Computer Networks 181 (2020) 107438

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Hybrid scheduling for scientific workflows on hybrid clouds

Amirmohammad Pasdar [a], Young Choon Lee [a], Khaled Almi'ani [b,c,*]

[a] The Department of Computing, Macquarie University, Sydney NSW 2109, Australia
[b] Higher colleges of Technology, Fujairah, United Arab Emirates
[c] Al-Hussein Bin Talal University, Ma'an, Jordan

ARTICLE INFO

Keywords:
Data placement
Scientific workflows
Task scheduling
Hybrid cloud

ABSTRACT

Scientific workflows consist of many interdependent tasks dictated by their data dependencies. As these workflows are becoming resource-intensive in both data and computing, private clouds struggle to cope with their resource requirements. Private cloud's limitations are claimed to be addressed by public clouds, however, the complete offloading of workflow execution to public clouds causes excessive data transfer. In this paper, we address the problem of scheduling scientific workflows on hybrid clouds aiming at cost reduction while improving execution time, with the consideration of public cloud costs, by minimizing data movements between private cloud and public cloud. To this end, we develop **H**ybrid **S**cheduling for **H**ybrid **C**louds (**HSHC**) as a novel data-locality aware scheduling algorithm. HSHC adopts a hybrid approach consisting of static phase and dynamic phase. The former solves the problem of workflow scheduling, using an extended genetic algorithm, with static information of workflows and resources. The latter dynamically adapts the static schedule in response to changing execution conditions, such as locality of intermediate output data and performance degradation of tasks and resources. We evaluate HSHC with both random workflows and real-world scientific applications in execution time and cost. Experimental results compared with seven state-of-the-art algorithms demonstrate HSHC significantly reduces the cost by up to 40% and improves the execution time by up to 25%.

1. Introduction

Cloud computing has been brought forth with advances in x86 virtual machine (VM) techniques from, such as VMware [1] and Xen [2]. The main difference between traditional computing systems and clouds is the elastic resource provisioning, with pay-as-you-go pricing, of clouds from the support of such VM techniques.

A cloud is classified as a private cloud or a public cloud depending on the access mode. A private cloud is an infrastructure for performing workloads within a single administrative domain. It gives full system control, privacy and security, and data governance while the resource capacity is limited or less flexible/elastic [3]. In the meantime, a public cloud, such as Amazon Web Services (AWS) [4] provides a wide range of cloud services with claims of virtually unlimited resources that users pay only for what is used.

Over the past few decades, scientific applications, for example in bioinformatics and astronomy, have become increasingly large scale which necessitates the need for extensive computation and storage resources. Many of these applications deal with a large number of interdependent tasks, in the form of a workflow (i.e., scientific workflows) and they are resource-intensive in terms of both computation and storage.

While scientists can offload scientific workflows to public clouds to take advantage of cost-efficiency and improved performance, the data-intensiveness of recent large-scale scientific workflows significantly hinders such complete offloading. Besides, many organizations including research labs often operate their own private clouds due to privacy, security and data governance, in particular, in medical sciences and health care. For example, a cloud platform named Bionimbus [5] deals with large genomics and phenotypic dataset under security and compliance required by controlled-access data, such as the database of Genotypes and Phenotypes (dbGaP). Hence, a more practical alternative is the hybrid use of private cloud and public cloud(s).

In this paper, we address the problem of scheduling scientific workflows on hybrid clouds explicitly taking into account data locality. Scientific workflow scheduling has been extensively studied in the literature including works with hybrid clouds, e.g., [6–14]. However, the majority, if not all, consider either workflow execution is static or data movements are insignificant [12–14]. Although public cloud usage is promising, it incurs costs as well as data movements between clouds that significantly affects the execution time, e.g., when a data-intensive

* Corresponding author at: Higher colleges of Technology, Fujairah, United Arab Emirates.
E-mail addresses: amirmohammad.pasdar@hdr.mq.edu.au (A. Pasdar), young.lee@mq.edu.au (Y.C. Lee), kalmiani@hct.ac.ae (K. Almi'ani).

https://doi.org/10.1016/j.comnet.2020.107438
Received 18 February 2020; Received in revised form 2 July 2020; Accepted 22 July 2020
Available online 8 August 2020

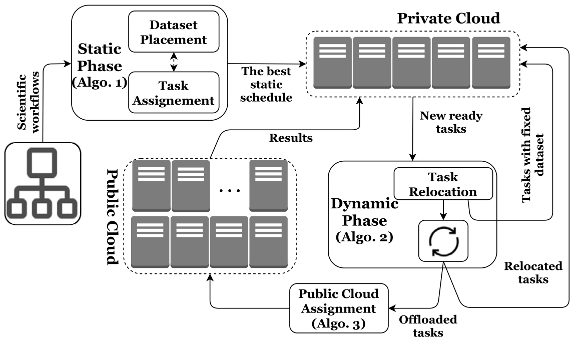


Fig. 1. The overall structure of HSHC.

application is deployed on the private cloud. Performance (i.e., execution time) and cost are conflicting objectives which are not *trivial* to optimize both at the same time, thus, there should have to be a trade-off between them. From the cost perspective, this trade-off could be effectively addressed by acknowledging different billing cycle policies of public cloud providers (e.g., per hour vs per second). However, the hybrid cloud has been agnostic to cloud providers billing cycles and studies focused on a specific (or non-specific) pricing model [15–19].

This paper significantly extends our previous work [20] in which main contributions are stated as follows. (1) Public cloud billing policies are explicitly considered for cost-efficient execution of workflows in the hybrid cloud. (2) Workflow characteristics and resource requirements are taken into account for scheduling decisions aiming at reducing delays within the workflow for performance improvement. (3) The work is compared with state-of-the-art algorithms considering the size and heterogeneity of the private cloud as well as different type of workflows.

To this end, we present a novel data-locality aware scheduling algorithm, called **H**ybrid **S**cheduling for **H**ybrid **C**louds (**HSHC)**. The overall structure of HSHC is shown in Fig. 1. HSHC as an optimization algorithm reconciles the conflict between cost and performance with its two-phase approach based on a genetic algorithm (GA) and a dynamic polynomial algorithm. They are defined as *static* phase and *dynamic* phase. In the static phase, offline scheduling decisions are made through the genetic algorithm, with static information of workflow and resources, before the actual execution. Heuristics including meta-heuristics like GA in this study are considered as rather simple methods to tackle many complex scheduling problems such as task scheduling in this paper, as they do not guarantee finding optimal solutions; however, they can be “unreasonably effective" as evident in many examples given by Turing-award winner Richard Karp [21].

In the dynamic phase, the static schedule is adjusted based on the changing conditions of both tasks and resources. These changing conditions include communication delays caused by changes in intermediate data generation, volume, and performance variations of resources due to resource contention. In case of cost-effective offloading any task to the public clouds, a cost comparison takes place for selecting the proper cloud environment based on their billing policies with respect to the task resource requirements and constraints.

We evaluate HSHC with extensive experiments using real-world scientific workflows such as Montage [22], LIGO [23], and CyberShake [24] as well as a random workflow generator [25]. The performance of HSHC is compared with well-recognized Heterogeneous Earliest-Finish-Time (HEFT) and Critical-Path-on-a-Processor (CPOP) algorithms [25], AsQ [26], meta-heuristic-based algorithms such as HSGA [27] and Particle Swarm Optimization (PSO) [28], First Come First Serve (FCFS), and a best-fit (BF) algorithm. Our results confirm the efficacy of HSHC in terms of execution time and cost that roots in optimizing data-locality with judicial scheduling decisions across the hybrid cloud. HSHC significantly reduced the cost of real-world and random workflows up to 40% and improved the execution time up to 25%.

The remainder of the paper is organized as follows. In Section 2, we review related work on workflow scheduling followed by Section 3 in which we give a description of scientific workflows and hybrid clouds with their characteristics, and the problem is formulated. We present HSHC in Section 4. Evaluation results are presented in Section 5. We conclude this paper in Section 6.

2. Related work

Workflow scheduling has been extensively studied [13,14,28–32]. Due to their precedence constraint dictated by data dependency, the quality of schedule (e.g., makespan) is mainly dependent on reducing communication costs between tasks [29,30,33–38]. This communication cost stems from transferring data among different VMs, and it could happen in any cloud environments. This also becomes far more significant when workflows are scheduled in a hybrid cloud [3,6–11, 19,39–43].

It has been a challenge to acquire the proper amount of resources for workflow execution. Such acquisitions significantly affect the execution time and the usage cost, particularly, in public clouds. Studies [13, 14,25,27,31,32] aimed at improving resource usage and better resource provisioning through an extended Partial Critical Paths (PCP), priority-based scheduling, or Cluster Combining Algorithm (CCA), task prioritization based on task depth in the workflow structure with the help of genetic algorithm, respectively. Also, research works such as [29,30,33,34,44] addressed data-intensiveness of scientific workflows for reducing data movements through machine learning strategies such as k-means or using a graph-cut strategy and as a result improving execution performance and cost significantly in a public cloud.

Workflow scheduling has recently leveraged hybrid clouds as well to take advantage of both internal and external resources. The scheduling approach may be defined as a cost model to address minimum execution time as well as public resources rental cost by presenting a list of resources for execution purposes to the user [45]. The division for the cost is based on considering the budget as a part of the user application requirement. However, exploiting a data management scheme has been a neglected matter as it is assumed datesets are accessible at any time [7,43].

Yuan et al. [19] proposed a hybrid-based cost minimization approach for delay-bounded tasks in a hybrid cloud that uses mixed integer linear program to be solved by simulated annealing and particle swarm optimization algorithms considering energy cost. Yuan et al. [39] also followed similar approach for profit maximization considering public clouds characteristics in terms of resource prices and energy specifications (e.g., electricity rate). Clemente-Castello′ et al. [40] took into account the public resources costs for MapReduce jobs into a performance model to predict the execution time for application of big data. Mechtri et al. [41] designed a heuristic algorithm leveraging topology patterns and bipartite matching in hybrid clouds to reduce mapping delays (i.e., between nodes and data centers) for large scale networks.

Malawski et al. [43] presented a resource calculator for scientific applications on multiple cloud environments through a mathematical model and integer programming. In [7] the cost was modeled and solved through mixed-integer nonlinear programming for workflow scheduling by considering data accessibility time and queue time services on VMs where resources might be limited or unlimited. Lin et al. [8] proposed an online strategy for scheduling continuous workflow submission on hybrid clouds. Their approach aimed to execute the workflow within the given deadline while keeping the public cloud utilization at a lower cost. The approach leveraged a hierarchical iterative application partition (HIA) to cluster the workflow into a set of dependent tasks.

Table 1
Symbol description.

Symbol	Description
CP	Cloud provider
BC_{CP}	Billing cycle of a cloud provider
BT_{CP}	Cloud provider booting time
R	Set of cloud resources
r	Virtual Machine
VS	A set of tasks in a VM
v	An arbitrary task
CV	A set of tasks running concurrently in a VM
DS	A set of datasets in a VM
CT_{ij}	Communication time between tasks v_i and v_j
$\Pi_C/\Pi_B/\Pi_S$	Computation/Bandwidth/Storage cost
D_i	Task i deadline
ds	Input data
IDS_i	Task i produced intermediate dataset
DP_{ij}	Data dependency between tasks i and j
DI_{r_j}	Internal data dependency between datasets in VM j
DO_{r_j}	External data dependency of datasets in VM j and other VMs
PT_{ij}	Processing time of task i on VM j
FT_{ij}	Finish time of task i on VM j
DA_{ij}	Dataset availability of task i on VM j
Λ_{ij}	Task i execution delay on VM j
Λ_T	Total execution delay
P_r	Overall computational ratio of a VM
U_j	Ratio of VM j usage
F/NF	Tasks with/without flexible dataset
l_{off}	List of offloading tasks
DT_{ij}	The gap between task i deadline and its finishing time on VM j

Shaghahyegh et al. [3] proposed a Software as a Service application (SaaS) for scheduling workflows to address privacy concerns while taking into account optimizing execution time and task deadlines. Rahman et al. [10] developed an Adaptive Hybrid Heuristic (AHH) scheduling strategy to map tasks to the available VMs through a genetic algorithm to stay with the deadline and budget while user constraints are also satisfied. Luiz and Edmundo [11] with the help of Heuristic Path Clustering (HPC) managed resource assignment to stay with the expected deadline. It aimed to choose proper resources from the public cloud to make them available for the private cloud for providing sufficient processing power, i.e., CPU cores, to stay with the expected execution time. Zhanghui et al. [38] addressed reducing the data movements and transmission between data centers through a combination of particle swarm optimization and genetic algorithm. However, they did not consider workflow scheduling and how it could be cost-efficient in conjunction with their approach.

Workflow scheduling within a hybrid cloud is also implicitly related to the workflow type, and a majority of studies have addressed Bag-of-Tasks applications due to its simple structure and parallelism [6,29,42,46–55]. This parallelism can also happen within a workflow structure when tasks waiting for their intermediate datasets become ready during the workflow execution. Resource allocation for tasks can be done through usage of optimization algorithms such as meta-heuristic algorithms [28] or a multi-dimension multi-choice knapsack problem [26] to find a near-optimal solution considering cost and deadline constraints.

This work differs from the mentioned works as (1) it explicitly considers the scientific workflow structure in a hybrid cloud environment, (2) takes into account the data locality while scheduling the workflow, and (3) explicitly considers public cloud billing cycles for cost-efficient execution.

3. Models and problem formulation

In this section, we describe application and system models. We then formulate the problem. The description of symbols used in this paper is shown in Table 1.

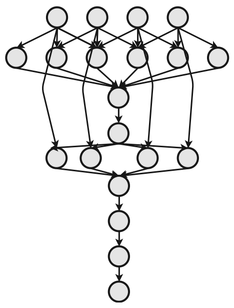


Fig. 2. A sketched sample workflow structure of Montage astronomical image mosaic engine.

3.1. Scientific workflows

A workflow structure is presented as a Directed Acyclic Graph (DAG), $G = (V, E)$ comprising a set V of tasks $V = \{v_1, \dots, v_n\}$, and a set E of edges, each of which connects two tasks representing their precedence constraint or data dependency. A task v_i is the parent task of v_j if v_j relies on the output of v_i. For a given task, the last parent to finish its execution is termed the Most Influential Parent (MIP).

