A biobjective model for resource provisioning in multi-cloud environments with capacity constraints

Luce Brotcorne, Joaquín Ezpeleta, Carmen Galé

Operational Research (2023) 23:31
https://doi.org/10.1007/s12351-023-00773-x

ORIGINAL PAPER

A biobjective model for resource provisioning in multi-cloud environments with capacity constraints

Luce Brotcorne[1] · **Joaquín Ezpeleta**[2] · **Carmen Galé**[3]

Received: 5 May 2022 / Revised: 8 February 2023 / Accepted: 29 March 2023 /
Published online: 3 May 2023

Abstract
Private and public clouds are good means for getting on-demand intensive computing resources. In such a context, selecting the most appropriate clouds and virtual machines (VMs) is a complex task. From the user's point of view, the challenge consists in efficiently managing cloud resources while integrating prices and performance criteria. This paper focuses on the problem of selecting the appropriate clouds and VMs to run bags-of-tasks (BoT): big sets of identical and independent tasks. More precisely, we define new mathematical optimization models to deal with the time of use of each VMs and to jointly integrate the execution makespan and the cost into the objective function through a bi-objective problem. In order to provide trade-off solutions to the problem, we propose a lexicographic approach. In addition, we introduce, in two different ways, capacity constraints or bounds on the number of VMs available in the clouds. A global limit on the number of VMs or resource constraints at each time period can be defined. Computational experiments are performed on a synthetic dataset. Sensitivity analysis highlights the effect of the resource limits on the minimum makespan, the effect of the deadline in the total operation cost, the impact of considering instantaneous capacity constraints instead of a global limit and the trade-off between the cost and the execution makespan.

Keywords Cloud computing · Mathematical programming · Bag of tasks · Multi cloud

1 Introduction

Currently cloud computing is considered as the step after clusters or grids for massive and intensive computation in parallel or distributed environments. More precisely, according to Buyya et al. (2009) a cloud can be viewed as a distributed

✉ Carmen Galé
cgale@unizar.es

Extended author information available on the last page of the article

collection of inter-connected virtual computers that can be provisioned by users in a dynamic way. Even if clouds and grids share some characteristics, clouds have some specificities as pointed by Foster et al. (2008): (i) they are 'massively scalable", (ii) they are "abstract entities" able to deliver different levels of services, (iii) they are driven by "economies of scale" and (iv) cloud services can be configured and delivered dynamically on demand.

Public cloud providers offer different types of instances of virtual machines (VMs) which can be provisioned to meet the computational demands of users. The prices vary with the type of hardware, the amount of memory required, the operating system, the storage capacity, the availability zone (the geographical area where the services physically stay), the bandwidth, etc.

Cloud providers offer different types of computing power services in a Platform-as-a-Service (PaaS) market and different modalities for hiring VMs. In the "on-demand" modality, instances can be hired or released when needed. In the "reserved" modality users pre-pay for long-term reservations to insure lower tariffs and a better availability. In most cases, there is a maximum number of instances that can be used at a given moment. There is also an opportunistic market in which providers offer instances when they have some excess of capacity (unused resources), such as the Spot instances of Amazon web services or Preemptible VM of Google cloud. And more recently, also "burstable" instances could be hired. A burstable instance provides a baseline level of performance, but that can burst to a higher level for occasional spikes in usage.

From the cloud user's point of view, the challenge consists in efficiently managing cloud resources while integrating prices and performance criteria. Díaz et al. (2017) solve an allocation problem with reserved VMs and on-demand VMs that takes into account the predicted load of each time slot to minimize the cost while a required level of performance is guaranteed for the service. The load is expressed in a "request per second" metric and it determines firstly the optimal allocation of reserved VMs followed by the number of on-demand VMs if necessary, i.e. the later decision is made for only one time slot depending on its load. Genez et al. (2020) define an integer linear programming model to determine the VMs to be leased from multiple cloud providers to execute workflows while meeting the user's deadline and minimizing the leasing costs. The loads are scientific workflows decomposed into smaller tasks to be processed in a defined order. In the proposed model only one VM can execute each processing demand.

The focus of this work is related to another very popular type of workload named a Bag-of-Task (BoT). It consists in a set of identical independent tasks that can be operated in parallel. The execution of a BoT usually consumes a large amount of computational time as well as data storage and transfer. Determining the allocation of VM instances to tasks is a scheduling problem which is NP-hard (Abdi et al. 2018; Abed-alguni and Alawad 2021). The goal of the paper is to solve a BoT scheduling problem in a multi-cloud environment.

