A Flexible Scheduler for Workflow Ensembles

Thiago A. L. Genez
Institute of Computing
University of Campinas
Campinas, São Paulo, Brazil
thiagogenez@ic.unicamp.br

Luiz F. Bittencourt
Institute of Computing
University of Campinas
Campinas, São Paulo, Brazil
bit@ic.unicamp.br

Rizos Sakellariou
School of Computer Science
University of Manchester
Manchester, UK
rizos@manchester.ac.uk

Edmundo R. M. Madeira
Institute of Computing
University of Campinas
Campinas, São Paulo, Brazil
edmundo@ic.unicamp.br

ABSTRACT

In this paper, we propose a flexible workflow scheduler that facilitates the replacement of the objective function according to the user's needs. The possibility of replacing the objective function extends the usability of the scheduler for a variety of objectives. The proposed flexible scheduler uses Particle Swarm Optimization (PSO) to assist the production of schedules on cloud resources. We perform the evaluation via simulation using a set of scientific workflows grouped into ensembles. Three conflicting objective functions were evaluated: minimization of the overall makespan, maximization of fairness, and minimization of monetary costs. Simulation results show that the flexibility of the proposed scheduler has been achieved since each function could produce schedules that satisfied its corresponding objective.

CCS Concepts

•**Networks** → **Network services;** *Cloud computing;*

Keywords

Cloud computing, scheduling, workflow ensemble

1. INTRODUCTION

A scheduler is a module responsible to make the distribution of works on resources. As there are numerous ways to conduct such distribution, the scheduler is designed to optimize certain objectives to make an efficient schedule. For example, when scheduling a set of scientific workflows grouped into an *ensemble* [11] onto a number of cloud resources, those workflows compete for the same set of resources. To share the resources equally among the workflows, the use of a fairness-aware scheduler should be more appropriate to schedule the tasks on resources [2, 13]. For other purposes, for instance, when it is necessary to save cost expenses with executions of ensembles on clouds, the use of cost-aware schedulers would be more suitable in this case [3, 8]. On the other hand, when it is necessary to speed up executions, the use of a makespan-aware scheduler is more convenient since it aims to accomplish schedules that minimize the overall makespan (execution time) of the workflow ensemble [12].

Maximization of fairness, minimization of makespan, and minimization of cost are the most common objective functions for a workflow scheduler in the literature. Schedulers are generally designed to optimize one or two objective functions; however, they are *not* designed to allow the replacement of these objective functions, specially the schedulers that cope with workflow ensembles on clouds. Normally, for each objective established for the scheduling procedure, a different scheduler is often used to address such goal. The flexibility of replacing the objective function is not common to occur in current schedulers because these functions are usually tightly attached to the scheduling algorithm.

We consider that a workflow scheduler is *flexible* when it allows the replacement of the objective function according to the user's needs. Similar to a modular design, the objective function can be seen as a "module" that can be switched when necessary, making the scheduler usable in various scenarios. The contribution of this paper is to propose a flexible scheduler to schedule workflow ensembles on clouds that allows the replacement of the objective function according to the user's needs. To the best of our knowledge, this is the first paper in the literature that proposes a flexible scheduler that addresses scientific workflow ensembles on cloud resources. The proposed scheduler employs a method called Particle Swarm Optimization (PSO), a population-based heuristic search algorithm [6]. By having a population (called a swarm) of candidate schedule solutions (called particles), PSO uses a fitness function to select the best solution within the swam. By having different fitness functions, PSO can select the best solution in different ways.

The flexibility of the proposed PSO-based scheduler is accomplished by modelling the workflow scheduling problem as a PSO and developing a fitness function that acts as an objective function of the scheduler. The proposed PSO-based flexible scheduler is evaluated by employing four different fitness functions on the PSO. The fitness function f_1 aims to drive PSO in achieving solution that minimizes the overall makespan of the ensemble, while the functions g_1 and g_2

UCC '16, December 06-09, 2016, Shanghai, China

DOI: http://dx.doi.org/10.1145/2996890.2996910

aim to maximize the fairness among the workflows of the ensemble. Lastly, the function h_1 attempts to drive PSO in producing low-cost schedules. Simulation results show that the desired flexibility for the proposed scheduler has been achieved since each fitness function could produce schedules that satisfied its corresponding objective.

This paper is organized as follows. Section 2 presents related work, while Section 3 reviews the backgrounds and key concepts of the workflow scheduling problem. Section 4 describes the assumptions of the scheduling problem addressed in this paper. The proposed PSO-based flexible workflow scheduler is introduced in Section 5 and evaluated in Section 6. The results are discussed in Section 7, while Section 8 provides final remarks and concludes the paper.

2. RELATED WORK

Zhao and Sakellariou proposed in [13] six initial heuristics to schedule multiple DAGs (workflows) onto heterogeneous resources on grids. These heuristics focus not only on minimizing the overall makespan but on achieving fairness as well. The term fairness is defined on the basis of the slowdown that each DAG would experience (as a result of sharing the resources with other DAGs as opposed to having the resources on its own). Basically, the heuristics depend on the use of any single-DAG scheduling algorithm, since several DAGs are merged into a single-DAG. Bittencourt and Madeira also described [2] four initial strategies for scheduling multiple DAGs on grids in terms of makespan and fairness. The DAGs are not merged in this case, but the strategies also depend on the use of a single-DAG scheduling algorithm to produce the schedules. These papers experimentally demonstrated that it is possible to achieve fairness based on similar distribution of the slowdown amongst the DAGs, without this happening at the expense of the overall makespan. Nevertheless, these schedulers were not flexible to comply with the objective of the minimization of monetary cost, for example.

