of the workload and (45.0, 136.709) for the rest. We normalize the runtimes of the tasks of each workflow in such a way that its overall runtime is equal to the processing load specified by the hyper-Gamma random generator.

The size of the workflows varies uniformly between [30,38] and [90,138]. The average task runtime on a Type $a$ resource is 26.4 with standard deviation of 37.8. These sizes are selected for two reasons. First, because of the type of work performed by each workflow, its possible sizes are limited to certain values. Second, the proposed method applies fairness and improves QoS constraints using the aggregation of tasks' improvements. These improvements are achieved by finishing a task earlier than its sub-deadline and/or cheaper than its sub-budget. These are very tight for workflows with lots of tasks, which restricts the improvement. Therefore we avoid very large workflows.

As performance metrics, we use unfairness, normalized makespan, normalized cost and utilization. The normalized makespan of a workflow is defined as its makespan on the shared system divided by its makespan on the reference sys-

tem. The same holds for the normalized cost. We measure the utilization with the average gap ratio of all resources because it is important to consider only their rented timeslots. All these metrics must be minimized.

Prior to applying the policies, the preliminary scheduling is done, and the budget and deadline workflows are scheduled on the reference system using the BHEFT and IC-PCP scheduling methods, respectively. We use two types of deadlines, tight and loose [26][27]. For tight deadlines, the HEFT makespan (which doesn't consider the budget) of each workflow is calculated and multiplied by values that are selected from the normal distribution with average 2 and standard deviation 1.4. For loose deadlines, the average and standard deviation are 8 and 3, respectively.

For budget workflows, tight and loose budgets are considered. For tight budgets, the cost of scheduling all tasks of each workflow in a Type *a* resource is multiplied by a sample from the normal distribution with mean and standard deviation 2 and 1.4 respectively. For loose budgets, the average and standard deviation are 8 and 3, respectively.

In the workloads we use, 20% (80 %) of the workflows have a tight (loose) constraint (deadline or budget). However, only deadlines and budgets are accepted that are met by the reference scheduling methods, and the rest are considered non-realistic and omitted.

## 6. Experimental Results

In this section, we examine the impact of the timeslot length, the resource factor, the workload size, and the budget-ratio on the performance of the proposed polices. Except for the experiments with the workload size, the simulation is done using 8 workloads consisting of 80 workflows each. These are randomly generated as described in the previous section. The averages of the results are reported. Since this paper focuses on studying the requirements of fair scheduling, we analyse factors with more influence on fairness.

### *6.1. Timeslot length analysis*

With the first experiment, we investigate the impact of the timeslot length on the performance. On the reference system, the average gap ratio for budget and deadline workflows with timeslot 60 are 0.10 and 0.12, respectively. The corresponding values are much lower for timeslot 5: 0.009 and 0.012. This can be explained by the task size. When the average task size is longer than the timeslot, almost all of the timeslots are utilized by tasks, therefore fewer gaps are formed. This makes it hard and even unnecessary to improve the schedule plan via resource sharing.

In Figure 3, the gap ratio as a function of the resource factor for timeslot lengths of 5 and 60 is shown. For timeslot 5, the average gap ratio of the reference resources is 0.01. It is shown that it is decreased to 0.003 for the shared system. It means that gaps are shrunk 0.7% in average, which is hardly an improvement. Therefore, all remaining tests are done using timeslot 60.

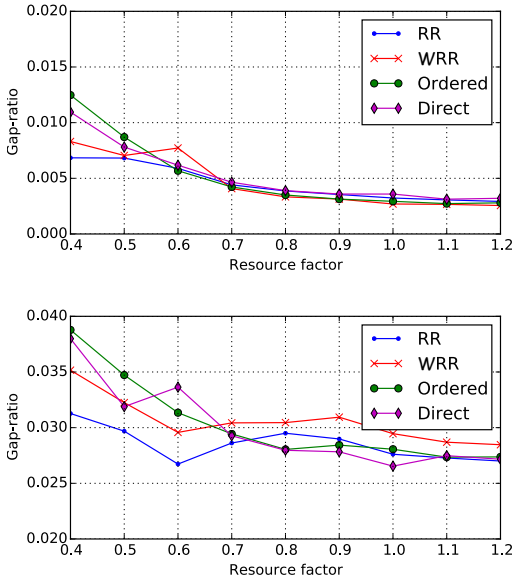



Figure 3: The gap ratio as a function of the resource factor for a timeslot length of 5 (top) and 60 (bottom) for different scheduling polices. RR and WRR are abbreviations for Round Robin and Weighted Round Robin, respectively.