Tasks start their execution on either an initial data submitted by the user or data sent by an intermediate task. The time required to transfer data from task v_i to task v_j is referred to as the Communication Time (CT_{ij}). If both tasks v_i and v_j are on the same VM, the communication time is zero. Fig. 2 shows a sketched sample workflow structure of Montage astronomical image mosaic engine, which stitches multiple sky images. A 6.0-degree Montage workflow creates a 6-by-6 degree square mosaic centered at a particular region of the sky (e.g., M16). The number of jobs and input data files increases with the number of degrees of the mosaic. A 6.0-degree Montage workflow contains 8586 jobs, 1444 input files with a total size of 4.0 GB, and 22,850 intermediate files with a total size of 35 GB.

3.2. Hybrid clouds

A hybrid cloud in this study consists of a private cloud and one or P public clouds referred to as public cloud platforms (CP). Thus, the hybrid cloud can be stated as $H = (\bigcup_{r=1}^{m} R^{prv}) \cup (\bigcup_{r=1}^{P} R^{CP})$. Compute resources in both cloud environments are virtualized; hence, VMs or instances. These VMs are homogeneous or heterogeneous with respect to their resources, such as storage, computation, and bandwidth. In particular, a cloud consists of a set R of VMs; $R = \{r_1, \dots, r_m\}$. They are characterized by their computing capacities $C = \{c_1, \dots, c_m\}$, bandwidth $B = \{b_1, \dots, b_m\}$ as set of links between resources and storage capacities $S = \{s_1, \dots, s_m\}$.

3.3. Cost

Within a hybrid cloud, tasks may be offloaded to the public cloud (e.g., when the private cloud experiences workload surges), and their results may be sent back to the private cloud. This process necessitates communications between the private cloud and public clouds and launching VMs for hosting tasks and datasets. Consequently, it incurs costs as in public clouds charges are applied based on the amount of allocated storage (Π_S), consumed bandwidth (Π_B), and the number of accountable time units (i.e., a billing cycle (BC)) per virtual machine/instance (Π_C). A public cloud billing cycle (BC_{CP}) may have remaining time whose VM r_i could be used free of charge. Hence, there may be a completely new billing cycle (BC^n_{CP}) that can incur charges or already active one (BC^a_{CP}) which could be used free of charge before the end of the billing cycle. Also, each CP has an approximate virtual machine boot-up time (BT_{CP}) to make the virtual machine ready to use.

For a VM r_i in the public cloud, the overall execution cost (Π_{r_i}) depends on the cost of used storage, consumed bandwidth, and computation. More formally,

$$\Pi_{r_i} = \Pi_{b_i} + \Pi_{s_i} + \Pi_{c_i} \tag{1}$$

Note that for the cost of schedule in this study, we only consider the cost of public clouds.

3.4. Data dependency

A workflow consists of n interdependent tasks that uses input datasets and produces intermediate datasets. Each task v_i has a user-specified deadline D_i, required input data DS_i and intermediate datasets IDS_i. Each DS_i consists of some ds that may be flexible or fixed dataset. In the case of fixed datasets, they have to be placed at the designated VM. If the number of required ds for v_i is u, DS_i is defined as follows:

$$DS_i = \{ds_1, \dots, ds_u\} \tag{2}$$

A task within a workflow may need certain datasets for execution and some of these requested datasets can also be accessed by the other tasks. Therefore, it is advantageous to place these datasets *ideally* in one VM if possible. This leads to introducing a degree of correlation between two datasets referred to as dependency (DP_{ij}) which is the number of required datasets shared between two tasks. Formally, suppose tasks v_i and v_j require datasets DS_i and DS_j, respectively. DP_{ij} is defined as follows:

$$DP_{ij} = |DS_i \cap DS_j| \tag{3}$$

3.5. Task execution

The finish time (FT_{ij}) of a task v_i on a VM r_j is defined as the summation of Processing Time (PT_{ij}) and communication time (CT_{ij}). The former shows how long r_j takes to process v_i. The latter represents the amount of time that the task v_i has to wait to get *all* the required datasets for the target VM (r_j). This time implicitly includes the communication time between task v_i and task $v_{j'}$ which produces intermediate datasets required by v_i ($\sum_{j'=0}^{|IDS_i|} CT_{ij'}$).

$$FT_{ij} = PT_{ij} + CT_{ij} \tag{4}$$

3.6. Problem definition

HSHC for scheduling scientific workflows focuses on execution time reduction considering data locality determined by Eq. (3). The data locality influences a task's execution time as it also relies on the communication time in Eq. (4). Moreover, HSHC aims to minimize public cloud usage cost Eq. (1) through comparison and selection of the most cost-efficient billing cycles of cloud providers. Note that we have modeled both scientific workflows and hybrid clouds (cost calculation in public clouds in particular) based on real-world applications and clouds, such as Amazon EC2 and Google Cloud Platform.

Thus, for a workflow with n interdependent tasks which require datasets, the execution of a task (v_i) on VM (r_j) is influenced by the amount of required data-set DS_i which needs to be transferred to that VM. If the VM belongs to public clouds, the task execution incurs charges which has to be minimized. In other words, for task (v_i), VM (r_j) of P public clouds is chosen by the following Eq. (5):

$$r_j = min_{r \in \bigcup_{r=1}^{P} R^{CP}} \Pi_r \tag{5}$$

Hence, the problem in our study is how to schedule tasks and their required datasets of a given workflow to the resources within a hybrid cloud environment such that the schedule is optimized in terms of execution time and cost.

Algorithm 1: Static scheduling (Genetic Algorithm)

```
   Data: private VM list, task list, dataset list, max
         #generation and mutation probability
   Result: The best static schedule
1  prepare the initGeneration
2  evaluate initGeneration via Eq. (10)
3  while #generations not exceeded do
4      apply tournament-selection
5      consider the chosen candidates for making new solution
       with
6      apply k-way-crossover on the chosen candidates
7      apply concurrent-rotation mutation based on the
       mutation probability
8      add new solutions to the population
9      evaluate the current population
10     keep track of the best static schedule
11 end
12 return the best static schedule
```

4. Hybrid scheduling for hybrid clouds (HSHC)

To cost-efficiently improve the performance of workflow execution considering data locality, we present HSHC as an optimization algorithm which consists of two phases: static and dynamic. The static phase is only applied to the private cloud for *improving* the private cloud utilization through effective selection of resources for scheduling, and *reducing* task offloading to the public clouds in regard to cost-efficiency. In fact, this phase mainly deals with finding the optimal placement for the tasks and their corresponding datasets in the private cloud. This is performed by using a genetic algorithm in an offline manner and as a pre-processing step (i.e., a static schedule) for execution time and cost improvement. In the dynamic phase, some tasks are expected to be re-allocated due to changing execution conditions. Also, based on the execution environment, tasks re-allocation can lead to performance improvement.

4.1. Static phase

At the core of static phase is a modified genetic algorithm (Algorithm 1), which deals with the problem of finding the *optimal* placement for datasets and tasks such that the overall delay in execution time is minimized. This requires satisfying fixed dataset constraint (e.g., preserving data privacy) and *pre-scheduling* of the workflow to the most correlated VMs in terms of data locality availability and capacity.

To generate the initial population ($initGeneration$), datasets and tasks are randomly assigned to VMs while making sure the fixed tasks and datasets reside in the designated VMs. In order to have a better initial population, the following heuristics are taken into account. On top of designated VMs for fixed tasks and datasets, potential VMs with high computational power are prioritized to be randomly assigned to the rest of tasks and datasets. A cut-off ratio is also leveraged to avoid overloading a VM disk space as VMs should have enough storage space when retrieving the required datasets.

4.1.1. Chromosome structure

The representation of a solution (chromosome) in our algorithm is shown in Fig. 3. It is a structure that has a length of k and composed of cells. Each cell consists of task-set (VS_i), data-set (DS_i) and a VM (r_i) that hosts the assigned data-set and task-set. This structure represents the overall structure of VMs in real-world examples, e.g., Amazon EC2 instances, which may host multiple tasks and datasets to process tasks simultaneously if the required resources are available.

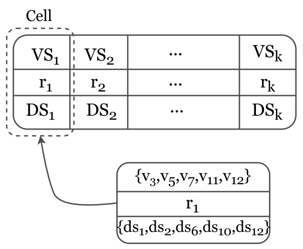

Cell
VS$_1$
VS$_2$
...
VS$_k$
r$_1$
r$_2$
r$_k$
DS$_1$
DS$_2$
DS$_k$
{v$_3$,v$_5$,v$_7$,v$_{11}$,v$_{12}$}
r$_1$
{ds$_1$,ds$_2$,ds$_6$,ds$_{10}$,ds$_{12}$}

Fig. 3. Chromosome structure.

4.1.2. Fitness function

The design of fitness function requires considering several factors that are divided into two main groups; controlling parameters and quality variables. The former determines the eligibility of solutions as valid static schedules against existing constraints. The latter facilitates discovering the most suitable solutions which consist of a mixture of data availability, dependency, and the tasks delays inside a VM. They will be illustrated in this section.

During the construction of a solution, for any task, in order to reduce the delay in execution, this task must be assigned to a VM that results in increasing the number of available datasets for its execution. We denote the percentage of available datasets for task v_i on VM r_j by DA_{ij}. It computes the total required dataset and the available datasets size to determine what percentage of the required datasets exists.

Moreover, for any VM (r_j), the dependency between datasets assigned to this VM and datasets placed at different VMs must be minimized. In other words, the dependency between datasets assigned to the same VM must be maximized. This dependency can be represented as internal DI_{r_j} and external DO_{r_j} dependencies. The former (DI_{r_j}) denotes the correlation degree among datasets, which are placed within a particular VM r_j. The latter (DO_{r_j}) shows the correlation degree between datasets within VM r_j and other available VMs within the private cloud. If VM r_j contains VS_j tasks, they are defined as follows:

$$DI_{r_j} = \sum_{i=0}^{|VS_j|} \sum_{j=0,j\neq i}^{|VS_j|} DP_{ij} \tag{6}$$

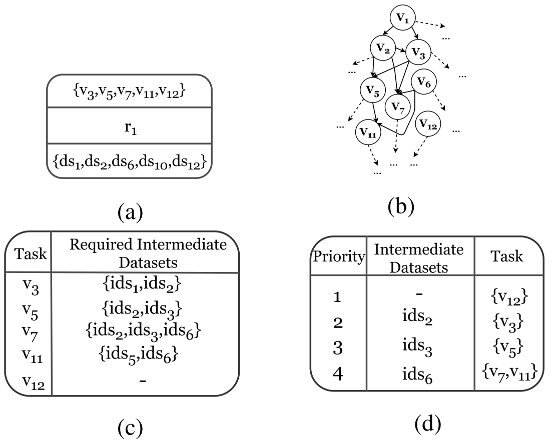
$$DO_{r_j} = \sum_{i=0}^{|VS_j|} \sum_{j=0,j\neq r}^{|R^{prv}|} DP_{i,VS_j} \tag{7}$$

In Eq. (7), DP_{i,VS_j} presents the correlation degree between task v_i of VM r and VS_j of VMs other than r.

Tasks assigned to a VM may have the ability to be executed concurrently. Thus, the waiting time termed as Λ_{ij} is defined based on the concurrent tasks (CV) which is determined by Eq. (8) and is interpreted as a task i execution delay on VM j. This equation helps the fitness function with the selection of solutions that has less concurrent values based on the structure of the workflow and tasks assigned to a VM. In Eq. (8) $|CV_j|$ is the number of tasks that are ahead of task v_i in the concurrent list. The deadline D_i of task v_i in that list is subtracted from its corresponding v_i finish time to be understood that those tasks within the list of that VM are not behind their deadlines. If they are, the value is negative and shows some tasks miss their deadlines on the same VM.

$$\Lambda_{ij} = \sum_{i=0}^{|CV_j|} D_i - FT_{ij} \tag{8}$$

Eq. (8) examines tasks when they will be available for execution in accordance with their required intermediate datasets. They are then categorized, and their execution time is examined against their assigned deadline. If within a VM, there are tasks that they do not need any intermediate datasets, they will also influence concurrent tasks. The lesser the delay, the better the solution is provided. The process is shown in Fig. 4. Once the priorities are ready, the amount of time they are behind their defined deadline is evaluated.


{v$_3$,v$_5$,v$_7$,v$_{11}$,v$_{12}$}
r$_1$
{ds$_1$,ds$_2$,ds$_6$,ds$_{10}$,ds$_{12}$}
(a)
v$_1$
v$_2$
v$_3$
v$_5$
v$_6$
v$_7$
v$_{11}$
v$_{12}$
(b)
(c)
(d)

Task	Required Intermediate Datasets
v$_3$	{ids$_1$,ids$_2$}
v$_5$	{ids$_2$,ids$_3$}
v$_7$	{ids$_2$,ids$_3$,ids$_6$}
v$_{11}$	{ids$_5$,ids$_6$}
v$_{12}$	-

Priority	Intermediate Datasets	Task
1	-	{v$_{12}$}
2	ids$_2$	{v$_3$}
3	ids$_3$	{v$_5$}
4	ids$_6$	{v$_7$,v$_{11}$}

Fig. 4. The *task delay* process on VM (r_j). (a) A cell, (b) a part of an arbitrary workflow, (c) tasks with sorted required intermediate datasets, and (d) tasks priorities within VM.

If tasks needed intermediate datasets rather than those within VM (r_j) which make tasks be executed concurrently, the amount of delay would be implicitly considered within FT_{ij} defined in Eq. (4) as $\sum_{j'=0}^{|IDS_i|} CT_{ij'}$. In other words, the implicit delay for task v_i is obtained when it recursively goes through tasks that produce the required intermediate datasets for the execution of v_i with respect to the VMs they are assigned to.