Malawski et al. presented in [7] three strategies to schedule workflow ensembles under budget and deadline constraints on clouds. The goal of these strategies is to maximize the amount of user-prioritized workflows in an ensemble that can be completed on clouds given budget and deadline constraints. Pietri et al. extended [7] in [9] by including energy constraints along with either budget or deadline constraints. Both works also experimentally demonstrated that it is possible to maximize the number of higher-priority workflows in the ensemble under the given constraints. However, these schedulers may not produce good schedules to minimize the overall makespan or maximize the fairness since they are inflexible to be re-configured to meet such objectives.

Although these works present various strategies to schedule a set of workflows simultaneously, none of them is considered flexible since they do not allow the replacement of the objective function. In contrast to related work, our paper proposes a flexible scheduler in which the objective of the scheduler can be developed according to the user's needs. Our workflow scheduler can be configured to minimize the overall makespan and achieve fairness, as presented in [2,13], as well as produce schedules under budget or deadline constraints, as described in [7,9]. Finally, our flexible scheduler is ready to handle the scheduling of a set of workflows simultaneously on a set of heterogeneous cloud resources.

3. BACKGROUND AND KEY CONCEPTS

3.1 Workflow Representation

A workflow application is typically represented by a directed acyclic graph (DAG) in which nodes represent computational tasks, and edges represent data- or control-flow dependencies among the tasks. Each task depicts how much computing power a task requires to be executed, and each edge depicts how much data must be transferred between two dependent tasks. We consider an acyclic workflow as a representation of a sequence of tasks in a program that solves a scientific problem. A task can only start after all of its parent tasks have finished and all necessary data has been received.

3.2 Workflow Ensemble vs Multiple Workflows

The term *workflow ensemble* usually represents an entire scientific application that comprises a set of inter-related workflows [5,7,11]. Workflows in an ensemble typically have a similar structure, distinguished only by their sizes (number of tasks), input data, and individual computational task sizes. For example, Montage is an application for astronomy research [10] that creates science-grade astronomical image-mosaics using data collected from several telescopes. The Galactic Plane project [10], in turn, uses several Montage workflow applications grouped into an ensemble to combine several mosaics of different regions of the sky to generate a single large image-mosaic of the entire sky. The Galactic Plane ensemble consists of 17 workflows, each of which contains 900 sub-workflows [7]. The term *multiple workflows*, on the other hand, is usually used where there are several different types of non-inter-related workflows grouped into the same set. Although this paper considers only workflows in ensembles, an evaluation using sets of multiple workflows could also be feasible but currently left as future work.

3.3 The Problem of Scheduling Workflows

The scheduler is the element responsible for determining which task of a workflow will run on what computational resources. The distribution of tasks on resources must be performed according to certain objectives, such as the minimization of the makespan and maximization of fairness. Fairness and makespan are two common issues that are usually addressed by a scheduler that copes with workflow ensembles. Currently, financial costs is also another issue that needs to be taken into consideration since scientific applications are frequently executed on cloud resources [11]. With the emergence of large-scale workflow applications, it is quite common to consider a scenario in which more than one workflow need to be scheduled simultaneously on resources [7].

4. ASSUMPTIONS

In this paper, we address the problem of scheduling scientific applications on cloud resources by using a PSO-based flexible scheduler. We assume that these applications are ensembles of inter-related scientific workflows modelled as DAGs, and cloud resources are based on virtual machines (VMs) that are provisioned on-demand and are billed by the hour, with partial hours being rounded up, similar to the resource model of the Amazon's Elastic Compute Cloud (EC2). We consider that tasks may run concurrently on multi-core VMs, and there is no preemption during the execution of

Algorithm 1 PSO procedure overview [6]

```
1: procedure PSO
2:     Create a swarm with a fixed number of particles
3:     for each particle of the swarm do
4:         Initialize the particle with random solutions
5:         Set the particles's pbest as null
6:     end for
7:     Set the swarm's gbest as null
8:     while the number of iterations is not reached do
9:         for each particle of the swarm do
10:            Calculate its fitness value
11:            if the current fitness value is better than particle's pbest then
12:                Set current fitness value as the new pbest
13:            end if
14:        end for
15:        Pick the best pbest among all particle
16:        if pbest is better than previous gbest then
17:            Set the current pbest as the new gbest
18:        end if
19:        for each particle of the swarm do
20:            Update particle's velocity and position
21:        end for
22:        Increment iteration
23:    end while
       return position gbest as the solution of the problem
24: end procedure
```

the tasks. Finally, scheduling occurs statically and the migration of task executions among VMs is not supported.

5. FLEXIBLE WORKFLOW SCHEDULER

5.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population-based heuristic global search algorithm introduced by Kennedy and Eberhart [6], inspired by social behavior of bird flocking or fish schooling. PSO is composed by a population (called a swarm) of candidate solutions (called particles) and it solves the problem by moving these particles around in the search space according to a fitness function. In PSO, the particles "fly" through the problem search space by "following" the best particles that currently represent the best solution. Each particle's movement is influenced by its locally best known position (named as *pbest*), but is also guided toward the best globally known positions (named as *gbest*) of the entire population. PSO does not apply evolution operators (mutation or crossover) between individuals of the population as it occurs in genetic algorithms. Instead, in each iteration, all the particles individually update themselves to improve the solutions that they represent.