### *6.2. Cost and makespan analysis*

In this section, we examine the impact of the resource factor on cost, makespan, and unfairness.

#### *6.2.1. Cost- and makespan-resource factor dependency*

Figure 4 shows the normalized makespan (top) and the normalized cost (middle) as functions of the resource factor. Regardless of the scheduling policy, it is observed that there is a trade-off between makespan and cost. When the number of resources in the shared system is limited, the cost is lower but the makespan is higher, and vice versa. The reason is that when resources are not sufficient, some tasks that are scheduled on fast resources in the reference plans, are forced to be scheduled on slow ones. This causes reduction in cost and increase in the makespan. On the other hand, when the resources offered by the shared system are more than by the reference system, it is more likely that tasks with slow resources in the reference plans are absorbed by the already rented fast resources. This causes shorter makespans and more costly plans. However, there is point of resource factor, on which the performance metrics are desirable. Based on the desired performance, this point can be selected from the given figures.

#### *6.2.2. Resource utilization analysis*

Another observation in cost in Figure 4 (middle) is that all policies are taking advantage of resource sharing and they decrease the average overall cost. The reason for this decrement can be found via the resource utilization. Figure 3 (bottom) shows the gap ratio as a function of the resource factor. It is observed that for all proposed policies, as the resource factor is increased, the gap ratio is decreased. It can be explained from the perspective of tasks' runtimes and freedom to select

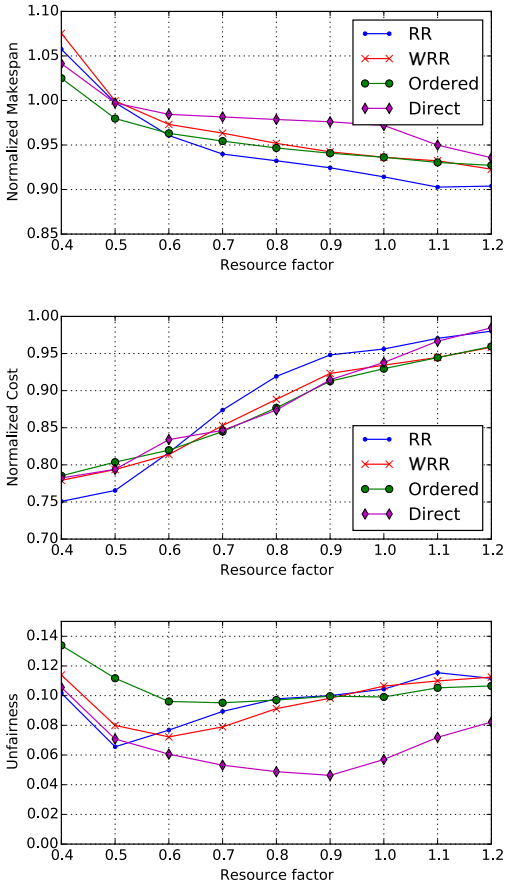



Figure 4: The normalized makespan (top), the normalized cost (middle) and the unfairness (bottom) as functions of the resource factor

resources. The average runtime of tasks on Type *a* and Type *b* resources are 13.2 and 6.6, respectively. Therefore, for timeslot 60, the chance to utilize a timeslot of a Type *b* resource is more than a Type *a* resource. Since the number of resources grows by increasing the resource factor, the chance of the scheduler to select a better resource is increased. The cost of using an already rented timeslot for a new task is zero for the scheduler. Therefore, it is very likely to schedule a task in an already rented timeslot, instead of launching a new one. Since faster resources are more likely to be selected by increasing the resource factor, it also explains the decrease in the makespan.

Another explanation for improving the cost more than the makespan is that both task scheduling policies (for deadline and budget workflows) are designed to meet the deadline as their first priority. Then, between affordable resources they select the one which suits them the most.