If each cell within a solution consisted of k cells contains VS_r tasks, the total amount of delay (Λ_T) for a solution is obtained as follows.

$$\Lambda_T = \sum_{j=0}^{k} \sum_{i=0}^{|VS_j|} \Lambda_{ij} \tag{9}$$

To ensure the feasibility of a solution, we use the variable ck_{ds} to check if the assignment for the fixed datasets and tasks does not violate the locality constraints. We also use the variable ck_{ret} to check if the assigned task can retrieve the required datasets from its assigned VM. We use P_r that reflects the overall computational power of the selected VMs. P_r is defined as the summation of relative computing capacities of selected VMs multiplied by the number of CPU cores they have. For example, for a private cloud with four types of VM, computing capacities (except for the number of CPU cores) are normalized. Then normalized computing capacities of all selected VMs are summed up and are multiplied by the number of corresponding CPU cores they have. The higher ratio represents the robustness of schedule. In some situations, a solution may have VMs that do not have either a task-set or data-set. Thus, to evaluate how many VMs are used within a solution, we introduce the variable U_j, which denotes the percentage of the used VM j in the solution. Given these variables, the fitness function (f) aiming at achieving the best static schedule is defined as follows.

$$f = \frac{(ck_{ret} \times ck_{ds} \times \sum_{j=0}^{k} U_j)}{P_r} \times (\sum_{i=0;j=0}^{|VS|,k} DA_{ij} \times (\sum_{i=0;j=0}^{|VS|,k} DI_{r_j} - \sum_{i=0;j=0}^{|VS|,k} DO_{r_j}) - \Lambda_T) \tag{10}$$

The first section of the fitness function is related to the controlling parameters. The parameters ck_{ret} and ck_{ds} ensure the solution meets criteria regarding fixed dataset constraint and the feasibility of transferring task v_i datasets to the designated VMs, respectively. Variables P_r and U_j aim to pick up more competent solutions in terms of computation capacity while the selected VMs are properly in charge

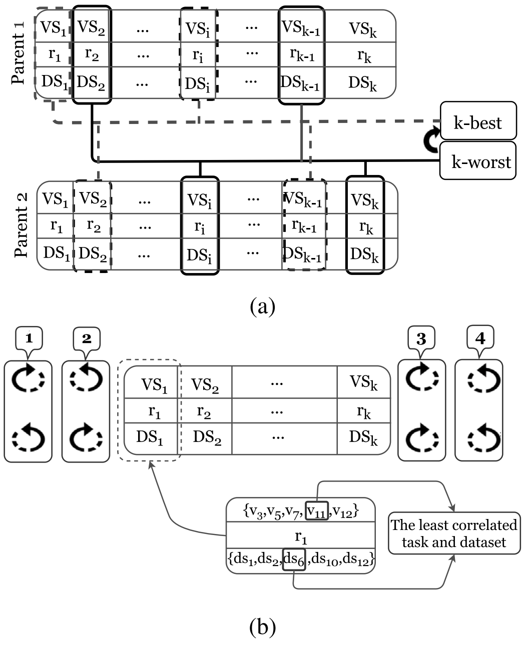


Fig. 5. Genetic algorithm operators. (a) k-way-crossover, (b) the concurrent-rotation process.

of hosting data-sets or task-sets. The second part is considered as the quality variables that facilitate discovering the most suitable solutions.

If the denominator was increased, the fitness function would make sure the selected VMs (U_j) leaning toward resources with higher computation capacities. If this could be aligned with a positive numerator due to the negative impact of Λ_T and/or the difference between DI and DO, the fitness function would lead to expressing solutions in which datasets placed on resources with higher data dependencies while tasks experiencing fewer delays.

4.1.3. Genetic algorithm operators

As solutions are evaluated, the new solution should be produced based on the available ones. Due to the proposed structure of the chromosome (i.e., tasks and datasets correlation), the normal crossover operation cannot be applied to the solutions. Thus, the newly proposed crossover known as *k-way-crossover* is introduced in Fig. 5(a) in which the parameter k shows how many cells within a solution should be selected for the crossover. In this operation, after getting the candidates based on a tournament selection (i.e., a specified number of candidates are randomly selected which eventually the fittest candidate will be chosen), k-worst and k-best cells of each parent are swapped with each other. To select the best/worst cells, cells within a solution are examined against parameter DA_{ij} and the internal dependency (DI). Then, cells are sorted in a descending and ascending order to get the best and the worst cell, correspondingly.

By swapping the best and worst cells of two parents, the solution should be updated as non-swapped cells might also contain some of tasks and/or datasets in the swapped cell. Thus, they should be removed from non-swapped cells of the two parents. If either the best or worst cell had any fixed datasets, they would remain within the cell along with their corresponding tasks.

Concurrent-rotation is introduced to mutate a solution and the structure is shown in Fig. 5(b). It creates four different ways to mutate a solution. For each cell within a solution, the task and dataset with the least dependency are selected and by the direction – clockwise or counterclockwise – they are exchanged with the next or previous cell, correspondingly. The least dependency of task and dataset within a VM r_j is defined with respect to the task required data availability, and how much the data is correlated to tasks assigned to the VM, accordingly. If the least dependency is related to a fixed dataset and its corresponding task, the next least dependency of task and dataset is selected.

The outcome of this phase (the best static schedule) will be used to initially place datasets and assign their corresponding tasks to the most suitable VMs inside a private cloud environment. This outcome would provide better situation during the dynamic phase in terms of execution time as well as data transfer.

4.2. Dynamic phase

Typically, during the actual process of executing the workflow, some VMs may become overloaded or unavailable due to hardware failure and/or storage limitation. In this case, tasks influenced by this restriction will be re-allocated to different VMs. In the dynamic phase, based on the status of the available VMs, task reallocation will be done to another VM(s) in the private cloud or public cloud if their best static schedule (BSS) cannot take place. Toward this end, we devise a task reallocation algorithm (Algorithm 2), which uses a dynamic polynomial algorithm. Given the ready-to-execute tasks, the algorithm maps them to the VMs such that the total delay in execution is minimized and execution time can be improved. This algorithm begins by scheduling tasks which become ready based on their required intermediate datasets in the private cloud. Then, we offload the tasks that cannot be scheduled in the private cloud to be performed in the public cloud considering the proper billing cycle led to the cost-efficiency.

In the beginning, we divide the ready-to-execute tasks into flexible and non-flexible sets. The non-flexible set contains tasks with fixed datasets, and this results in restricting the locality of the VMs, where these tasks must be executed. The flexible set includes the tasks that can be executed at any VMs, as long as necessary criteria such as deadline and storage are met. If those tasks assigned to VMs by the static phase met the criteria, they would be scheduled in their assigned VMs. Our algorithm is mainly concerned with scheduling the flexible tasks.

We start by calculating the available resources (Res) in terms of storage, memory, CPU cores, and workload of the private VMs. We also sort the tasks in descending order based on the number of successors task (v_i) can have. In other words, the tasks with higher successors are prioritized to be executed on the private cloud to reduce the time of data availability for delay minimization. If there were any non-flexible tasks, the workload (and the corresponding resources) would be updated with their estimated finish times on VM(s) assigned to the tasks. To avoid any execution delay considering deadlines for these tasks on the corresponding VMs, it may be necessary to offload any flexible task waiting on their queues to the public clouds. Hence, if there were any, they would be removed, the VM workload would be updated, and any reserved resources would be released as long as making sure the non-flexible tasks would meet their resource requirements and deadlines. During the execution of our algorithm, for each task (v_i) and VM (r_j), we maintain a value that represents the time when r_j can finish executing v_i. This is used to determine the time in which these VMs can execute new tasks. We refer to this value as FT_{ij}. These values are stored in the $FTMatrix$ (line 1). The objective of our algorithm is to determine the identity of the task that can be assigned in each round (for loop line 9). The allocation is established by identifying the task ($v_i \in V$) and the VM ($r_j \in VM$) such that FT_{ij} value is the highest possible value in $FTMatrix$ considering its assigned task deadline and required resources. If assigning v_i to r_j does not violate this task deadline, its required storage, and causes the lower amount of required datasets, this assignment will be confirmed. In this case, we will refer to v_i and r_j as the last confirmed assignment unless the prior task schedule made by *the static phase* does not take place. Otherwise, this task will be added to an offloading list (l_{off}), where all tasks belonging to this list will be scheduled on the public cloud at a later stage (line 28). There is an exception for the list, and it is when the estimated finish time on the private cloud considering its deadline is

Algorithm 2: Task reallocation algorithm

```
Data: F(flexible tasks), NF(non-flexible tasks), private
      cloud $VMs$, and the best static schedule ($BSS$)
Result: FS(Final Schedule)
1  initialize $FTMatrix$ with size $|F| \times |VMs|$, $Res[|VMs|]$, ς
   sort $F$ in descending order w.r.t. successors of $v_i \in F$
2  $l_{off} \leftarrow null, V \leftarrow null$, assign NF tasks
3  for $r_i \in VMs$ do
4      update $Res[r_i]$, ς calculate $r_i$ workload
5      if $r_i \in NF_{VMs}$ then
6          update $Res[r_i]$ and $r_i$ workload w.r.t $FT_{v_{NF},i}$
7      end
8  end
9  for $i \leftarrow 0$ to $|F|$ do
10     for $v_k \in F \setminus (l_{off} \cup V)$ do
11         for $r_j \in VMs$ do
12             if $i = 0$ then
13                 $FTMatrix[v_k][r_j] \leftarrow FT_{kj}$
14             end
15             else
16                 if $r_j = r_c$ and $v_k \neq v_r$ then
17                     $FTMatrix[v_k][r_j] \leftarrow$
                       $FTMatrix[v_k][r_j] + FT_{rj}$
18                 end
19             end
20         end
21     end
22     $[v_r, r_c] = FindMaxValue(FTMatrix, Res)$
23     if $D_{v_r}$ is satisfied ς $BSS_{v_r}$ is violated then
24         assign $v_r$ to $r_c$ ς add $v_r$ to $V$
25         update $FTMatrix$, $Res[r_c]$ ς add $[v_r, r_c]$ to $FS$
26     end
27     else
28         add $v_r$ to $l_{off}$, $v_r \leftarrow null$
29     end
30 end
31 $FS \leftarrow FS \cup$ Public Cloud Assignment ($l_{off}$)
```

Algorithm 3: Public Cloud Assignment

```
Data: $l_{off}$ and the list of public cloud active billing cycles
      $BC^a_{CP}$
Result: PCS(Public Cloud Schedule)
1  $BC^n_{CP} \leftarrow null$
2  for $v_i \in l_{off}$ do
3      if $BC^a_{CP} \neq \emptyset$ then
4          for $r_j \in BC^a_{CP}$ do
5              calculate the $r_j$ workload
6              estimate $DT_{ij}$ within the remaining paid cycle
7              if $DT_{ij} \geq 0$ then
8                  add $[v_i, r_j]$ to $PCS$ ς update $r_j$ workload
                   with $FT_{ij}$ ς remove $v_i$ from $l_{off}$
9              end
10         end
11     end
12 end
13 if $l_{off} \neq \emptyset$ then
14     for $v_i \in l_{off}$ do
15         if $BC^n_{CP} \neq \emptyset$ then
16             for $r_j \in BC^n_{CP}$ do
17                 remove $v_i$ from $l_{off}$ if $DT_{ij} \geq 0$ w.r.t the
                   remaining paid cycle ς update $r_j$ workload
                   with $FT_{ij}$
18             end
19         end
20         estimate the $\Pi^{hour}_{CP_{v_i}}$ of launching a $BC^n_{CP_{hour}}$
21         estimate the $\Pi^{sec}_{CP_{v_i}}$ of launching a $BC^n_{CP_{sec}}$
22         if $\Pi^{sec}_{CP_{v_i}} \leq \Pi^{hour}_{CP_{v_i}}$ then
23             launch $BC^n_{CP_{sec}}$ ς add $[v_i, BC^n_{CP_{sec}}]$ to $PCS$
24         end
25         else
26             launch a $BC^n_{CP_{hour}}$ ς add $BC^n_{CP_{hour}}$ to $BC^a_{CP}$ and
               $BC^n_{CP}$
27             add $[v_i, BC^n_{CP_{hour}}]$ to $PCS$
28         end
29         remove $v_i$ from $l_{off}$
30     end
31 end
```

lower than the task finish time on the public clouds. Thus, the task would be removed from the offloading list and would be assigned to a VM where the task would have the least possible finish time. This is due to reducing delays which might be propagated to the rest of tasks within the workflow.

In the first round, we determine the expected finishing time (FT) for every task at each VM while it considers the required data availability. For any given task and VM, this is calculated by adding the execution time for this task to the time where this VM becomes available to execute a new task (line 13). After the execution of the first round, we examine where the task with the lowest FT_{ij} value can be assigned (in case of its *static phase* schedule does not take place) with the lower amount of required datasets, without violating its deadline, and required storage. If this assignment is confirmed, we update the values in the $FTMatrix$ to state that r_j will be responsible for executing task v_i. If this assignment violates v_i deadline and/or storage, this task will be added to the offloading list (line 28). In both cases, v_i will be removed from consideration on consecutive rounds.

Assuming that task (v_j) is currently being processed and VM (r_j), where the last confirmed assignment in the previous round is represented by v_c to r_c. If r_j is actually r_c, the expected finishing time of task v_j on r_j is increased, since this VM is responsible for executing v_c (line 17). In other situations, the value of the task FT will stay the same. By adopting this strategy, we aim to maximize the number of tasks to be executed in the private cloud while improving performance.

All of the offloaded tasks will be scheduled on the public cloud. Algorithm 3 shows the steps in this scheduling process. Scheduling these tasks uses a strategy similar to the private cloud scheduling taking into account the cloud providers billing policies. We start by checking the active billing cycles if there are any. Then, the workload for all active VMs within BC^a_{CP} is determined. In this algorithm, for each task (v_i) and VM (r_j), we maintain a value (DT_{ij}) that represents the gap between this task deadline (D_i) and the expected finishing time for this task on r_j. In situations where this value is less than zero, this task cannot be executed on this VM (line 7).