The particles are assessed by a fitness function, which calculates a fitness value to evaluate the quality of the solutions given by them. Iteratively, PSO is guided to select the particle that contains the minimum (or the maximum) fitness value of the entire population, and set this particle as the best candidate solution to the problem. This process is repeated until the stop condition is reached, when the PSO returns the best solution achieved so far. Algorithm 1 describes the PSO procedure.

PSO is a popular tool due to its simplicity and its effectiveness in solving a wide range of NP-Complete problems at a low computational cost [6]. However, the PSO technique makes few or no assumptions about the problem being optimized, and thus this technique does not guarantee the bounds for the optimality of the achieved solution.

5.2 Particle as a Candidate Schedule Solution

In PSO, the swarm is composed of a number of particles that encompass the solution search-space. To define a particle, let $\mathcal{E} = \{\mathcal{G}_1, \ldots, \mathcal{G}_n\}$ be a DAG ensemble and $\mathcal{R}$ be a set of resources. In this paper, a particle is represented by a vector of length k, where k is the total number of tasks contained in the ensemble $\mathcal{E}$. A k-dimensional particle is illustrated in Figure 1. The value assigned to each particle's dimension is an integer that represents the index of a computing resource in $\mathcal{R}$, and therefore the entire particle represents a candidate schedule that allocates $\mathcal{E}$ to $\mathcal{R}$.

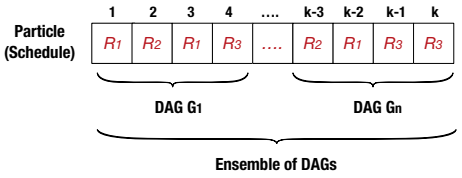


Figure 1: A sample of a particle that represents a schedule for an ensemble of n inter-related DAGs, where $\mathcal{R} = \{R_1, R_2, R_3\}$ is a set of resources.

5.3 Scheduling Policy

In this paper, we assume that the DAGs in an ensemble (represented by a particle) are scheduled independently, one after another, on the basis of "first come, first served". The merging of DAGs into a one single-DAG, as it was proposed in [13], is not considered. Before initiating scheduling, we assume that each DAG in an ensemble is given a numeric priority which indicates how important the DAG is to the user. The highest priority DAG is the first DAG of the ensemble. Regarding the node ordering within a DAG, it follows the topological order given by the depth-first search algorithm. Other types of node ordering could be used instead, such as the upward rank employed by the HEFT algorithm [12, 13].

As the DAGs are scheduled independently, one after another, we consider that the proposed PSO-based scheduler uses a gap search or backfilling technique to schedule the tasks on resources [2]. Basically, this technique searches for free spaces between tasks already scheduled, and thus a task from a DAG that is being scheduled can be executed before tasks already scheduled, and it does not interfere in their starting time. This method takes advantage of the gaps left on resources between scheduled tasks as a result of communication dependencies. By using idle times left on resources to schedule tasks from other workflows, it is possible to achieve the reduction of both the overall makespan as well as the costs involved with leasing VMs, besides trying to maximize the resources utilization.

5.4 Fitness Function as Objective Function

Let $f : \mathbb{N}^k \rightarrow \mathbb{R}$ be the fitness (also called cost) function that must be minimized. This function takes a k-dimensional particle as argument in the form of a vector of integers, and produces a real number as output, which indicates the particle's fitness value. The aim of PSO is to find a particle ρ such that $f(\rho) \leq f(q)$ for all particles q in the swarm. It means that ρ is the particle that represents the global minimum in the swarm. The maximization can be performed by considering the function $h = -f$ instead.

In this paper, the flexibility of the proposed scheduler is

given by the possibility of developing a specific fitness function f to work as the objective function of the proposed scheduler. To demonstrate the proposed flexibility, we employed three conflicting objective functions: minimization of the overall makespan, maximization of fairness and minimization of costs. Based on this, four fitness functions were proposed. Before describing them, consider a given ensemble $\mathcal{E}$ composed of n inter-related DAGs. Also consider a set of resources $\mathcal{R}$ and a k-dimensional particle ρ that represents a candidate schedule that allocates $\mathcal{E}$ to $\mathcal{R}$. The number n can also be defined as $n = |\mathcal{E}|$, where $|\mathcal{E}|$ denotes the cardinality of the ensemble $\mathcal{E}$ in terms of number of DAGs presents in the ensemble. An overview of how these fitness functions are used by PSO is illustrated in Algorithm 2.

Algorithm 2 Fitness Function Overview

1: **procedure** FITNESS FUNCTION (a particle $\rho[\,]$)
2: Calculates the fitness value of the candidate schedule given by the particle ρ
3: **return** the fitness value of ρ to Algorithm 1
4: **end procedure**

5.4.1 Minimization of the Overall Makespan

The most common objective of a workflow scheduler is to minimize the schedule makespan (application execution time) [2]. To comply with this, the fitness function f_1 was developed. This function produces as output numbers associated with the overall makespan of a given particle (schedule) ρ. By using this function, the PSO is setup to minimize the resolution of the problem, since its goal is to find particles that represent schedules with lowest overall makespan. The fitness function f_1 is defined as follows:

$$f_1(\rho) = \mu(\rho) \tag{1}$$

where $\mu(\rho)$ is the overall makespan of the schedule given by the k-dimensional particle ρ, that is, the start time of the first task of the first DAG in the ensemble $\mathcal{E}$ to the finish time of the last task of the last DAG in the ensemble $\mathcal{E}$.