### *6.2.3. Unfairness-resource factor dependency*

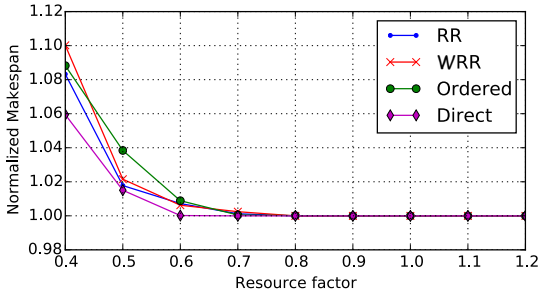
In Figure 4 (bottom), the unfairness as a function of the resource factor is shown. As expected, the Direct policy shows better performance in this metric. However, for all policies, a minimum point of unfairness can be found. For all policies, when there is not enough resources, the chance to rent an appropriate resource decreases for all workflows. In order to describe this situation, consider the scenario when the QoS requirements of a task are met better with a new resource, and the scheduler leases the resource. The chance to find an eligible resource between unleased resources decreases when the total number of resources is decreased. In addition, the lack of resources causes newly leased resources to be more eligible for all workflows than current available resources. However, only limited number of workflows earn the chance to lease better resources. This increases the unfairness criterion.

On the other hand, when there are too many resources, it is easy for the scheduler to find an eligible resource for every task, but outstanding resources are also found rarely. For example, some workflows take more advantage of earlier rented timeslots of faster resources. Parts of these resources are rented by other workflows previously, and they are not available for every workflow to use. This causes the unfairness to rise.

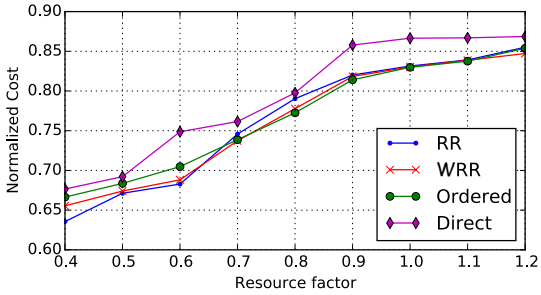
Based on the given result, when the resource factor is near one, i.e 0.9, the Direct policy is more fair and the unfairness criterion is at its lowest value.

## *6.3. Budget-ratio analysis*

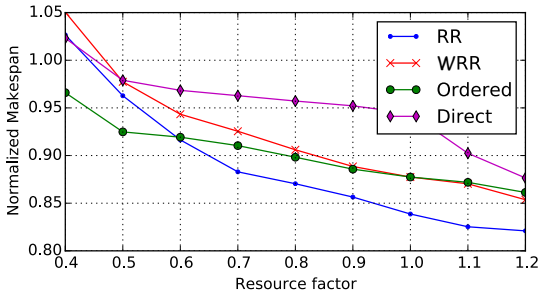
In this section, we examine the impact of budget-ratio on the scheduling metrics. Budget-ratio is set to determine the ratio of budget and deadline workflows. Budget-ratio is inspected to reveal the impact of type of constraints on the scheduling. In addition to the impact of the resource factor, we notice that the makespan and cost for deadline and budget workflows are different. In Figure 5, the normalized makespan and the normalized cost as functions of the resource factor, for budget and deadline workflows are shown separately.



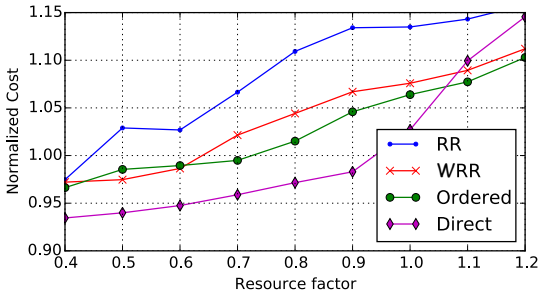

(a) The makespan for budget workflows




(b) The cost for budget workflows




(c) The makespan for deadline workflows




(d) The cost for deadline workflows

Figure 5: The normalized makespan and the normalized cost as functions of the resource factor, for budget and deadline workflows.