In each round, we identify the task ($v_i \in V$) and VM ($r_j \in VM$) such that DT_{ij} value is positive. This is established to minimize the execution cost since any active billing cycles of VMs that satisfies the task deadline will result in cost-free execution. Once they are identified, we perform this assignment and remove this task from consideration in consecutive rounds. Then in each round, the workload of the corresponding VM r_j is updated with FT_{ij}. If there were still some tasks in l_{off} (line 13), it is required to launch new VMs for them. Since different billing cycles exist, we estimate the cost of task execution v_i (lines 20–21). Based on the cost comparison happens (line 22), a VM from the corresponding type of billing cycle is launched considering meeting the task deadline in a cost-effective manner. For any new launched VM we examine whether the unscheduled tasks within l_{off} can be executed *free of charge* with respect to tasks DT and within the remaining time of the new active VM billing cycle (lines 15–18).

4.3. HSHC time complexity

In this section, we discuss the worst-case upper bound for the HSHC running time. HSHC consists of static phase and dynamic phase. The static phase employs a genetic algorithm that its running time depends on the *fitness function* used for evaluation. In this direction, if the static phase has N_P populations over G_I iterations, the static phase will converge in $O(N_P \cdot N_I)$ steps.

The running time of the dynamic phase depends on the number of flexible tasks available ($|F|$) at time τ that will be scheduled onto private cloud ($|VMs|$) and the public cloud. Assume α denotes the number of tasks that will be scheduled on the private cloud, and β represents the number of tasks that will be scheduled on public cloud ($\alpha + \beta = |F|$).

In this case, the runtime complexity for scheduling tasks on the private cloud will take $O(\alpha^2 \cdot |VMs|)$. Beside β, the runtime complexity of scheduling tasks on the public clouds depends also on the active VMs ($|BC_{CP}|$). Thus, the runtime complexity of scheduling tasks on public cloud is $O(\beta \cdot |BC_{CP}|)$, hence, the dynamic phase requires time T_d:

$$T_d \leq O(\alpha^2 \cdot |VMs| + \beta \cdot |BC_{CP}|)$$

Note that in practice, we may assume that $\beta \leq \alpha$ and $|BC_{CP}| \leq |VMs|$. Thus, the bound can be represented as follows:

$$T_d \leq O(\alpha^2 \cdot |VMs|)$$

5. Evaluation

In this section, we evaluate the performance of HSHC in terms of execution time, cost, and deadline compliance. In particular, we have conducted a comparison study with the well-recognized Heterogeneous Earliest-Finish-Time (HEFT) and Critical-Path-on-a-Processor (CPOP) algorithms [25], AsQ [26], HSGA [27], PSO [28], FCFS, and a best-fit algorithm. HEFT and CPOP are novel scheduling algorithms which target at objectives of high performance and providing a fast schedule time on a heterogeneous environment. The building block of HEFT relies on upward ranks for task selection considering processors where minimize tasks earliest finish time with an insertion based approach. In contrast, CPOP calculates upward and downward ranks for task prioritization. It assigns tasks on the critical path to the processor that minimizes the total execution time of those tasks. AsQ intends for scheduling deadline-based applications in a hybrid environment to maximize private cloud utilization and reduce the task response time. It also utilizes a cost strategy for dispatching tasks to the public cloud. HSGA is a hybrid heuristic algorithm that uses genetic algorithm to find the optimal scheduling for workflows. It applies task prioritization based on the workflow topology through evaluation of a task depth in a critical path. The chosen PSO algorithm is a revised version of the standard one in which the inertia weight is gradually reduced and controlled by a decreasing strategy [56] over iteration to provide better convergence speed. FCFS is the traditional scheduling algorithms for assigning tasks to VMs through a queue-based system that the first-assigned task to the VM is prioritized for execution. The best-fit (BF) algorithm aims at assigning tasks to VMs that can adequately fulfill the task requirements.

5.1. Experimental settings

The parameters used for the experiments are extracted from the study [57,58] for the private cloud considering the usage of scientific workflows for the input. The study [57] thoroughly evaluates and profiles workflow characteristics based on real test environments. Hence, we use Montage and LIGO as data-intensive scientific workflows, CyberShake as a computation-intensive workflow, and random workflows generated based on [25] with user-defined deadlines as the input for our simulation. Also, Table 2 shows that tasks for each scientific workflow are assigned the resource requirements with respect to [57].

In our experiments, results are reported based on a small cluster in the private cloud where consists of 20 VMs [59]. Also, results are obtained when the private cloud cluster becomes bigger where has 80 (4x) and 160 VMs (8x) (Section 5.3). The private cloud VMs can be homogeneous or heterogeneous in terms of computing capacity (i.e., VM capacities). This is due to policies which may be followed for decommissioning devices, setting up the same or new ones for the private cloud during its operational time.

For the heterogeneous environment, computation capacities are chosen from quad-core CPUs with 2130MIPS and 2700MIPS processing speeds and dual-core CPUs with 2000MIPS, 2400MIPS, and 2600MIPS CPU speeds with respect to the test bed configuration used in [57]. Each VM has 4 GB RAM, 512 GB disk space, and 1GbE network links. For the homogeneous private cloud the computation capacity of all VMs is set to 2130MIPS which is a quad-core CPU.

The public clouds are simulated based on well-known cloud providers such as Amazon Web Services (AWS), Microsoft Azure [60], and Google Compute Engine [61] where are assumed to provide nearly unlimited resources. General-purpose and computation-optimized VMs are dynamically acquired from the US East region and they are subject to charges with respect to the charging policy (i.e., per hour vs per second). The charging policy also is in effect for bandwidth and storage usage according to the cloud provider rules [4,60,61]. It is also assumed that the data transfer uses the full bandwidth. Moreover, the delay (BT_{CP}) to have a virtual machine ready to process a task is 30 s, 45 s, and 60 s for Google, AWS, and Microsoft, correspondingly.

The workflow deadline is obtained from the synthetic workflows.[1] The size of scientific workflows (i.e., number of tasks) are chosen from 50–1000 for Montage and CyberShake, and 100–1000 for LIGO to evaluate performance of HSHC and the other algorithms. In fact, each experiment setting will be evaluated for the given workflows *similar* to a real-world situation studied in [57].

The random workflow generator accepts the following parameters for creation of DAGs [25]; number of tasks (α), shape parameter (γ) which affects the height ($\frac{\sqrt{\alpha}}{\gamma}$) and width ($\sqrt{\alpha} \times \gamma$) of a workflow, output degree of tasks (θ_o), input data size domain (μ_{in}), number of input datasets required by a task (β_{in}) determined by $\sqrt{\alpha} \times \log \alpha$, the total input datasets (β_{in}^{total}) calculated by $\alpha \times \log \alpha$, output data size domain (μ_{out}), fixed dataset ratio (fd) with probability (p_{fd}), task length domain for execution time estimation (ω), and communication to computation ratio (ccr) for making a DAG data-intensive or computation-intensive. There is also computation on processor ratio parameter (σ) which diversifies the range of computation cost on processors.

The shape and height parameters considering the output degree highlight how tasks are placed in the structure, e.g., the more the height is, the less tasks are dependent when the output degree is small. Larger output degrees increase tasks dependency (i.e., CM_{ij} in the workflow structure) that is also affected by ccr ratio toward being computation-intensive or data-intensive. Hence, with respect to the study [25], considering 0.5 for σ, α has a range of [20–200], γ and ccr choose a ratio from {0.5,1.0,5.0}, and ω has [40000-160000]

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

Table 2
Workflows resource requirements [57].

Montage		LIGO		CyberShake	
Task Name	Memory (MB)	Task Name	Memory (MB)	Task Name	Memory (MB)
mProjectPP	11.81	TmpltBank	404.54	ExtractSGT	20.64
mDiffFit	5.76	Inspiral	533.17	SeisSynth	817.59
mConcatFit	8.13	Thinca	2.63	ZipSeis	6.25
mBgModel	13.64	Inca	2.3	PVCOkaya	3.11
mBackground	16.19	Data_Find	10.06	ZipPeakSA	6.16
mImgTbl	8.06	Inspinj	1.99		
mAdd	16.04	TrigBank	2.04		
mShrink	4.62	Sire	1.93		
mJPEG	3.96	Coire	1.91		

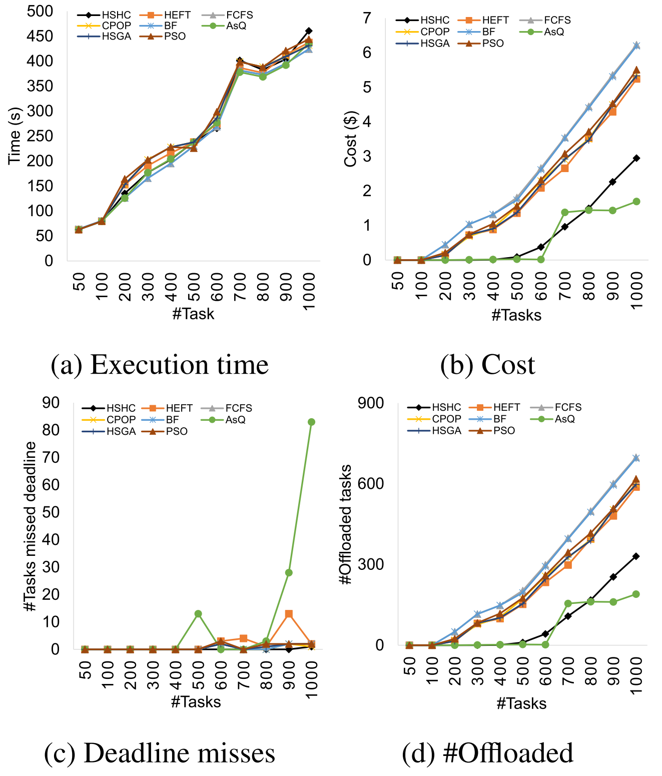


Fig. 6. Montage in the hybrid cloud with a homogeneous private cloud.

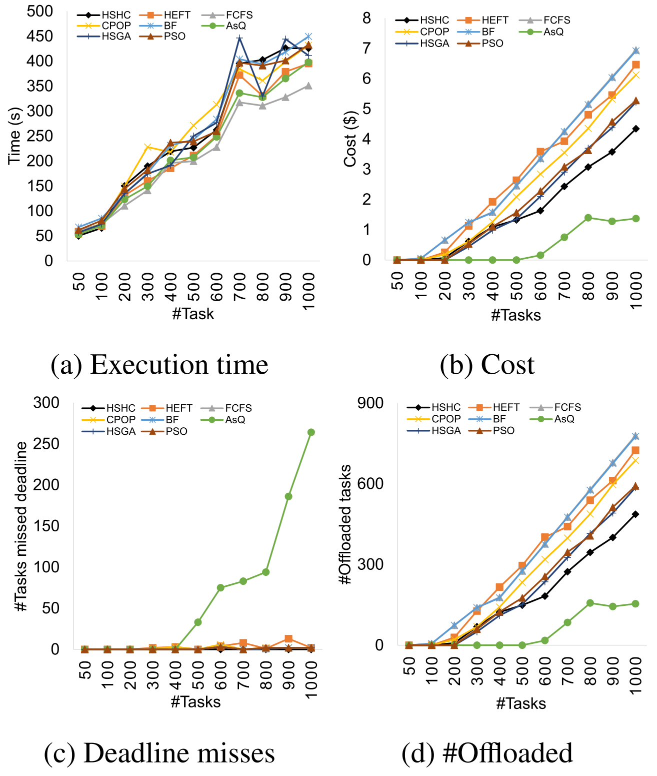


Fig. 7. Montage in the hybrid cloud with a heterogeneous private cloud.

domain. A task memory in MB is selected from [16-2048] with respect to the study [57], and considering θ_o 3, the input and output dataset size in MB are chosen from [64-512], and [32-128], respectively. The results are extracted based on the 10% fixed datasets and tasks with the probability of (p_{fd}) close to 0.7.

For the *static* phase, the initial population size is 70, and max generation is 300. The probability of the mutation after several experiments is considered close to 0.2.

5.2. Results

In this section, we first present the results for scientific and random workflows with the default number of VMs in the private cloud. In Section 5.3, the average result when the private cloud has more resources (i.e., increasing the VMs) is presented.

5.2.1. Montage results

Figs. 6 and 7 show how HSHC responds to a data-intensive workflow in both homogeneous (Figs. 6(a)–6(d)) and heterogeneous (Figs. 7(a)–7(d)) private cloud in comparison to other algorithms. This figure illustrates that HSHC outperformed HSGA, PSO, AsQ, HEFT, and CPOP in terms of cost and deadline. The static phase of HSHC provided the best possible schedule considering the data movements and deadline. In comparison to HSGA as a heuristic algorithm, HSHC could achieve better cost than HSGA and PSO (Figs. 6(b) and 7(b)). This is due to sending fewer tasks to public clouds by HSHC (Figs. 6(d) and 7(d)). HSGA and PSO relied more on public resources but achieved almost the same execution time per each task-set (Figs. 6(a) and 7(a)). Whenever the static schedule could not take place (line 24 in Algorithm 2), HSHC would send off tasks to the public cloud considering the cost-efficient billing cycles.

Although AsQ achieved the minimum execution time, the number of deadline misses is the highest (Figs. 6(c) and 7(c)). FCFS achieved the minimum execution time, however, both FCFS and BF algorithms could not execute the workflow in a cost-efficient way. In contrast, HSHC could utilize the private resources in a way to do the most of execution on VMs capable of meeting the expected deadline and sending off fewer tasks to the public clouds (Fig. 6(d) and Fig. 7(d)) through the dynamic phase (lines 10–21 in Algorithm 2).