5.4.2 Maximizing Fairness

When scheduling several DAGs simultaneously on the same set of resources, each DAG may experience a *slowdown* in its execution, when compared to the runtime of the same DAG when it has the resources on its own. Due to resource sharing, it may be interesting to distribute the tasks on the resources in a way that the processing capacities can be fairly shared among the DAGs. To prevent an unfair sharing, the scheduler should consider *fairness* during the producing of schedules to share the resources equally among the DAGs. A schedule is fair when each DAG of the ensemble experiences equal (or similar) slowdown in its execution.

Derived from [13], the slowdown value of a DAG $i \in \mathcal{E}$, $\sigma(i)$, in the schedule given by the particle ρ is defined as:

$$\sigma_i = \frac{\mu_i^{own}(\rho)}{\mu_i(\rho)}, \tag{2}$$

where μ_i^{own} is the makespan of a DAG i if it had the available resources on its own, and μ_i is the makespan of a DAG i when sharing resources along with all other DAGs. It is expected that slowdown values will be between 0 and 1, with values closer to 1 indicating a small slowdown.

In an attempt to maximize fairness, we employ two fitness functions in this paper: g_1 and g_2. These functions produce as output numbers associated with slowdowns. The aim is to achieve particles that could represent schedules in which the slowdown difference among all DAGs is small.

Derived from [1], the function g_1 is defined as the ratio between the lowest (minimum) slowdown and the peak (maximum) slowdown experienced by the DAGs in the schedule. In this case, the PSO is configured to maximize the resolution of the problem since its goal is to find particles with higher $g_1(\rho)$ values, because the maximum expected slowdown value for a DAG is 1. The function g_1 is defined as:

$$g_1(\rho) = \frac{\min\limits_{\forall\, i \in \mathcal{E}}(\sigma_i)}{\max\limits_{\forall\, i \in \mathcal{E}}(\sigma_i)} \tag{3}$$

The fitness function g_2, derived from [13], is defined on the basis of the absolute value of the difference between the slowdown of each DAG and the average slowdown of all DAGs. This function is defined as:

$$g_2(\rho) = \frac{1}{|\mathcal{E}|} \sum_{\forall\, i \in \mathcal{E}} \left|\sigma_i - \bar{\sigma}\right|, \tag{4}$$

where $\bar{\sigma}$ is the average slowdown value for all DAGs in $\mathcal{E}$ given by:

$$\bar{\sigma} = \frac{1}{|\mathcal{E}|} \sum_{\forall\, i \in \mathcal{E}} \sigma_i \tag{5}$$

A low $g_2(\rho)$ value indicates schedules where slowdown difference between DAGs is small, and therefore these schedules are reasonably fair to each DAG. Therefore, the use of the fitness function g_2 demands that the PSO be configured to minimize the resolution of the problem since its goal is to find particles that result in smaller $g_2(\rho)$ values.

5.4.3 Minimizing Monetary Costs

In cloud scenarios, scheduling of workflow ensembles has to take into account not only performance-related metrics such as makespan or fairness, but also financial expense, since the resources obtained from commercial clouds have monetary costs associated with them. In an attempt to minimize costs, the fitness function h_1 was developed, and it is defined on the basis of the monetary execution cost of a schedule represented by a particle ρ. This function demands that the PSO minimizes the resolution of the problem, since its goal is to achieve particles that represent low-cost schedules. The function h_1 is defined as follows:

$$h_1(\rho) = \mathcal{C}(\rho), \tag{6}$$

where $\mathcal{C}(\rho)$ is the overall monetary execution cost of the schedule given by ρ, which is defined as follows:

$$\mathcal{C}(\rho) = \sum_{\forall\, r \in \mathcal{R}} \left(c_r \cdot \lceil t_{r,\rho} \rceil\right) \tag{7}$$

where c_r is the cost per time unit for using the resource $r \in \mathcal{R}$, and $t_{r,\rho}$ is the total time units of usage of r in the schedule given by ρ, where partial units are rounded up.

5.5 Summary

This section presented the proposed PSO-based flexible scheduler of workflow ensembles and four different fitness

Table 1: Summary of fitness functions as the objective function of the proposed PSO-based scheduler

Objective Function	Fitness function	Short description of the fitness value produced by the fitness function
Minimizing Overall Makespan	f_1	Overall makespan of the ensemble
Maximizing Fairness	g_1	Ratio between the minimum and the maximum slowdown values
	g_2	Absolute difference between slowdown values and the average slowdown
Minimizing Costs	h_1	Monetary execution cost of the ensemble

Table 2: Cloud Scenario

Type	Cores	Performance per core	Price per unit of time	Quantity
Small	2	2	$ 0.026	4
Large	2	6.5	$ 0.120	4
X-Large	4	13	$ 0.239	4

functions that work as the objective function of the scheduler. The feature of flexibility is accomplished by the possibility of developing a specific fitness function that addresses the user's needs. To highlight this flexibility, we employ four different (conflicting) fitness functions related to the minimization of overall makespan, maximization of fairness, and minimization of costs. A summary of these functions is shown in Table 1.