*6.3.1. The performance of budget workflows versus deadline workflows*

Considering Figures 8a and 8b leads to an inspiration for the system setup for workloads consisting of only budget workflows.

For example, resource factors of 0.8 or 0.9 give the best minimization in cost and meets the reference plans' makespans in Figures 8a and 8b. We set budget-ratio to 1.0 and resource factor to 0.9. The normalized cost is changed as a function of the workload size. However, for workloads with sizes greater than 10, the changes in the normalized cost is very short, i.e. standard deviation is 0.005. The mean of this set is 0.87. As it is expected, the normalized makespan also has no obvious changes, and its mean and standard deviation are 0.9999 and $1.2 \times 10^{-5}$, respectively. However, the cost is improved 13%. It seems that this much reduction is the bound of the cost improvement, and the explanation can be found within the resource utilization. For workloads with sizes greater than 10, gap ratio values do not change significantly, i.e. mean and standard deviation are 0.034 and 0.0005. Therefore, it seems that for this set of tests the highest possible resource utilization is reached and further improvement is not possible.

For budget workflows, unfairness is measured based on their makespan, and since there is no improvement in makespans for all workflows, the calculated unfairness remains too low. Mean and standard deviation are $3.6\times10^{-10}$ and $2\times10^{-5}$, respectively. However, considering fairness, again the Direct policy performs about 50 percent better than other policies.

For scheduling deadline constrained workloads, we consider Figures 8c and 8d. It seems that resource factor 0.5 gives the best minimization of cost, while meeting the deadline.

*6.3.2. Summarizing budget-ratio analysis*

Although the hard constraints are not the aim of the minimization, in both cases of deadline and budget workflows, the hard constraint is decreased more than the soft constraint. The explanation for this behaviour can be found via the sub-deadline and sub-budget assignment process. For budget workflows, sub-deadlines are defined unchangeable, while sub-budgets are changed if a task consumes less cost than its sub-budgets. For deadline workflows, although task sub-budgets are updated via the scheduling, but their deadlines are independent from the reference scheduling, and they are assigned using the *latest finish time* process [15].

*6.4. Workload size analysis*

In this section, we inspect the impact of workload size on the performance metrics. The normalized makespan, the normalized cost and the gap ratio as functions of the workload size, for deadline constrained workloads are shown in Figure 6. The normalized makespan is close to 1.0 again, but only the Direct and Ordered policies meet the deadline. The reason for this is that in order to attain better cost minimization, we select a small resource factor value. As the size of the workload grows, the makespans of different policies do not change, but their costs are decreased. This also is explained via the resource utilization.

The gap ratio is decreased by increasing the workload size. This happens because the average gap ratio of deadline workflows are greater than that of the budget workflows. In other

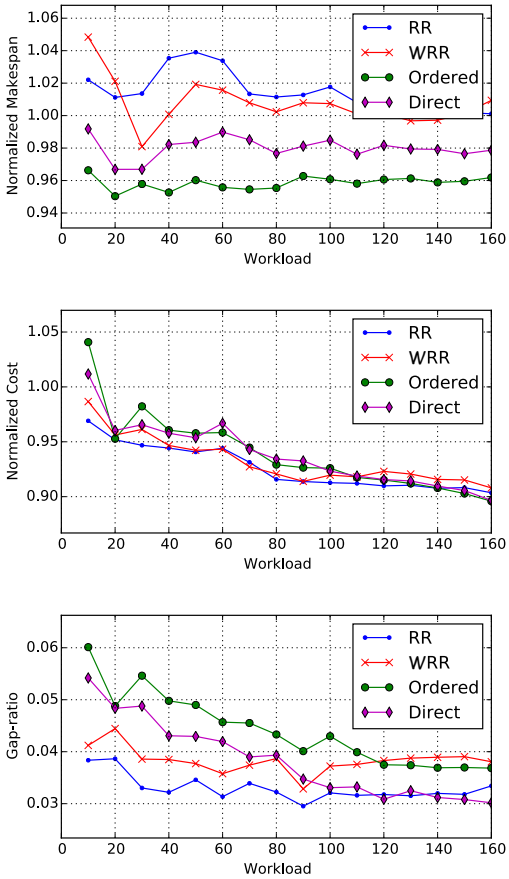



Figure 6: The normalized makespan (top), normalized cost (middle) and gap ratio (bottom) as functions of the workload size, for budget-ratio=0 and resource factor of 0.5

words, deadline workflows' reference plans contain more gaps and therefore, they can take more advantage of resource sharing than the budget workflows.