Inter-dependency of tasks within the workflow affects the task deadline as an unexpected delay of a task output propagates the delay to the entire workflow. This delay should be properly addressed by

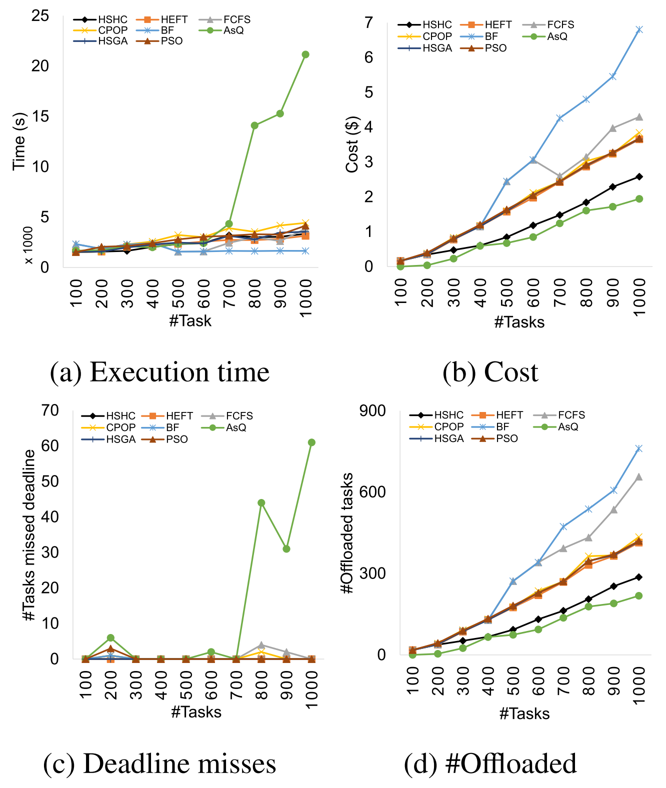


Fig. 8. LIGO in the hybrid cloud with a homogeneous private cloud.

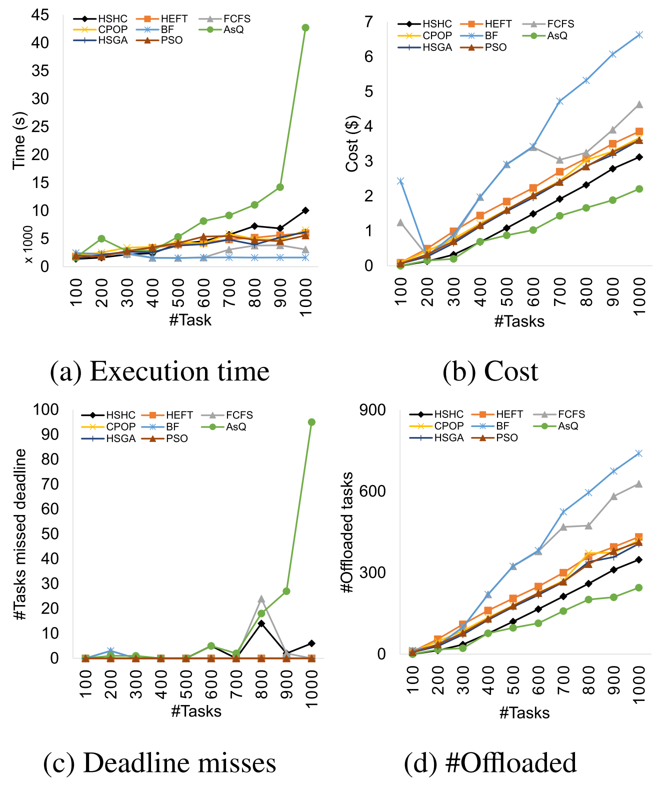


Fig. 9. LIGO in the hybrid cloud with a heterogeneous private cloud.

moving the task(s) to another VM(s) or offloading to the public cloud. Even though HEFT and CPOP seemed to achieve better execution time than HSHC (Figs. 6(a) and 7(a) considering their corresponding costs in Figs. 6(b) and 7(b)), they utilized more public resources. Therefore, our results in Figs. 6 and 7 align with the fact that maximizing the private cloud utilization plays an essential part in a hybrid cloud structure for cost efficiency.

5.2.2. *LIGO results*

Figs. 8 and 9 present results for LIGO as another data-intensive workflow. While AsQ obtained the lowest execution time (after FCFS) for Montage (Fig. 7(a)), it achieved the highest execution time for LIGO due to the different workflow structure which is shown in Fig. 8(a) and Fig. 9(a). They are both data-intensive but the number of concurrent tasks for execution varies based on their required intermediate datasets. These figures show that BF and FCFS have the best execution time, however, they sent off many tasks to the public clouds which incurred higher costs than other algorithms (Figs. 8(b), 8(d) and Figs. 9(b), 9(d)) in both homogeneous and heterogeneous private cloud.

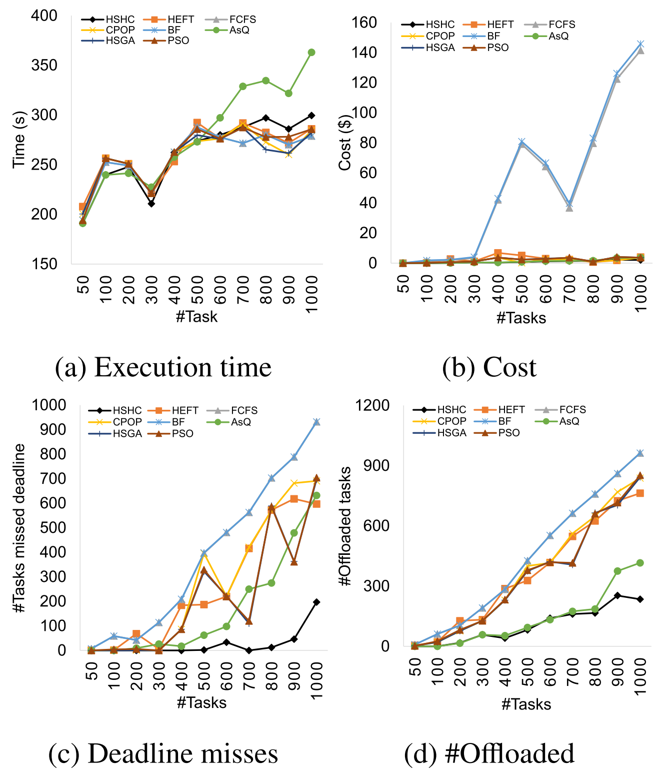
Increasing the number of tasks requires available VMs in which the private cloud could not provide when the workload is high, therefore, it explicitly represents that dispatching tasks to the public cloud is the way to meet the deadline constraint.

HSHC achieved the lowest cost after AsQ which could be inferred from Figs. 8(d) and 9(d) but AsQ has missed more tasks deadlines (Fig. 8(c) and Fig. 9(c)). Also, HSGA and PSO almost achieved the same cost as HEFT and CPOP and higher than HSHC for the homogeneous private cloud (Figs. 8(b) and 9(b)). However, HSGA outperformed HEFT and CPOP, so did PSO, in terms of cost for the heterogeneous private cloud but still achieved higher cost than HSHC due to sending more tasks to the public cloud (Figs. 8(d) and 9(d)). HSHC with the dynamic phase efficiently reacts to the changes in the resource availability within the private cloud during the actual execution. Relying on VMs in the private cloud taking into account deadline compliance caused HSHC achieved relatively higher execution time in comparison to HEFT and CPOP for heterogeneous private cloud. However, it is almost the same for the homogeneous environment as the computation capacities are the same. Therefore, they could not take advantage of their algorithms which are based on the upward and downward rank in addition to scheduling the critical path on the most powerful VMs.

5.2.3. *CyberShake results*

For a computation-intensive workflow, HSHC and other algorithms behaved differently and the results are shown in Figs. 10 and 11. In comparison to Montage (Fig. 6 Fig. 7) and LIGO (Figs. 8 and 9) for homogeneous (and heterogeneous) private cloud, workflow execution under BF and FCFS algorithms incurred the highest cost (Figs. 10(b) and 11(b)). Detailed cost is shown in Table 3. Moreover, HSHC and other algorithms relatively have the same execution time except for AsQ. But Figs. 10(c) and 11(c) show HSHC has the least deadline misses.

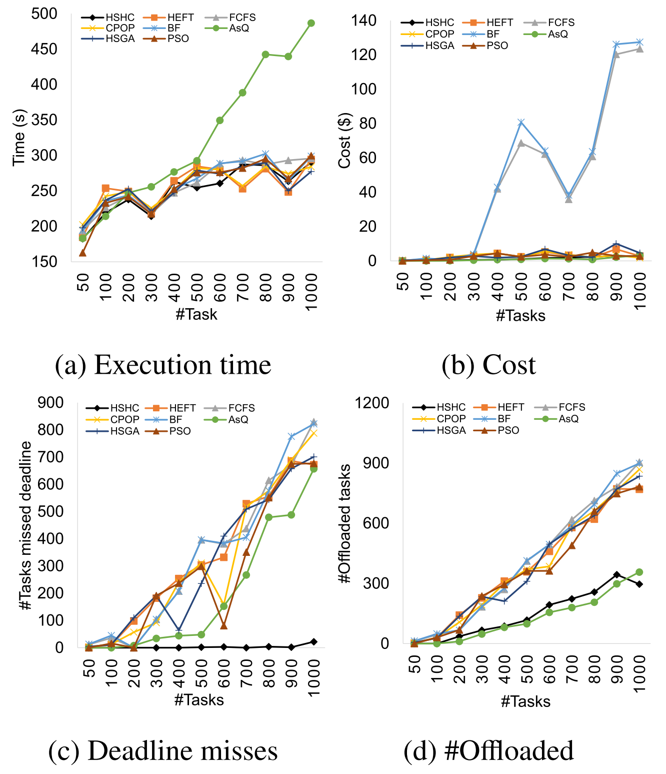
Since VMs in the homogeneous private cloud come with the low computation capacities, consequently, the higher offloading rate to the public cloud is noticeable to stay with the task's deadline which can be inferred from Figs. 10(d) and 11(d). However, the utilization of public cloud leads to having minimum execution time as it would reduce the average execution time but would increase the cost. PSO and HSGA compared to HSHC sent more tasks to the public clouds (Figs. 10(d) and 11(d)) lead to higher cost than HSHC, missed more tasks' deadline (Fig. 10(c) and Fig. 11(c)), and achieved slightly better execution time for the task-set higher than 800 shown in Figs. 10(a) and 11(a). Table 3 shows that AsQ has the lowest cost in comparison to the other algorithms. Considering the cost in Table 3, Figs. 10(a) and 11(a) argue that despite offloading to the public cloud, a delay in a task's output could incur a bigger delay for the rest of tasks within the workflow. Hence, HSHC by prioritizing tasks for scheduling concerning their static schedule (line 1 in Algorithm 2) with higher successors diminished the effect of delays which led to the lowest deadline misses for the homogeneous private cloud. Therefore, HSHC performed cost-efficiently while keeping execution time close to the other algorithms.



(a) Execution time (b) Cost

(c) Deadline misses (d) #Offloaded

Fig. 10. CyberShake in the hybrid cloud with a homogeneous private cloud.



(a) Execution time (b) Cost

(c) Deadline misses (d) #Offloaded

Fig. 11. CyberShake in the hybrid cloud with a heterogeneous private cloud.

5.2.4. Random workflow results

Figs. 12, 14, and 16 present the results for a computation-intensive random workflow to a data-intensive workflow in a homogeneous private cloud (the lower ccr, the higher computation-intensiveness and vice versa). These figures are followed by Figs. 13, 15, and 17 for the heterogeneous private cloud. These figures illustrate that FCFS and BF ended up having a higher cost than HSHC as well as CPOP and HEFT. Also, HEFT and CPOP achieved higher cost than HSHC while they relatively had higher execution time. HSHC compared to HSGA achieved lower cost (HSGA has achieved almost the same cost as HEFT and PSO in Fig. 12(b)) but similar to HSHC in Fig. 13(b). As the random workflow structure is changed when ccr ratio is increased, it is expected that it could have an indirect impact on offloading tasks to the public cloud and consequently, the cost of execution and its execution time.

By making a comparison between Fig. 12a and Fig. 16a (and their heterogeneous results in Figs. 13a and 17a), it is understood that how ccr ratio can affect the execution time when a workflow becomes more data-intensive. Also, Figs. 12d, 14d, and 16d (and for the heterogeneous private cloud Figs. 13d, 15d, and 17d) represent that for different random workflow structures, different number of dispatched tasks are expected. This could be explained from different perspectives: (1) the private cloud could handle all concurrent tasks that become activated due to their required intermediate datasets and (2) the level of task concurrency differs from a workflow to another workflow. This concurrency also affects the private VMs workload at different execution stages which might be high or low. Hence, having all the execution within a private cloud can cause a few tasks to be sent off to the public cloud which could be inferred from Figs. 12b, 14b, and 16b (and for the heterogeneous private cloud Figs. 13b, 15b, and 17b).

Fig. 12c (and Fig. 13c), Fig. 14c (and Fig. 15c), and Fig. 16c (and Fig. 17c) can explicitly explain that to avoid missing tasks deadlines, they have to be offloaded to the public cloud. However, a delay in the output of the tasks could cause tasks missing their deadlines. As a result, BF, FCFS, HEFT, HSGA, PSO, and CPOP have a higher number of deadline misses, and HSHC has the lowest deadline misses (along with the better execution time) for a random data-intensive workflow. The dynamic nature of HSHC when the best static schedule cannot take place for the VM selections and passive strategies followed by other algorithms have created differences regarding deadline misses. Moreover, Fig. 12d (and Fig. 13d), Fig. 14d (and Fig. 15d), and Fig. 16d (and Fig. 17d) illustrate that HSHC sent off fewer tasks in comparison to other baseline algorithms, in particular, HSGA (and PSO). This explains why HSHC outperformed the algorithms in terms of cost.

Table 3
CyberShake cost ($) in the hybrid cloud.