The primary motivation of this work is to develop an integrated model to consider both cost and makespan criteria in the provisioning and scheduling procedure of BoT applications in multi-cloud environments with capacity constraints using on-demand instances. For this purpose, the contributions of the paper are the following:

– We define new optimization models integrating three type of decisions: instance type selection, resource scaling and workload allocation.
– We introduce two optimization criteria: the cost and the execution makespan. A lexicographical approach is used to deal with both criteria.
– To the difference of previous works by Abdi et al. (2017) and Abdi et al. (2018), tasks of the same BoT are allowed to run on different instance types and the requirement for an instance to be hired from the beginning to the deadline is relaxed.
– We propose and compare two approaches to introduce capacity constraints into the model.

The paper is organized as follows. Section 2 presents related work. In Sect. 3, we first introduce a new integer programming model to schedule BoTs in a multi-cloud environment in order to minimize the cost of executing the whole set of tasks while satisfying a specified deadline. Next, a new criteria related to the minimization of the makespan is introduced into the objective function. In Sect. 4, we extend the models defined in Sect. 3 to integrate instantaneous capacity constraints and present a comprehensive study of its properties. In Sect. 5, we put into highlight through computational experiments the performance of the proposed models and, in particular, the impact of including instantaneous capacity constraints. Section 6 is devoted to conclusions and future research proposals.

2 Related work

The scheduling of general tasks in (multi) cloud environments has been deeply studied (Alkhanak et al. 2015; Mann 2015; Keivani and Tapamo 2019; Kumar et al. 2019. In a survey and taxonomy of resource optimization for BoTs, (Thai et al. 2018) put into highlight three relevant characteristics to be taken into account: the type of instances to hire, the time during which each instance must work and the workload to be allocated to each hired instance. The special characteristics of BoT problems has made it possible to develop solutions specifically adapted to this field. We next review some recent papers involving a mathematical model. All these articles and the references they contain provide an update on this subject that is currently attracting so much interest.

Wang et al. (2016) present a heuristic algorithm for solving a binary non-linear program whose goal is to minimize the cost of renting VMs while respecting the respective deadlines. Abdi et al. (2017) and Abdi et al. (2018) provide a binary linear programming formulation for the scheduling of BoT applications with a fixed deadline. The model consists in allocating an instance type of one of the clouds to each BoT of the submitted applications. The restrictive assumption that only one instance type can be used allows them to compute the coefficients of the mathematical model. More precisely, the authors determine the number of required VMs to complete the BoT for each instance type. For this purpose, the so-called master VMs are run until the deadline is met. If necessary, the required time of running an extra VM is added to execute remaining tasks. Wang et al. (2019) solve the optimization

problem maximizing the global utilization of the servers by using a scheduling method with different degrees of task parallelism. Each task is characterized by a deadline requirement and its parallel degree. More precisely, the number of cores executing the task at a time, is tuned between one and its maximum value according to the available cores of the server during its execution. The authors define a nonlinear program solved by heuristic methods.

Teylo et al. (2021) propose a framework for the execution of BoT applications with deadline constraints including, both spot and on-demand burstable VMs, aiming at minimizing the monetary cost and the execution time. The objective function is a weighted function of both terms. In order to ensure the deadline is respected, the framework considers the possibility of migrating tasks in the hibernated spot instances and those not yet executed to on-demand instances. They use an iterated local search to obtain an approximate solution at an acceptable computing time of the non linear model.

Karaja (2022) focus on solving the BoT scheduling problem in a dynamic multi-cloud environment. They rely on a dynamic batch approach: tasks to be scheduled are collected and scheduled in batches at predefined times. They propose a bi-level multi-follower model. The upper level represents the multi-cloud scheduler that aims to minimize the completion time of the BoT while taking into consideration the budget constraints. The lower level aims to minimize simultaneously the execution cost and the completion time of tasks belonging to the BoT. Metaheuristics are defined to efficiently solve the problem.

Chhabra et al. (2022) also adopt a solution based on a heuristic approach. The problem of scheduling concurrent BoTs is considered as a biobjective optimization problem to minimize the makespan and energy consumption. The biobjective problem is reformulated as a single objective one and a heuristic algorithm is defined on the basis of the fitness function. More precisely, they define the Whale Particle Swarm Optimization algorithm integrating the search mechanism of opposition-based learning and particle swarm optimization mechanisms together with the standard whale optimization algorithm.

Yin et al. (2022) focus on the uncertainty around the real time required to execute a BoT task and the performance of any given VM. The time required can vary depending on many aspects such as the concrete data to be processed. In addition, due to virtualization, the same VM can result in varying execution times for the same task. These authors consider such times as random variables and propose a BoT scheduling problem whose aim is to maximize the profit of the cloud provider while considering the deadline constraints imposed by the user. The problem takes the point of view of a a service cloud provider, and considers the integration of the private cloud resources within public clouds. The stochastic optimization problem is solved by a metaheuristic named Immune Algorithm-based BoT Scheduling (IABS).