From the fairness perspective, it is shown in Figure 7 that the WRR and RR policies perform better. It can be explained by the trade-off between makespan, cost, and the fairness. Looking over all figures, it is revealed that there is a trade-off between these metrics. In all of our experiments, whenever a policy is better in two metrics, it loses the third. Therefore, although WRR and RR policies give smaller unfairness and almost the same normalized cost as others, they don't meet the deadline.

### *6.4.1. Mixed workload analysis*

We also analyse the impact of the resource factor, when budget-ratio is 0.5. We try to find the resource factor which gives better performance and lower unfairness.

According to Figure 4 (c), when the resource factor is 0.9, the lowest unfairness is achieved. Although, from the performance perspective, it is not useful. The normalized makespan (a), the normalized cost (b), and the unfairness (c) as functions of the workload size, for budget-ratio equal to 0.5 and resource factor of 0.9 are shown in Figure 8. Although the Direct policy outperforms other policies in fairness and cost metrics, it has

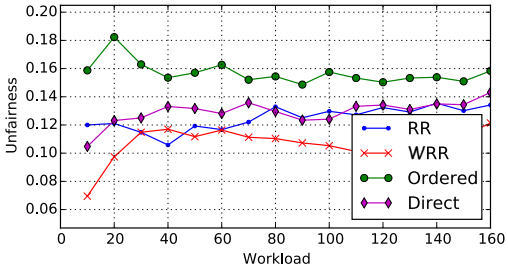



Figure 7: The unfairness as a function of the workload size, for budget-ratio=0 and resource factor of 0.5

higher makespan than the others. Again, the trade-off between three metrics is observed. For the Direct policy, since it maintains the normalized makespan close to 1, and the impact of increasing the workload size is decreasing in normalized cost, we expect the increase in unfairness in our experiments.

It is also observed that increasing the workload from 10 to 60 causes the normalized makespan for almost all of the polices to be improved in Figure 8 (a). It shows the positive effect of resource sharing on the schedules with more workflows. However, for larger workloads, the high competitions to grab the best resources stops the improvement.

The percentage of workflows that fulfilled their QoS requirements (budget and deadline) are shown in Figure 8 (d). It is calculated by dividing the number of successful schedules by the total number of schedules. In case of failing the QoS requirements, the preliminary schedule plans of the workflows are applied. Almost all policies perform the same for relatively large workloads. However, for smaller workloads, since the resource sharing is still limited, the chance of having unsuccessful schedules increases.

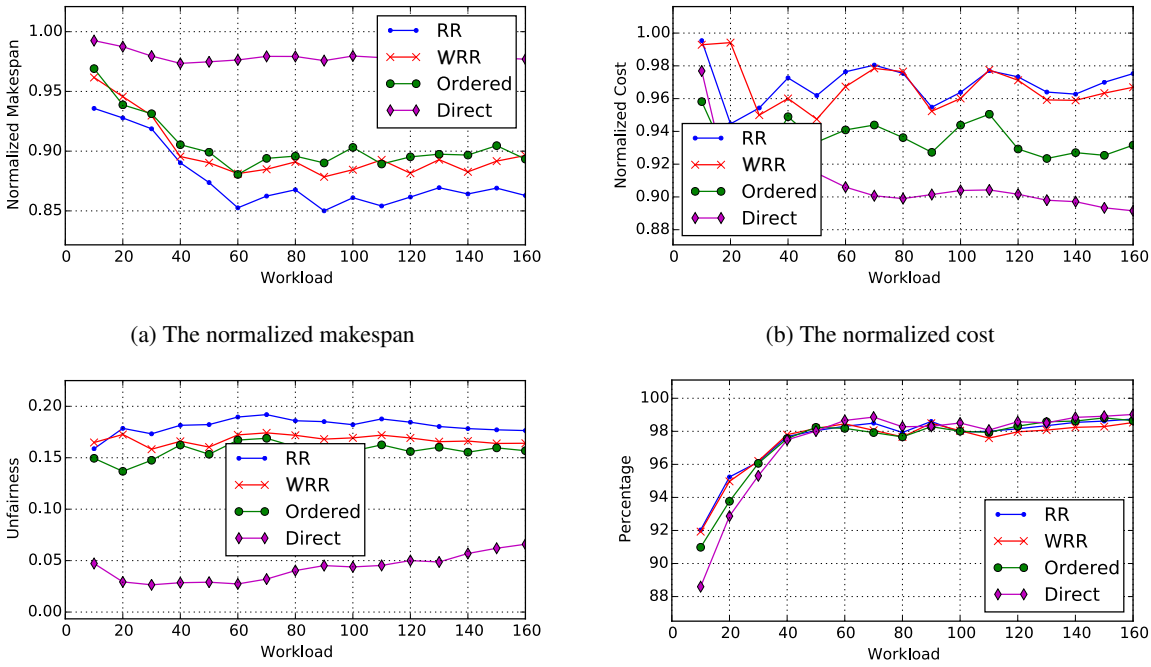
### *6.4.2. Minimizing cost and makespan*