#Tasks	HSHC	HEFT	FCFS	CPOP	BF	AsQ	HSGA	PSO
(a) With a homogeneous private cloud.								
50	0.00	0.05	0.08	0.02	0.08	0.00	0.03	0.02
100	0.00	0.20	1.61	0.19	1.9	0.00	0.19	0.22
200	0.16	2.80	2.03	0.69	2.27	0.14	0.69	0.73
300	0.52	1.19	3.51	1.13	4.00	0.52	1.13	1.13
400	0.37	6.85	42.11	3.77	43.03	0.47	3.77	3.78
500	0.73	5.16	79.16	0.22	80.92	0.84	2.43	2.44
600	1.25	3.01	64.23	2.71	66.66	1.44	3.05	2.78
700	1.44	2.40	36.72	2.45	39.70	1.55	3.63	3.71
800	1.48	0.75	79.75	1.08	83.14	1.66	1.07	1.07
900	2.26	1.87	122.36	1.97	126.20	3.53	4.17	4.14
1000	2.10	4.00	141.5	3.68	145.82	3.71	3.68	3.68
(b) With a heterogeneous private cloud.								
50	0.00	0.02	0.12	0.40	0.12	0.00	0.40	0.00
100	0.00	0.52	1.20	0.52	1.46	0.00	0.41	0.40
200	0.32	2.03	0.61	2.29	0.60	0.10	2.01	0.62
300	0.60	2.82	3.43	3.67	3.95	0.43	2.82	2.82
400	0.78	4.42	41.99	4.42	42.96	0.72	1.89	4.42
500	1.04	2.43	68.69	2.43	80.80	0.88	2.43	2.43
600	1.72	5.17	62.16	5.47	64.13	1.39	6.83	3.60
700	1.99	3.26	35.92	3.65	38.36	1.25	3.25	2.41
800	2.29	2.08	60.81	2.07	63.70	0.86	2.07	4.89
900	3.07	6.86	120.29	3.28	126.15	2.19	10.07	2.92
1000	2.65	2.78	123.57	1.78	127.45	3.18	4.63	2.84

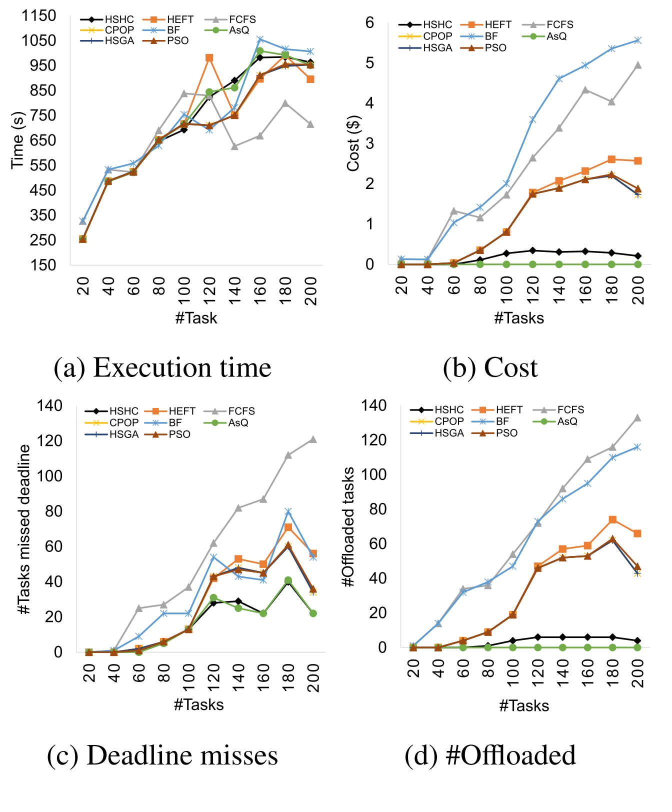

(a) Execution time (b) Cost (c) Deadline misses (d) #Offloaded

Fig. 12. Random (γ=ccr=0.5) in the hybrid cloud with a homogeneous private cloud.

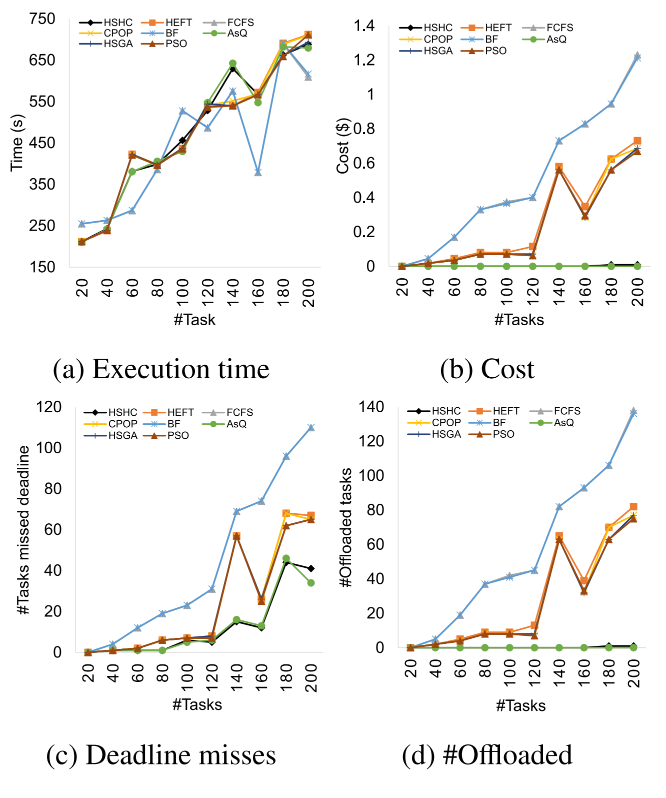

(a) Execution time (b) Cost (c) Deadline misses (d) #Offloaded

Fig. 14. Random (γ=ccr=1.0) in the hybrid cloud with a homogeneous private cloud.

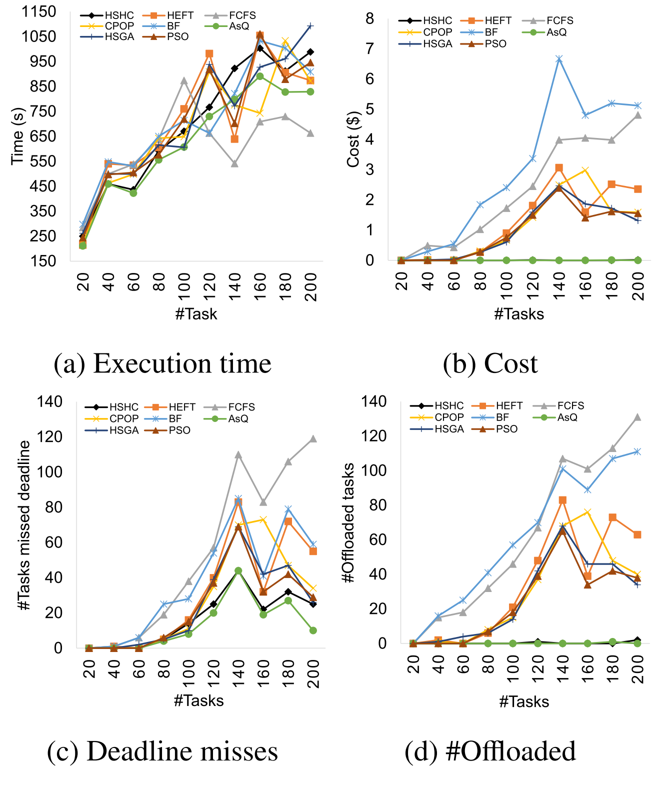

(a) Execution time (b) Cost (c) Deadline misses (d) #Offloaded

Fig. 13. Random (γ=ccr=0.5) in the hybrid cloud with a heterogeneous private cloud.

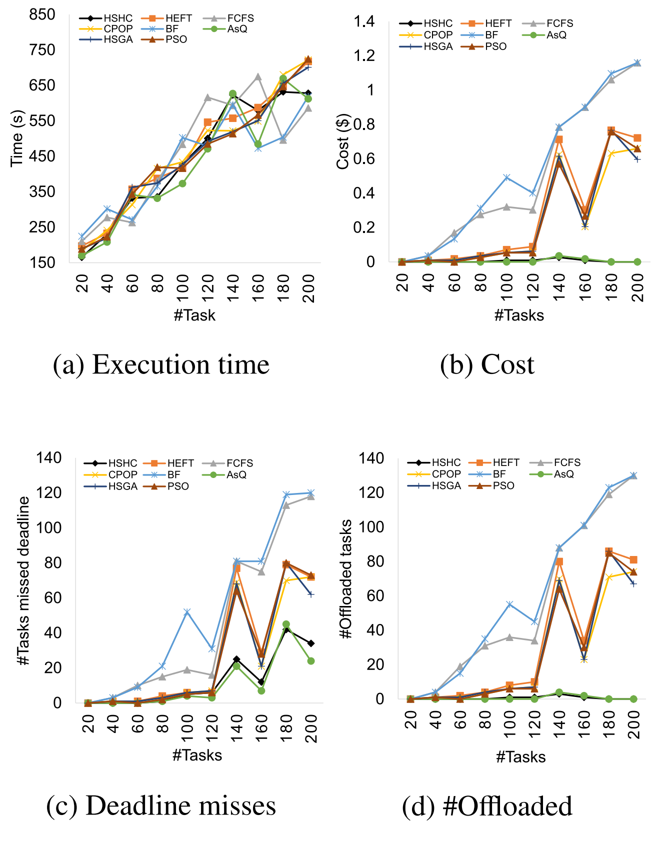

(a) Execution time (b) Cost (c) Deadline misses (d) #Offloaded

Fig. 15. Random (γ=ccr=1.0) in the hybrid cloud with a heterogeneous private cloud.

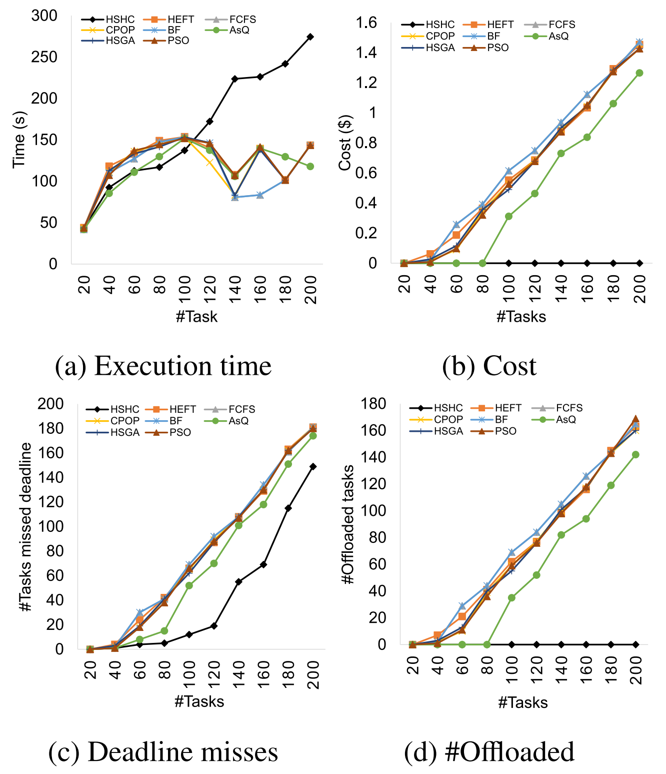


Fig. 16. Random (γ=ccr=5.0) in the hybrid cloud with a homogeneous private cloud.

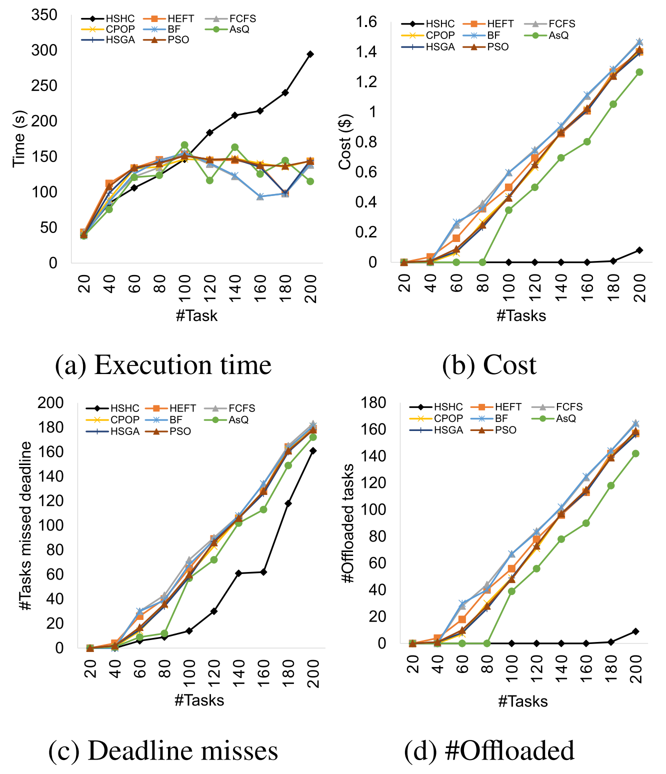


Fig. 17. Random (γ=ccr=5.0) in the hybrid cloud with a heterogeneous private cloud.

In fact, assigning tasks to available VMs either on the private or public cloud would not guarantee that tasks could be performed within the expected deadline. For example, if the queue-based structure of FCFS cannot have the right vision for the tasks within the queue, it leads to dispatching many tasks to a VM for execution which ends up starving tasks at the back of the queue. From a cost perspective, it also explains why HSHC achieves the lower cost. The lower the offloading rate, the lower the cost of public cloud utilization proved by HSHC. However, the offloading process may be influenced by the workflow structure as for random workflows in Fig. 14b (and Fig. 15b) the costs were decreased for algorithms HEFT and CPOP that HSGA (and PSO) followed the similar pattern as HEFT. Therefore, HSHC by the right usage of data locality as well as the proper assignment of their corresponding tasks outperformed the other algorithms in both random and scientific workflows.

5.3. *Increasing private cloud VMs*

In this section, the number of private cloud VMs is increased to 160. In particular, the private cloud has 4x and 8x more VMs. Although adding more VMs will reduce the usage of hybrid cloud, it will increase the capital and operational expenditure for the private cloud lifetime [62]. Average results for Montage, LIGO, CyberShake, and random workflows are reported (Tables 4 and 5) in both homogeneous and heterogeneous private cloud. Increasing the private cloud VMs will provide more computation capacities, and this will show how HSHC and baseline algorithms respond to this change.