6. EVALUATION METHODOLOGY

6.1 Workflow Ensembles

In order to evaluate the proposed flexible scheduler on a set of workflows, we created ensembles using workflows available from the workflow generator gallery[1]. The gallery is composed of synthetic workflows modelled using patterns taken from real applications [11], such as: CyberShake, Epigenomics, LIGO, Montage, and SIPHT. Due to space limitation, we present only results for the Montage and Epigenomics applications. As an initial evaluation of the proposed flexible scheduler, we considered only workflows with 50 tasks (DAG nodes). For each workflow application, 20 different workflows were generated differing in the weights of the DAG nodes (computational demands) and DAG edges (communication demands). Using this collection of synthetic workflows, we constructed ensembles for each types of workflow application. The number of workflows in an ensemble depends on the particular application, but, in this first evaluation, we assume that ensemble sizes are in the order of between 1 and 10 workflows. Given an ensemble size, the ensemble is constructed by choosing randomly workflows from this collection of workflows.

6.2 Cloud Scenario

The resource scenarios to execute the workflow are shown in Table 2. They employ resources based on the configurations of the Amazon's Elastic Computing Cloud. The cloud scenario of the evaluation contains three types of virtual machines (S, L, XL), resulting in a total of 12 virtual machines. The availability of bandwidth of 100 Mbps was considered in the communication channels that connect the resources inside the provider. We assume that the resources are exclusively used for the workflow ensemble execution.

6.3 Experimental Simulator

Figure 2 shows an overview of how the evaluations were carried out. The data inputs for the scheduler are DAX files (ensemble) and a VMs file. Each DAX file contains the description of the workflow in XML format, while the VMs file contains information about the resources from Table 2. DAX is a file format used by the Pegasus Workflow Management System[1]. After the schedule is produced by the scheduler, it is used as an input to the simulator, which simulates the schedule and calculates the metric values such as: the overall makespan, slowdown values, monetary execution costs and speedup. Section 5.3 has described how the workflows in the ensemble are ordered and how the tasks of a workflow are ranked at scheduling time. However, at simulation time, all tasks of all workflows are re-ranked according to the execution finish time estimated by the produced schedule.

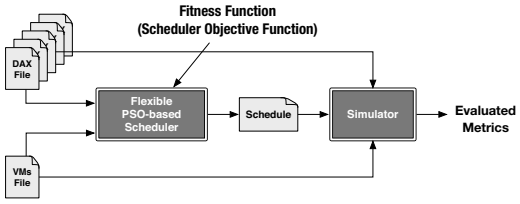


Figure 2: Overview of the experimental evaluation.

6.4 Experimental PSO Parameters

We used the JSwarm-PSO package[2] to conduct the experimental evaluation of the proposed PSO-based flexible scheduler. The number of particles was set to 500 and the number of iterations was set to 50. The particle's inertia ω was set to 1.5, and it decreases 1% at each iteration. The particle's acceleration values a_1 and a_2 were set to 0.95, and both decrease 1% at each iteration as well.

6.5 Evaluated Metrics

The evaluated metrics are the overall makespan of the workflow ensemble, the monetary cost of the schedule, and the runtime of the scheduler to find a solution. To measure fairness, we employ the Jain's fairness index [4], an index that was developed for resource sharing or allocation problem. Consider a set of values $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, the Jain's fairness index for the set $\mathcal{X}$ is defined as follows:

$$\mathcal{J}(x_1, x_2 \ldots, x_n) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n\left(\sum_{i=1}^{n} x_i^2\right)} \quad (8)$$

The Jain's fairness index ranges between $1/n$ (worst case, unfair) and 1 (best case, fair). An index value closer to 1 indicates that the slowdown difference between the DAGs is small, and thus the schedule is reasonably fair to each DAG.

The main advantage of sharing multiple resources is to attempt to speed up the execution of the application by exploring the gaps (idle times) left on resources between

[1]https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator

[2]http://jswarm-pso.sourceforge.net/

dependent-tasks already scheduled. To compare the performance of the resource sharing among the workflows, the speedup was employed. Given a schedule for a workflow ensemble, we define speedup as the ratio between the overall makespan taken to process the workflows sequentially without sharing the resources, and the overall makespan taken to process the same set of workflows but with the sharing of resources among the workflows. The speedup of a schedule (particle) ρ of an ensemble $\mathcal{E}$ is given by:

$$Speedup\,(\rho) = \frac{1}{\mu(\rho)} \sum_{\forall i \in \mathcal{E}} \mu_i^{own}(\rho), \tag{9}$$

Speedup values grater than 1 are expected. Otherwise, resource sharing among the workflows is ineffective to speedup the execution of the ensemble.

7. RESULTS

This section shows the experimental results obtained over 100 simulations with a confidence interval of 95%.

7.1 Minimization of the Overall Makespan

Figures 3a and 3b show the average overall makespan of the evaluation results for the Montage and Epigenomics applications, respectively. As it was expected, the fitness function f_1 results in lower average overall makespans for all applications and ensemble sizes. For example, by using f_1 to drive PSO in producing schedules for 10-workflows ensembles for the Montage Application, the overall makespan average values were approximately 24%, 31%, and 7% lower than the average values of the solutions achieved by the use of the functions g_1, g_2, and h_1, respectively. These reductions were approximately 43%, 55%, and 18% for 10-workflows ensembles of the Epigenomics application, respectively.