In order to find a resource factor which serves for both budget and deadline workflows, we select the mean of the resource factors that are found before. Those resource factors are 0.5 and 0.9 for deadline and budget workflows, respectively. The normalized makespan , the normalized cost and the unfairness as functions of the workload size, for budget-ratio equal to 0.5 and resource factor of 0.7 are shown in Figure 9. In this experiment, although direct policy shows a great decrease in unfairness, it doesn't perform well in the makespan. The best overall performance is achieved using WRR and ordered policies.

On the other hand, we compare the results in Figure 9, with resource factor equals 0.7 in Figure 5. Despite the mixed selection of budget and deadline workflows, the results are similar to the results of deadline workflows, shown in Figure 5 (c) and (d). It is caused by the preference of the deadline over budget in task scheduling algorithm.

## *6.5. Workflow type analysis*

In this section, we inspect the impact of workflow types on the fairness and the performance of the scheduling. The gap ratio of different workflow types on timeslots of 5 and 60 are




(a) The normalized makespan

(b) The normalized cost

(c) The unfairness

(d) The percentage of workflows that fulfilled their QoS requirements

Figure 8: The normalized makespan (a), the normalized cost (b), the unfairness (c), and the percentage of workflows that fulfilled their QoS requirements (d) as functions of the workload size, for budget-ratio=0.5 and resource factor of 0.9.

Table 2: The gap ratio of different workflow types on timeslots of 5 and 60

| Workflow type | timeslot 5 | timeslot 60 |
|---|---|---|
| SIPHT | 0.003 | 0.038 |
| Cybershake | 0.008 | 0.082 |
| Epigenomics | 0.010 | 0.083 |
| Montage | 0.011 | 0.118 |
| LIGO Inspiral | 0.010 | 0.130 |

provided in Table 2. The results indicate that the gap ratio for SIPHT workflow is between one third to one half of the other workflows. The reason is that the SIPHT workflow consists of several tasks with very short execution times. These tasks are finished in minutes. However, the workflow includes one to three tasks with long execution times which take hours to be finished. This caused the SIPHT schedule plan to be very tight.

We omit the SIPHT workflow from our experiments, because its gap ratio is too low. In other words, in comparison with other tested workflow types, it doesn't leave that much gaps that can be used during the resource sharing.

## 7. Conclusion

In this paper, we have proposed a multi-criterion multi-workflow scheduler and we have studied the effect of four workflow selection policies on the fairness, the overall cost and the makespan. In order to evaluate the proposed method, in addition to unfairness calculation, we examined the changes in makespan and cost of the workflows, with and without the resource sharing. We also inspected the effect of the number of scheduled workflows in achieving fairness and decreasing the cost and makespan. Our first conclusion is that there is a trade-off between three factors of overall cost, makespan, and the fairness.

We also measured the gaps in the preliminary schedule of each workflow, and examined their effect in the final multi-workflow scenario. We found that workflows with larger gap averages take more advantages from multi-workflow scheduling, and they can save more budget and finish earlier than workflows with smaller gap ratios. Our simulation shows the same condition when time-slots are small. Therefore, the shorter the time-slots, the less fair is the multi-schedule.