5.3.1. *Scientific workflows*

It is inferred from Table 4c that HEFT, CPOP, HSGA, AsQ, PSO still have higher deadline misses in comparison to HSHC. They also have the highest execution time for LIGO workflow. Since the number of available VMs within the private cloud was increased, HSHC relied more on the private instances in comparison to HSGA, PSO, HEFT and CPOP. Therefore, HSHC relatively achieved better or equal execution time for Montage and CyberShake (Table 4a) but better than HSGA and PSO. Having execution on the private VMs reduces the total cost of execution as it is seen that after AsQ, HSHC has the lowest cost among algorithms in Table 4b. Table 4b also shows that both PSO and HSGA could not outperform HSHC in terms of cost even when more resources were available in the private cloud. PSO could relatively reduce the cost in comparison to HSGA, however, it is not comparable to HSHC.

This minimized cost is also aligned with Table 4d as HSHC has the lower offloaded jobs to the public cloud. However, the overall cost for each workflow is reduced compared to the situation where the private cloud had fewer VMs (Table 4c 20 VMs vs 160 VMs). BF and FCFS for Montage workflows have the highest cost, and consequently, have the highest offloaded tasks to the public cloud without any significant impact on the execution performance (i.e., execution time) (Table 4c). While HSHC achieved a lower cost for LIGO, HEFT and CPOP sent off more tasks to the public cloud and had higher costs than HSHC but their scheduling decisions could not improve the execution time. Usage of public resources should have helped the algorithms as the schedule decisions made by them were done by pre-processing the workflows and assigning tasks to the most powerful VMs in the private cloud. HSHC could execute CyberShake workflow (Table 4c) cost-efficiently and could provide better performance than HEFT, CPOP, and AsQ for a heterogeneous private cloud.

5.3.2. *Random workflows*

Increasing the private cloud VMs for random workflows did not help BF and FCFS algorithms to reduce task offloading to the public cloud. Table 5d shows that HEFT and CPOP also sent off more tasks to the public cloud which led to a higher cost than HSHC for a data-intensive workflow. HSHC relatively achieved higher execution time in comparison to HEFT, CPOP, and AsQ, in particular, when the random workflow

Table 4
Average results for scientific workflows when the private cloud's VMs are increased.

#VMs	20						80						160					
Algo.	Montage		LIGO		CyberShake		Montage		LIGO		CyberShake		Montage		LIGO		CyberShake	
	Hom.	Het.	Hom.	Het.	Hom.	Het.	Hom.	Het.	Hom.	Het.	Hom.	Het.	Hom.	Het.	Hom.	Het.	Hom.	Het.
(a) Execution time (s).																		
HSHC	256	256	2414	4611	261	251	233	202	1594	1565	251	232	230	186	1594	1469	251	227
HEFT	257	231	2360	3994	263	256	237	202	1594	2433	261	254	230	195	1594	2016	260	248
FCFS	246	211	2392	2555	259	259	231	192	2570	2635	255	247	231	192	2576	2370	251	242
CPOP	262	262	3026	4214	259	257	240	204	1743	2313	261	242	230	192	1594	1997	254	238
BF	246	265	1858	1849	259	260	231	199	2500	2644	255	251	231	199	2507	2418	251	251
AsQ	249	226	6716	10300	280	325	230	188	2726	9316	251	227	230	184	2631	6374	251	215
HSGA	261	254	2515	3687	259	256	239	216	1688	2016	258	239	230	199	1594	1839	254	237
PSO	265	257	2786	4012	261	255	243	225	1850	2282	259	253	231	213	1719	2033	256	247
(b) Cost ($).																		
HSHC	0.74	1.65	1.18	1.39	0.94	1.31	0.00	0.09	0.80	0.17	0.00	0.03	0.00	0.00	0.81	0.00	0.00	0.00
HEFT	1.90	2.75	1.82	2.02	2.57	2.94	0.22	0.13	0.87	0.73	1.06	0.99	0.00	0.11	0.81	0.50	0.74	0.41
FCFS	2.44	2.89	2.20	2.55	52.10	47.16	0.86	1.35	1.14	1.11	51.05	49.06	0.02	0.37	1.08	0.66	51.03	47.30
CPOP	1.99	2.40	1.88	1.86	1.63	2.73	0.35	0.13	0.92	0.76	1.58	1.05	0.00	0.00	0.81	0.31	0.40	0.37
BF	2.42	2.88	2.93	3.47	53.97	49.97	0.86	1.35	1.26	1.19	52.77	51.77	0.02	0.37	1.20	0.75	52.63	51.51
AsQ	0.55	0.45	0.89	1.01	1.26	1.00	0.00	0.00	0.34	0.01	0.00	0.00	0.00	0.00	0.17	0.00	0.00	0.00
HSGA	1.96	1.92	1.84	1.77	2.17	3.35	0.98	0.39	0.59	0.51	1.07	0.85	0.00	0.00	0.53	0.30	0.74	1.01
PSO	2.07	2.01	1.86	1.80	2.16	2.49	0.91	0.48	0.62	0.61	1.02	1.10	0.00	0.00	0.53	0.33	0.67	0.68
(c) #Deadline misses.																		
HSHC	0	0	0	3	26	3	0	0	1	1	0	1	0	0	1	0	0	0
HEFT	2	3	0	0	261	330	1	0	1	1	1	1	0	0	1	1	1	0
FCFS	1	1	1	3	390	336	0	0	1	1	51	49	0	0	1	1	51	47
CPOP	1	2	0	0	279	313	0	0	1	1	2	1	0	0	1	0	0	0
BF	1	1	0	0	391	339	0	0	1	1	53	52	0	0	1	1	53	52
AsQ	12	67	14	15	168	198	0	0	0	0	0	0	1	0	0	0	0	0
HSGA	1	67	0	0	217	313	0	0	0	3	109	101	0	0	0	0	73	70
PSO	1	1	0	0	220	280	0	0	0	0	111	122	0	0	0	0	68	43
(d) #Offloaded tasks.																		
HSHC	83	185	131	154	105	147	0	10	88	19	0	3	0	0	89	0	0	0
HEFT	213	308	205	228	362	389	25	14	96	80	158	142	0	13	89	56	48	64
FCFS	274	324	291	322	443	409	96	152	99	82	378	351	2	42	92	32	343	306
CPOP	223	269	213	212	372	388	39	15	103	84	167	152	0	0	89	34	60	29
BF	271	323	327	361	443	412	96	152	99	83	378	365	2	42	92	35	343	326
AsQ	62	51	99	114	137	131	0	0	37	1	0	0	0	0	18	0	0	0
HSGA	220	215	208	200	352	386	110	43	65	57	140	164	0	0	59	34	94	119
PSO	232	225	210	205	356	367	102	55	69	68	139	125	0	0	59	37	95	63

is computation-intensive toward a data-intensive workflow. HSHC also has higher execution time compared to the meta-heuristic algorithms (Table 5a) due to more usage of public resources by HSGA and PSO and sending off more tasks to the public clouds (Table 5d). HSHC used more private resource than sending tasks to the public cloud. Table 5d shows that HSHC has a lower offloading rate to the public cloud which implies a lower cost than other algorithms (Table 5b).

It is inferred from Table 5 that HSHC has the highest execution time when the workflow is data-intensive but has the least number of deadline misses Table 5c). For other *ccr* ratios and when the number of VMs is 80, AsQ has the minimum deadline misses but HSGA and PSO (for $ccr = 0.5$ and $ccr = 1.0$), HEFT and CPOP have higher rates.

Even though other algorithms relied on public resources, they could not reduce deadline misses. As 10% of datasets are fixed for random workflows, HSHC could outperform HSGA, PSO, HEFT, CPOP, and AsQ considering cost, deadline misses, and the number of offloaded tasks. Therefore, due to availability of fixed datasets for random workflows, scheduling decisions need to consider this constraint. Results for random workflows show HSHC is capable of assigning tasks to the private cloud resource taking into the constraints.

6. Conclusion

In this paper, we have presented a novel data-locality aware scheduling algorithm (HSHC) for efficient execution of scientific workflows in a hybrid cloud environment. The scheduling of increasingly large-scale and data-intensive scientific workflows becomes further complicated with the dynamic nature of hybrid clouds. We have identified that the locality of data (intermediate output data in particular) is a key for efficient execution. To this end, we have adopted a hybrid scheduling approach with static phase and dynamic phase. Our hybrid scheduling approach is proven to make schedules robust by reacting to changing execution conditions including data locality. Also, considering the billing cycle policies that public clouds maintain can further reduce the cost of workflow execution when a private cloud offloads tasks. Results showed our approach effectively deals with the dynamicity of hybrid clouds and achieved better results compared to well-known HEFT and CPOP algorithms, HSGA and PSO as meta heuristic algorithms, FCFS, Best-Fit, and AsQ in terms of execution time and cost.

For future work, we are going to consider the private cloud utility cost and its contribution to the cost-efficiency. Moreover, resource allocation across the hybrid cloud can be facilitated through feasible recommendations provided by machine learning techniques in which job characteristics and resource requirements are learned. In this way, the static phase and/or dynamic phase can be assisted for the selection of cost-efficient resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Table 5
Average results for random workflows when the private cloud's VMs are increased.

#VMs	20						80						160					
Algo.	γ=ccr=0.5		γ=ccr=1.0		γ=ccr=5.0		γ=ccr=0.5		γ=ccr=1.0		γ=ccr=5.0		γ=ccr=0.5		γ=ccr=1.0		γ=ccr=5.0	
	Hom.	Het.	Hom.	Het.	Hom.	Het.	Hom.	Het.	Hom.	Het.	Hom.	Het.	Hom.	Het.	Hom.	Het.	Hom.	Het.
(a) Execution time (s).																		
HSHC	725	700	477	445	164	164	720	699	473	448	171	155	720	699	474	443	170	156
HEFT	715	711	476	465	123	126	696	711	476	449	122	124	696	685	477	437	125	125
FCFS	655	614	446	458	114	114	659	632	459	473	120	115	661	631	459	473	120	115
CPOP	691	681	477	458	118	126	696	694	476	449	124	127	696	682	477	429	124	124
BF	735	717	447	434	114	115	732	711	459	449	120	115	733	710	460	438	119	114
AsQ	728	633	477	429	115	119	726	613	476	404	168	155	727	612	476	407	165	157
HSGA	691	718	471	449	120	124	696	699	476	445	125	128	697	675	477	433	125	128
PSO	692	705	472	453	123	129	697	710	477	452	125	128	697	698	477	443	125	128
(b) Cost ($).																		
HSHC	0.19	0.00	0.00	0.01	0.00	0.01	0.18	0.00	0.00	0.00	0.00	0.00	0.18	0.00	0.00	0.00	0.00	0.00
HEFT	1.25	1.25	0.26	0.27	0.65	0.63	1.14	1.00	0.24	0.23	0.61	0.58	1.14	0.89	0.24	0.22	0.60	0.57
FCFS	2.38	2.30	0.51	0.50	0.68	0.67	2.38	2.29	0.50	0.49	0.62	0.60	2.38	2.29	0.50	0.49	0.62	0.60
CPOP	1.09	1.11	0.24	0.23	0.63	0.59	1.14	0.87	0.24	0.21	0.60	0.57	1.14	0.83	0.24	0.19	0.60	0.57
BF	2.88	3.02	0.50	0.53	0.68	0.67	2.91	2.75	0.50	0.52	0.62	0.60	2.88	2.75	0.50	0.51	0.62	0.60
AsQ	0.00	0.00	0.00	0.01	0.47	0.47	0.00	0.00	0.00	0.00	0.08	0.12	0.00	0.00	0.00	0.00	0.08	0.12
HSGA	0.99	0.71	0.26	0.26	0.70	0.66	1.04	0.71	0.27	0.26	0.67	0.71	1.04	0.78	0.27	0.28	0.67	0.71
PSO	1.01	0.86	0.26	0.30	0.70	0.66	1.04	0.74	0.27	0.21	0.66	0.71	1.04	0.79	0.27	0.23	0.67	0.71
(c) #Deadline misses.																		
HSHC	16	17	13	13	43	46	16	17	12	13	44	39	16	17	12	12	45	40
HEFT	29	30	24	28	80	80	27	25	24	25	79	78	27	20	24	24	79	77
FCFS	55	54	44	45	82	83	55	53	45	42	80	79	56	54	45	42	80	79
CPOP	25	28	24	25	80	77	27	21	24	23	77	77	27	19	24	21	79	78
BF	33	38	44	52	82	81	33	33	45	49	80	79	33	33	45	47	80	79
AsQ	16	13	12	11	69	69	15	11	12	8	64	62	16	11	13	8	65	63
HSGA	19	17	26	28	47	43	21	16	26	29	46	45	21	15	26	30	46	46
PSO	20	20	26	33	47	43	21	15	27	24	46	46	21	19	27	26	47	46
(d) #Offloaded tasks.																		
HSHC	3	0	0	1	0	1	3	0	0	0	0	0	3	0	0	0	0	0
HEFT	34	34	29	31	73	70	31	26	27	26	68	65	30	23	27	25	68	64
FCFS	66	63	57	56	77	76	66	63	56	55	69	67	66	63	56	55	70	67
CPOP	29	29	27	26	71	67	30	22	27	24	67	64	30	22	27	21	67	64
BF	61	62	56	60	77	76	62	58	56	59	69	67	62	58	56	57	70	67
AsQ	0	0	0	1	52	52	0	0	0	0	9	13	0	0	0	0	9	13
HSGA	26	19	29	29	79	74	28	18	30	29	75	80	28	20	30	31	75	80
PSO	27	33	30	34	78	75	28	20	30	24	75	80	28	21	30	27	75	81

References

[1] VMware, 2019, https://www.vmware.com VMware, Inc..

[2] XenProject, 2019, https://www.xenproject.org/, A Linux Foundation Collaborative Project.

[3] S. Sharif, P. Watson, J. Taheri, S. Nepal, A.Y. Zomaya, Privacy-aware scheduling SaaS in high performance computing environments, IEEE Trans. Parallel Distrib. Syst. 28 (4) (2017) 1176–1188, http://dx.doi.org/10.1109/TPDS.2016.2603153.

[4] AWS, 2019, https://aws.amazon.com/, Amazon Web Service(AWS), Inc..