By employing a dedicated fitness function to comply with the objective of minimize the overall makespan, the proposed PSO-based flexible scheduler was able to achieve schedules with reduced overall makespan in all experimental cases.

7.2 Maximization of Fairness

The goal of minimizing the overall makespan given by f_1 may drive PSO in producing unfair schedules. This section presents the results given by the functions g_1 and g_2, where both attempt to maximize the fairness of the schedules. Figures 3c and 3d present the average Jain's fairness index of the evaluation experiments for the Montage and Epigenomics workflows, respectively. As it was also expected, the functions g_1 and g_2 were able to drive PSO in achieving fairer schedules for all workflows and ensemble sizes, when compared to the schedules produced through the use of the functions f_1 and h_1. In fact, for all cases, the function g_1 could manage fairer solutions than g_2. For example, by using g_1 to conduct the production of schedules for 10-workflows ensembles of the Montage Application, the averages Jain's index were approximately 1%, 10%, and 8% higher than the averages index given by solutions produced by the functions g_2, f_1, and h_1, respectively. These increases were approximately 4%, 7%, and 5% for 10-workflows Epigenomics ensembles.

A Jain's index value closer to 1 indicates that the slowdown difference among the DAGs is small, and therefore the schedule is reasonably fair to each DAG. For most of the cases, the fitness function g_1 and g_2 achieved averages higher than 0.9. However, the approach of trying to reduce the difference between the minimum and maximum slowdown values (defined by g_1) seems to be more effective in maximize fairness than the approach of trying to reduce the average absolute difference between slowdown values and the average slowdown (defined by g_2).

Table 3: Average speedup for the Montage and Epigenomics ensembles.

		Ensemble Size									
	Functions	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Montage	f_1	1.0	1.9	2.3	2.5	3.0	3.7	4.2	4.8	5.5	6.0
	g_1	1.0	1.7	2.3	2.8	3.2	3.7	4.2	5.9	4.8	5.2
	g_2	1.0	1.7	2.2	2.7	3.2	3.5	3.9	2.4	4.5	4.6
	h_1	1.0	1.6	2.0	2.3	2.8	3.2	3.6	4.1	4.4	4.7
Epigeno.	f_1	1.0	1.6	2.2	2.7	3.0	3.2	3.5	3.6	3.9	4.0
	g_1	1.0	1.5	2.0	2.3	2.5	2.5	2.7	2.7	2.9	2.9
	g_2	1.0	1.5	2.0	2.1	2.2	2.3	2.3	2.3	2.5	2.5
	h_1	1.0	1.4	1.8	2.2	2.5	2.8	3.0	3.1	3.3	3.6

The objective of maximizing fairness involves a drawback with respect to financial costs. Due to conflicting objectives, a fair schedule implies a higher overall makespan, as shown in Figures 3a and 3b, which causes an increase in costs, since cloud resources have monetary costs associated with them. In the next section, we discuss the results given by the function h_1, which attempts to drive PSO to produce low-cost schedules.