As another conclusion, we showed that in presence of more than one criterion, the most suitable resources cannot be found using a linear search, unless it is proved that the comparing operator is transitive.

The empirical analysis shows that using the direct workflow selection policy decreases the unfairness in almost all cases.

As future work, we plan to change the scheduler to work dynamically. Although a reference schedule plan is necessary for the current method, we intend to determine the sub-deadline and sub-budget analytically.

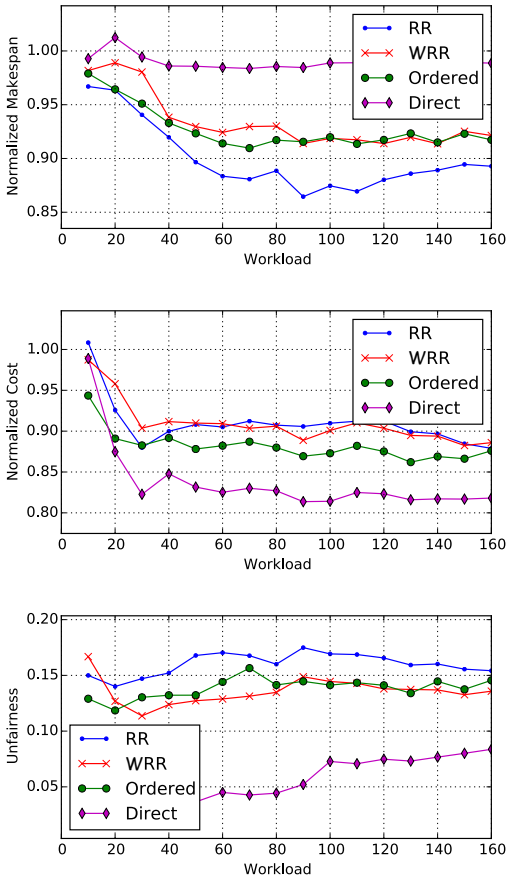



Figure 9: The normalized makespan (top), the normalized cost (middle) and the unfairness (bottom) as functions of the workload size, for budget-ratio=0.5 and resource factor of 0.7


## Acknowledgement

We would like to sincerely thank Dr. Saeid Abrishami for providing us with the source code of the ICPCP method [15] and discussing about its detail implementation.


## References


[1] J. D. Ullman, Np-complete scheduling problems, Journal of Computer and System sciences 10 (3) (1975) 384–393.

[2] R. L. Graham, Bounds on multiprocessing timing anomalies, SIAM journal on Applied Mathematics 17 (2) (1969) 416–429.

[3] H. Topcuoglu, S. Hariri, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE Transactions on Parallel and Distributed Systems 13 (2002) 260–274. doi:10.1109/71.993206. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumbe`

[4] S. Abrishami, M. Naghibzadeh, D. H. J. Epema, Cost-Driven Scheduling of Grid Workflows Using Partial Critical Paths, in: Proceedings of the 11th IEEE/ACM Int'l Conference on Grid Computing (Grid2010), 2010.

[5] L. F. Bittencourt, E. R. M. Madeira, HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds, Journal of Internet Services and Applications 2 (3) (2011) 207–227.

[6] A. Hirales-Carbajal, A. Tchernykh, R. Yahyapour, J. L. González-García, T. Röblitz, J. M. Ramírez-Alcaraz, Multiple workflow scheduling strategies with user run time estimates on a Grid, Journal of Grid Computing 10 (2012) 325–346. doi:10.1007/s10723-012-9215-6.

[7] Z. Yu, W. Shi, A planner-guided scheduling strategy for multiple workflow applications, in: Proceedings of the International Conference on Parallel Processing Workshops, 2008, pp. 1–8. doi:10.1109/ICPP-W.2008.10.

[8] H. Zhao, R. Sakellariou, Scheduling multiple DAGs onto heterogeneous systems, in: Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International, IEEE, 2006, pp. 14–pp.

[9] L. F. Bittencourt, E. R. M. Madeira, Towards the scheduling of multiple workflows on computational Grids, Journal of Grid Computing 8 (2010) 419–441.