[5] A.P. Heath, M. Greenway, R. Powell, J. Spring, R. Suarez, D. Hanley, C. Bandlamudi, M.E. McNerney, K.P. White, R.L. Grossman, Bionimbus: a cloud for managing, analyzing and sharing large genomics datasets, J. Am. Med. Inform. Assoc. 21 (6) (2014) 969–975.

[6] R. Van den Bossche, K. Vanmechelen, J. Broeckhove, Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds, Future Gener. Comput. Syst. 29 (4) (2013) 973–985.

[7] M. Malawski, K. Figiela, J. Nabrzyski, Cost minimization for computational applications on hybrid cloud infrastructures, Future Gener. Comput. Syst. 29 (7) (2013) 1786–1794.

[8] B. Lin, W. Guo, X. Lin, Online optimization scheduling for scientific workflows with deadline constraint on hybrid clouds, Concurr. Comput.: Pract. Exper. 28 (11) (2016) 3079–3095.

[9] B. Lin, W. Guo, N. Xiong, G. Chen, A.V. Vasilakos, H. Zhang, A pretreatment workflow scheduling approach for big data applications in multicloud environments, IEEE Trans. Netw. Serv. Manag. 13 (3) (2016) 581–594.

[10] M. Rahman, X. Li, H. Palit, Hybrid heuristic for scheduling data analytics workflow applications in hybrid cloud environment, in: Parallel and Distributed Processing Workshops and Phd Forum IPDPSW, 2011 IEEE International Symposium on, IEEE, 2011, pp. 966–974.

[11] L.F. Bittencourt, E.R.M. Madeira, HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds, J. Internet Serv. Appl. 2 (3) (2011) 207–227.

[12] S. Abrishami, M. Naghibzadeh, D.H.J. Epema, Cost-driven scheduling of grid workflows using partial critical paths, IEEE Trans. Parallel Distrib. Syst. 23 (8) (2012) 1400–1414, http://dx.doi.org/10.1109/TPDS.2011.303.

[13] S. Abrishami, M. Naghibzadeh, D.H.J. Epema, Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds, Future Gener. Comput. Syst. 29 (1) (2013) 158–169.

[14] E.-K. Byun, Y.-S. Kee, J.-S. Kim, E. Deelman, S. Maeng, BTS: Resource capacity estimate for time-targeted science workflows, J. Parallel Distrib. Comput. 71 (6) (2011) 848–862.

[15] F.J. Clemente-Castelló, R. Mayo, J.C. Fernández, Cost model and analysis of iterative mapreduce applications for hybrid cloud bursting, in: IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID, 2017, pp. 858–864.

[16] F.J. Clemente-Castello, B. Nicolae, M.M. Rafique, R. Mayo, J.C. Fernandez, Evaluation of data locality strategies for hybrid cloud bursting of iterative mapreduce, in: 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2017, pp. 181–185.

[17] H. Chu, Y. Simmhan, Cost-efficient and resilient job life-cycle management on hybrid clouds, in: IEEE 28th International Parallel and Distributed Processing Symposium, 2014, pp. 327–336.

[18] M.R. Hoseinyfarahabady, H.R.D. Samani, L.M. Leslie, Y.C. Lee, A.Y. Zomaya, Handling uncertainty: Pareto-efficient BoT scheduling on hybrid clouds, in: 2013 42nd International Conference on Parallel Processing, 2013, pp. 419–428.

[19] H. Yuan, J. Bi, W. Tan, M. Zhou, B.H. Li, J. Li, TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds, IEEE Trans. Cybern. 47 (11) (2016) 3658–3668.

[20] A. Pasdar, K. Almi'ani, Y.C. Lee, Data-aware scheduling of scientific workflows in hybrid clouds, in: International Conference on Computational Science, ICCS, 2018, pp. 708–714.

[21] M.K. Richard, Effective heuristics for NP-hard problems, in: Workshop on beyond Worst-Case Analysis, 2011.

[22] J.C. Jacob, D.S. Katz, G.B. Berriman, J.C. Good, A.C. Laity, E. Deelman, C. Kesselman, G. Singh, M.H. Su, T.A. Prince, R. Williams, Montage; a grid portal and software toolkit for science grade astronomical image mosaicking, Int. J. Comput. Sci. Eng. 4 (2) (2009) 73–87.

[23] A. Abramovici, W.E. Althouse, R.W.P. Drever, Y. Gürsel, S. Kawamura, F.J. Raab, D. Shoemaker, L. Sievers, R.E. Spero, K.S. Thorne, R.E. Vogt, R. Weiss, S.E. Whitcomb, M.E. Zucker, LIGO: The laser interferometer gravitational-wave observatory, Science 256 (5055) (1992) 325–333.
[24] CyberShake, 2019, https://scec.usc.edu/scecpedia/CyberShake, CyberShake.
[25] H. Topcuoglu, S. Hariri, M.-Y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE Trans. Parallel Distrib. Syst. 13 (3) (2002) 260–274.
[26] W.-J. Wang, Y.-S. Chang, W.-T. Lo, Y.-K. Lee, Adaptive scheduling for parallel tasks with qos satisfaction for hybrid cloud environments, J. Supercomput. 66 (2) (2013) 783–811.
[27] A.G. Delavar, Y. Aryan, HSGA: a hybrid heuristic algorithm for workflow scheduling in cloud systems, Cluster Comput. 17 (1) (2014) 129–137.
[28] X. Huang, C. Li, H. Chen, D. An, Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies, Cluster Comput. (2019) 1–11.
[29] D. Yuan, Y. Yang, X. Liu, J. Chen, A data placement strategy in scientific cloud workflows, Future Gener. Comput. Syst. 26 (8) (2010) 1200–1214.
[30] K. Deng, J. Song, K. Ren, D. Yuan, J. Chen, Graph-cut based coscheduling strategy towards efficient execution of scientific workflows in collaborative cloud environments, in: 2011 IEEE/ACM 12th International Conference on Grid Computing, 2011, pp. 34–41.
[31] M. Malawski, G. Juve, E. Deelman, J. Nabrzyski, Algorithms for cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds, Future Gener. Comput. Syst. 48 (2015) 1–18.
[32] A. Deldari, M. Naghibzadeh, S. Abrishami, CCA: a deadline-constrained workflow scheduling algorithm for multicore resources on the cloud, J. Supercomput. 73 (2) (2017) 756–781.
[33] K. Deng, L. Kong, J. Song, K. Ren, D. Yuan, A weighted k-means clustering based co-scheduling strategy towards efficient execution of scientific workflows in collaborative cloud environments, in: 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, 2011, pp. 547–554, http://dx.doi.org/10.1109/DASC.2011.102.
[34] F. Ma, Y. Yang, T. Li, A data placement method based on Bayesian network for data-intensive scientific workflows, in: 2012 International Conference on Computer Science and Service System, 2012, pp. 1811–1814.
[35] L. Teylo, U. de Paula, Y. Frota, D. de Oliveira, L.M. Drummond, A hybrid evolutionary algorithm for task scheduling and data assignment of data-intensive scientific workflows on clouds, Future Gener. Comput. Syst. 76 (2017) 1–17.
[36] E.I. Djebbar, G. Belalem, Optimization of tasks scheduling by an efficacy data placement and replication in cloud computing, in: R. Aversa, J. Koodziej, J. Zhang, F. Amato, G. Fortino (Eds.), Algorithms and Architectures for Parallel Processing: 13th International Conference, ICA3PP 2013, Vietri Sul Mare, Italy, December 18-20, 2013, Proceedings, Part II, Springer International Publishing, Cham, 2013, pp. 22–29.
[37] D. Yuan, Y. Yang, X. Liu, J. Chen, On-demand minimum cost benchmarking for intermediate dataset storage in scientific cloud workflow systems, J. Parallel Distrib. Comput. 71 (2) (2011) 316–332, http://dx.doi.org/10.1016/j.jpdc.2010.09.003.
[38] Z. Liu, T. Xiang, B. Lin, X. Ye, H. Wang, Y. Zhang, X. Chen, A data placement strategy for scientific workflow in hybrid cloud, in: 2018 IEEE 11th International Conference on Cloud Computing, CLOUD, 2018, pp. 556–563.
[39] H. Yuan, J. Bi, M. Zhou, Multiqueue scheduling of heterogeneous tasks with bounded response time in hybrid green IaaS clouds, IEEE Trans. Ind. Inf. 15 (10) (2019) 5404–5412.
[40] F.J. Clemente-Castelló, B. Nicolae, R. Mayo, J.C. Fernández, Performance model of mapreduce iterative applications for hybrid cloud bursting, IEEE Trans. Parallel Distrib. Syst. 29 (8) (2018) 1794–1807.
[41] M. Mechtri, M. Hadji, D. Zeghlache, Exact and heuristic resource mapping algorithms for distributed and hybrid clouds, IEEE Trans. Cloud Comput. 5 (4) (2015) 681–696.
[42] A. Rezaeian, H. Abrishami, S. Abrishami, M. Naghibzadeh, A budget constrained scheduling algorithm for hybrid cloud computing systems under data privacy, in: Cloud Engineering, IC2E, 2016 IEEE International Conference on, IEEE, 2016, pp. 230–231.
[43] M. Malawski, K. Figiela, M. Bubak, E. Deelman, J. Nabrzyski, Cost optimization of execution of multi-level deadline-constrained scientific workflows on clouds, in: R. Wyrzykowski, J. Dongarra, K. Karczewski, J. Waniewski (Eds.), Parallel Processing and Applied Mathematics: 10th International Conference, PPAM 2013, Warsaw, Poland, September 8-11, 2013, Revised Selected Papers, Part I, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 251–260.
[44] X. Liu, A. Datta, Towards intelligent data placement for scientific workflows in collaborative cloud environment, in: 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum, 2011, pp. 1052–1061.
[45] M.R.H. Farahabady, Y.C. Lee, A.Y. Zomaya, Pareto-Optimal cloud bursting, IEEE Trans. Parallel Distrib. Syst. 25 (10) (2014) 2670–2682.
[46] H.Y. Chu, Y. Simmhan, Cost-efficient and resilient job life-cycle management on hybrid clouds, in: 2014 IEEE 28th International Parallel and Distributed Processing Symposium, 2014, pp. 327–336.
[47] S. Wu, B. Li, X. Wang, H. Jin, Hybridscaler: Handling bursting workload for multi-tier web applications in cloud, in: Parallel and Distributed Computing, ISPDC, 2016 15th International Symposium on, IEEE, 2016, pp. 141–148.
[48] R.L. Cunha, E.R. Rodrigues, L.P. Tizzei, M.A. Netto, Job placement advisor based on turnaround predictions for HPC hybrid clouds, Future Gener. Comput. Syst. 67 (2017) 35–46.
[49] O.-C. Marcu, C. Negru, F. Pop, Dynamic scheduling in real time with budget constraints in hybrid clouds, in: International Conference on Grid Economics and Business Models, Springer, 2015, pp. 18–31.
[50] A. Zinnen, T. Engel, Deadline constrained scheduling in hybrid clouds with Gaussian processes, in: High Performance Computing and Simulation, HPCS, 2011 International Conference on, IEEE, 2011, pp. 294–300.
[51] B. Wang, Y. Song, Y. Sun, J. Liu, Managing deadline-constrained bag-of-tasks jobs on hybrid clouds, in: Proceedings of the 24th High Performance Computing Symposium, Society for Computer Simulation International, 2016, pp. 1–8.
[52] H. Yuan, J. Bi, W. Tan, B.H. Li, Temporal task scheduling with constrained service delay for profit maximization in hybrid clouds, IEEE Trans. Autom. Sci. Eng. 14 (1) (2017) 337–348.
[53] M.R. Hoseinyfarahabady, H.R.D. Samani, L.M. Leslie, Y.C. Lee, A.Y. Zomaya, Handling uncertainty: Pareto-efficient BoT scheduling on hybrid clouds, in: 2013 42nd International Conference on Parallel Processing, 2013, pp. 419–428.
[54] Y.C. Lee, B. Lian, Cloud bursting scheduler for cost efficiency, in: 2017 IEEE 10th International Conference on Cloud Computing, CLOUD, 2017, pp. 774–777, http://dx.doi.org/10.1109/CLOUD.2017.112.
[55] L.F. Bittencourt, C.R. Senna, E.R. Madeira, Scheduling service workflows for cost optimization in hybrid clouds, in: Network and Service Management, CNSM, 2010 International Conference on, IEEE, 2010, pp. 394–397.
[56] W. Al-Hassan, M. Fayek, S. Shaheen, Psosa: An optimized particle swarm technique for solving the urban planning problem, in: 2006 International Conference on Computer Engineering and Systems, IEEE, 2006, pp. 401–405.
[57] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, K. Vahi, Characterizing and profiling scientific workflows, Future Gener. Comput. Syst. 29 (3) (2013) 682–692, Special Section: Recent Developments in High Performance Computing and Security.
[58] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, R. Buyya, Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Softw. Pract. Exper. 41 (1) (2011) 23–50.
[59] IBM Knowledge Center, 2019, https://www.ibm.com/support/knowledgecenter/en/SSBS6K_3.1.2/installing/plan_capacity.html, IBM.
[60] Microsoft azure pricing, 2019, https://azure.microsoft.com/, Microsoft Inc..
[61] Google compute engine, 2019, https://cloud.google.com/compute/, Google Inc..
[62] J. Varia, The total cost of (non) ownership of web applications in the cloud, 2012, pp. 1–30.

Amirmohammad Pasdar is a Ph.D. student in the Department of Computing at Macquarie University. His research interests center around scheduling in cloud computing and the usage of machine learning in parallel and distributed systems.

Young Choon Lee is a senior lecturer at the Department of Computing, Macquarie University, Sydney, Australia. He received his Bachelor of Science (with honour) and Ph.D. from The University of Sydney, Australia, 2004 and 2008, respectively. His research interests include distributed systems and high performance computing.

Khaled Almi'ani is currently a member of the Computer Information System Department, Higher College of Technology, United Arab Emiratis. His research interests lie in algorithms for distributed systems, network optimization, and transportation network modeling. He received his Ph.D. in Information Technology from the University of Sydney in 2010.