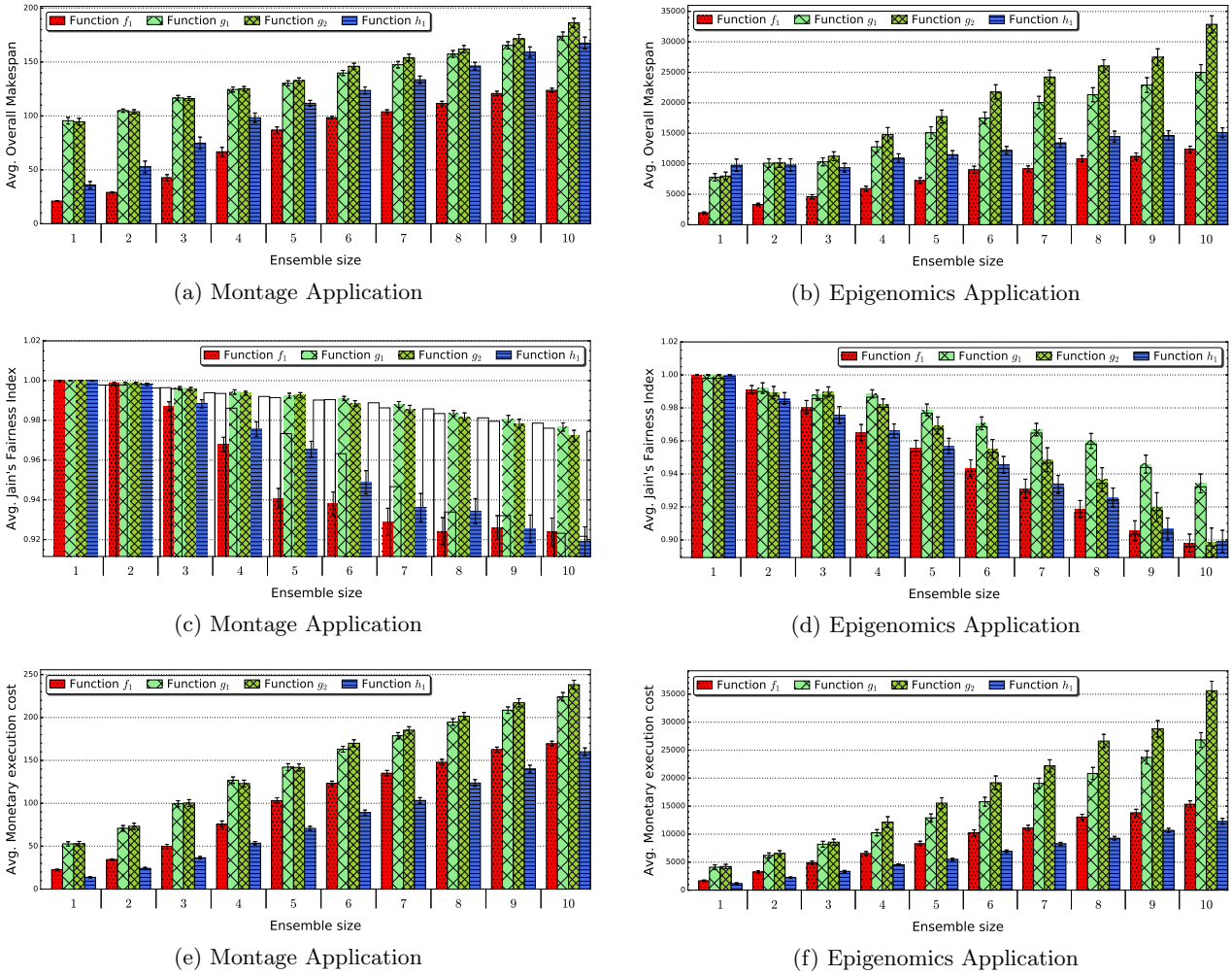
7.3 Minimization of Monetary Costs

In addition to minimizing the makespan and maximizing fairness, the proposed flexible scheduler can also be applied to minimizing monetary costs of the workflow ensembles scheduling on clouds. Figures 3e and 3f show the average monetary execution costs of the evaluation results for the Montage and Epigenomics applications, respectively, on resources described in Table 2. In this evaluation, it was expected that the fitness function h_1 would conduct the PSO algorithm in producing cheaper schedules, and indeed for all applications and ensemble sizes, this expectation has been achieved. For example, by using h_1 on the proposed scheduler, it was able to achieve schedules approximately 17%, 37%, and 39% cheaper on average than the schedules achieved by the use of the functions f_1, g_1, and g_2, respectively, for 8-workflows ensembles of the Montage Application. These cost reductions were approximately 29%, 55%, and 65% for the Epigenomics application, respectively.

The fitness function h_1 has complied with its duty to conduct the PSO to achieve cost-efficient schedules for scientific applications expressed as workflow ensembles. The use of the function h_1 is just an example of how flexible it is to drive the scheduler from a scheduling fairness maximization problem to a scheduling cost minimization problem.

7.4 Speedup

The speedup values of the experimental evaluation of Figure 3 is shown in Table 3. For all applications and ensemble sizes, the proposed scheduler was able to produce schedules in which speedup values were greater than 1. In other words, the scheduler was capable to take advantage of the free gaps left on the resources to schedule tasks from other workflows to reduce the application execution time. As it was expected, the function f_1 was able to produce higher speedup values than those produced by the functions g_1, g_2, and h_1, since it attempts to minimize the overall makespan. For instance, for 10-workflows ensembles of the Montage Ap-



(a) Montage Application

(b) Epigenomics Application

(c) Montage Application

(d) Epigenomics Application

(e) Montage Application

(f) Epigenomics Application

Figure 3: Average overall makespan, Jain's fairness index, and monetary execution costs for the Montage and Epigenomics workflow ensembles. The x-axis represents the number of workflows within the ensemble, which ranges from 1 to 10.

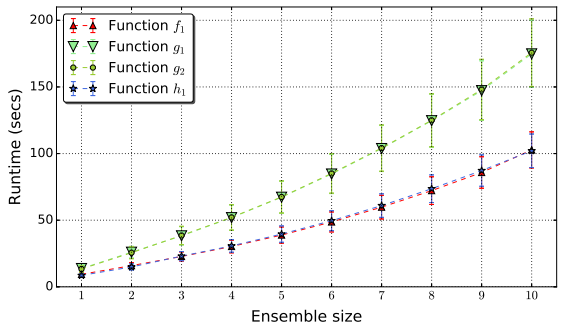


Figure 4: Average runtime of the proposed PSO-based scheduler over 100 runs

plication, the f_1 achieved an average speedup approximately of 6.0 against 5.2 , 4.6 and 4.7 produced by the functions g_1, g_2, and h_1, respectively.

7.5 Runtime of the Proposed Flexible Scheduler

The average runtime to execute our proposed PSO-based flexible scheduler, averaged over 100 runs using workflow ensembles from Montage and Epigenomics applications with four different fitness functions (f_1, g_1, g_2, and h_1), is shown in Figure 4. The running time appears to grow linearly with the number of workflows (DAGs) in the ensemble considered. The two fitness functions (g_1, g_2) are slightly slower, since both need to make an additional makespan computation for each a DAG of the ensemble. Unlike f_1 and h_1, the functions g_1, g_2 need to compute the makespan when a DAG is and is not sharing the resources among other DAGs.