[10] M. Naghibzadeh, Modeling and scheduling hybrid workflows of tasks and task interaction graphs on the cloud, Future Generation Computer Systems 65 (2016) 33–45.

[11] W. Zheng, R. Sakellariou, Budget-deadline constrained workflow planning for admission control in market-oriented environments, in: Economics of Grids, Clouds, Systems, and Services, Springer, 2012, pp. 105–119.

[12] L. F. Bittencourt, E. R. M. Madeira, A performance-oriented adaptive scheduler for dependent tasks on grids, in: Concurrency Computation Practice and Experience, Vol. 20, 2008, pp. 1029–1049. doi:10.1002/cpe.1282.

[13] R. Buyya, S. Pandey, C. Vecchiola, Cloudbus toolkit for market-oriented cloud computing, in: Cloud Computing, Springer, 2009, pp. 24–44.

[14] K. Bessai, S. Youcef, A. Oulamara, C. Godart, S. Nurcan, Resources allocation and scheduling approaches for business process applications in cloud contexts, in: Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on, 2012, pp. 496–503. doi:10.1109/CloudCom.2012.6427530.

[15] S. Abrishami, M. Naghibzadeh, D. H. Epema, Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds, Future Generation Computer Systems 29 (1) (2013) 158–169.

[16] L. F. Bittencourt, E. R. Madeira, F. R. Cicerre, L. E. Buzato, A path clustering heuristic for scheduling task graphs onto a grid, in: 3rd International Workshop on Middleware for Grid Computing (MGC05), 2005.

[17] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, K. Vahi, Characterization of scientific workflows, in: Workflows in Support of Large-Scale Science, 2008. WORKS 2008. Third Workshop on, IEEE, 2008, pp. 1–10.

[18] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, K. Vahi, Characterizing and profiling scientific workflows, Future Generation Computer Systems 29 (3) (2013) 682–692.

[19] J. Livny, Bioinformatic discovery of bacterial regulatory rnas using sipht, Bacterial Regulatory RNA: Methods and Protocols (2012) 3–14.

[20] G. B. Berriman, E. Deelman, J. C. Good, J. C. Jacob, D. S. Katz, C. Kesselman, A. C. Laity, T. A. Prince, G. Singh, M.-H. Su, Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand, Vol. 5493, 2004, pp. 221–232. doi:10.1117/12.550551.
URL http://dx.doi.org/10.1117/12.550551

[21] A. Abramovici, W. Althouse, R. Drever, Y. Gürsel, S. Kawamura, F. Raab, D. Shoemaker, L. Sievers, R. Spero, K. Thorne, et al., Ligo: The laser interferometer gravitational-wave observatory., Science (New York, NY) 256 (5055) (1992) 325–333.

[22] LIGO Project, Ligo - laser interferometer gravitational wave observatory, http://www.ligo.caltech.edu/, accessed: 2016-03-30.

[23] P. Maechling, E. Deelman, L. Zhao, R. Graves, G. Mehta, N. Gupta, J. Mehringer, C. Kesselman, S. Callaghan, D. Okaya, et al., Scec cybershake workflowsautomating probabilistic seismic hazard analysis calculations, in: Workflows for e-Science, Springer, 2007, pp. 143–163.

[24] R. F. da Silva, W. Chen, G. Juve, K. Vahi, E. Deelman, Community resources for enabling research in distributed scientific workflows, in: e-Science (e-Science), 2014 IEEE 10th International Conference on, Vol. 1, IEEE, 2014, pp. 177–184.

[25] A. Ilyushkin, B. Ghit, D. Epema, Scheduling workloads of workflows with unknown task runtimes, in: Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on, IEEE, 2015, pp. 606–616.

[26] R. N. Calheiros, R. Buyya, Cost-effective provisioning and scheduling of deadline-constrained applications in hybrid clouds, in: Web Information Systems Engineering-WISE 2012, Springer, 2012, pp. 171–184.

[27] S. K. Garg, C. S. Yeo, A. Anandasivam, R. Buyya, Environment-conscious scheduling of hpc applications on distributed cloud-oriented data centers, Journal of Parallel and Distributed Computing 71 (6) (2011) 732–749.