7.6 Discussion

As a general contribution of the presented results, we are able to conclude from these simulations that the feature of flexibility in replacing the scheduler objective function extends the usefulness of the workflow scheduler to be applied in various scenarios. The fitness function f_1 was able to drive PSO in achieving schedules with smallest overall makespan,

while fairest schedules were able to be found by the use of g_1 and g_2. The use of the function h_1 was more suitable to achieve cheapest schedules. These functions are only sample functions that were used to highlight the flexibility given the proposed scheduler. Other more sophisticated functions could be used instead to achieve better results.

The proposed flexible scheduler provides the user freedom in designing different objective functions to manage the production of schedules in different ways and according to its needs. For example, besides the scheduling scenarios shown in this paper, the user may need to develop a fitness function that leads the scheduler in achieving solutions that minimize the ratio between the cost and overall makespan, or even minimize the costs and comply with deadline constraints. The proposed scheduler handles a set of inter-related workflows grouped into ensemble simultaneously on a set of cloud resources. The problem of scheduling a set of workflows simultaneously has been neglected by most studies in scheduling, specially those studies on clouds.

8. CONCLUSION AND FUTURE WORK

In this paper we proposed a flexible scheduler based on Particle Swarm Optimization (PSO) to schedule workflow ensembles on cloud resources. The proposed scheduler allows the replacement of its objective function according to the user's needs. The proposed flexibility is given by the possibility of modeling the scheduler as a PSO and its objective function as a PSO's fitness function. Three types of objective function were evaluated: minimization of the overall makespan, maximization of fairness, and minimization of costs. Based on these objectives, four fitness functions were developed and employed on the PSO. The function f_1 aims to lead the PSO in achieving solution that minimizes the overall makespan of the ensemble, while the functions g_1 and g_2 aim to maximize the fairness among the workflows. Lastly, the function h_1 attempts to lead the PSO in producing low-cost schedules. The results of our experimental evaluation indicate that the desired flexibility for the proposed scheduler has been achieved since each fitness function could produce schedules that satisfied its objective.

This paper suggests several studies for future work. Our current experimental evaluation employed only ensembles composed of workflows in which all of them have the same number of tasks, which in this case was 50 tasks. In the future, we plan to extend this experimental evaluation by using different ensemble types, such as ensembles that contain a small number of larger workflows (1000 tasks, for instance) and a large number of smaller workflows (50 tasks for example). In other words, ensembles to have a slightly larger number of large workflows, which reflects behaviour commonly observed in many computational workloads. We also plan to evaluate a set of non inter-related workflows, i.e., multiple workflows. Furthermore, in a dynamic approach when it is necessary to switch the scheduling context on-the-fly, for example, the replacement of the objective function could occur during the scheduling of the ensemble. A study involving a dynamic approach is also left as future work.

9. ACKNOWLEDGMENT

The first, second, and fourth authors would like to thank the State of São Paulo Research Foundation (FAPESP – 2012/02778-6) for the financial support.

10. REFERENCES

[1] A. Ayodele and T. Boult. Towards application-centric fairness in multi-tenant clouds with adaptive CPU sharing model. In *9th IEEE International Conference on Cloud Computing (CLOUD)*, 2016.

[2] L. F. Bittencourt and E. R. M. Madeira. Towards the scheduling of multiple workflows on computational grids. *Journal of Grid Computing*, 8(3):419–441, 2010.

[3] T. A. L. Genez, L. F. Bittencourt, and E. R. M. Madeira. Workflow Scheduling for SaaS/PaaS Cloud Providers Considering Two SLA Levels. In *Network Operations and Management Symposyum*, pages 906 –912, 2012.

[4] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical report, Washington University in Saint Louis, 1984.

[5] Q. Jiang, Y. C. Lee, and A. Y. Zomaya. Executing large scale scientific workflow ensembles in public clouds. In *44th International Conference on Parallel Processing (ICPP)*, pages 520–529, Sept 2015.

[6] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE Int'l Conf. on Neural Networks*, volume 4, pages 1942–1948 vol.4, Nov 1995.

[7] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski. Algorithms for cost- and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds. *Future Generator Computer Systems*, 48:1–18, 2015.

[8] S. Pandey, L. Wu, S. Guru, and R. Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *24th IEEE Int'l Conf. on Advanced Information Networking and Applications*, pages 400–407, April 2010.

[9] I. Pietri, M. Malawski, G. Juve, E. Deelman, J. Nabrzyski, and R. Sakellariou. Energy-constrained provisioning for scientific workflow ensembles. In *3rd International Conference on Cloud and Green Computing (CGC)*, pages 34–41, Sept 2013.

[10] M. Rynge, G. Juve, J. Kinney, J. Good, G. B. Berriman, A. Merrihew, and E. Deelman. Producing an infrared multiwavelength galactic plane atlas using montage, pegasus and amazon web services. In *23rd Annual Astronomical Data Analysis Software and Systems Conference*, 2013.

[11] I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields. *Workflows for e-Science: Scientific Workflows for Grids*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[12] 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*, 13(3):260–274, Mar 2002.

[13] H. Zhao and R. Sakellariou. Scheduling multiple DAGs onto heterogeneous systems. In *15th Heterogeneous Computing Workshop (HCW) in conjunction with 20th International conference on Parallel and distributed processing (IPDPS)*, Washington, DC, USA, 2006.