All previous works based on mathematical optimization models, linear or non linear, rely on heuristic-based solution approaches. In this work the problem is modeled as a single level biobjective linear mathematical optimization model integrating time dependent decision variables which can be solved by one of the shelf software. Moreover, the biobjective optimization problem is solved by a lexicographic approach leading to the generation of two specific efficient solutions.

3 Mathematical models for scheduling BoT applications

A BoT application is composed of a set of BoTs. In a multi-cloud environment, assigning each application to a single cloud makes sense, as it is assumed that all the BoTs in an aplication share some common data and this single allocation avoids the time and monetary costs of transferring and storing the same data in multiple clouds. BoTs in an application are independent, that is, there are not precedence constraints among them, and all the tasks can be executed in parallel.

In this section we define three models generalizing the integer optimization problem **IP-NDS** defined by Abdi et al. (2017). They pre-process the problem for a given deadline by computing for each instance type the number of VMs needed to run each BoT. The decisions in model **IP-NDS** are thus reduced to deciding which type of instance is selected for each BoT. For completeness, this model is included in the Appendix and notations in Table 1. In this paper, we first relax the assumption that only one type of instance can be allocated to a given BoT

Table 1 Notation used in the mathematical problem formulations

Notation	Description
K	Set of clouds
J_k	Set of VM instances types in cloud $k \in K$
P_{kj}	The cost of using VM instance $j \in J_k$, per time unit.
ccu_{kj}	The performance level of a VM instance type $j \in J_k$
U_k	The maximum number of instances allowed in cloud $k \in K$
U_{kj}	The maximum number of VMs of VM instance type $j \in J_k$
I	Set of applications.
$\mathcal{B}_i$	Set of independent BoTs in application $i \in I$
δ_{ib}	The number of independent tasks in BoT $b \in \mathcal{B}_i$
ET_{ib}	The execution time of task $b \in \mathcal{B}_i$ in a VM with a performance of 1 CCU.
D	The deadline
T	Set of time periods
τ_{ibkj}	The execution time of a task in BoT $b \in \mathcal{B}_i$ on an instance type $j \in J_k$
w^d_{ibkj}	The number of tasks in BoT $b \in \mathcal{B}_i$ executed by an instance type $j \in J_k$ during exactly d time periods
$y_{ik} \in \{0,1\}$	Equals 1 if application i is submitted to cloud k
$N^d_{ibkj} \geq 0$	Integer. Number of VMs of instance type $j \in J_k$ working on BoT $b \in \mathcal{B}_i$ during exactly d time periods.
$z^d \in \{0,1\}$	Equals 1 if any VM is working during d time periods
$\theta \geq 0$	Time period
Additional notation **IP-NDS**	
η_{ibkj}	The maximum number of tasks of BoT $b \in \mathcal{B}_i$ solved by one VM of instance type $j \in J_k$ for a deadline D
ν_{ibkj}	The number of required master VMs of instance type $j \in J_k$ for executing tasks of BoT $b \in \mathcal{B}_i$
T_{ibkj}	The running time of the master VMs of instance type $j \in J_k$ executing BoT $b \in \mathcal{B}_i$

imposed in Abdi et al. (2017) to formulate the allocation problem. Relaxing this assumption allows to explore all the feasible solutions involving one or more instance types of one cloud, while **IP-NDS** model only computes a subset of feasible solutions. The second extension consists in considering a makespan criteria into the objective function in place of considering a pre-established deadline D. Finally the third one integrates simultaneously a cost minimization and makespan criteria into the objective function.

As a general rule, in cloud environments, the cost depends on the type of CPU, the amount of memory required, the storage capacity used for data management, etc. Usually, CPU and memory are considered as a whole by the cloud providers in the set of the proposed VM instances. In the following, we use the classical CCU metric developed by CloudHarmony (as mentionned in Abdi et al. 2017) to evaluate the performance of instance types belonging to a variety of clouds rented for executing different tasks. A value of 1 CCU indicates a CPU capacity of 1.0–1.2 GHz Opteron or 2007 Xeon processor.

Table 1 shows the notations used in the mathematical formulation of the models.

Let K be the set of clouds. Each cloud $k \in K$ provides a set J_k of VM instance types which are characterized by the fee P_{kj} of running per time unit, and the performance level, ccu_{kj}, $k \in K$, $j \in J_k$. We assume a cloud provider-level restriction scenario, i.e there exist some capacity constraints. More precisely, resources are allocated in accordance to the limits defined by providers. For each cloud $k \in K$, U_k represents the maximal number of instances allowed by the cloud, while U_{kj} is the corresponding limit for each instance type $j \in J_k$.

On the other hand, let I be a set of applications. Each application $i \in I$ consists of a set of independent BoTs $\mathcal{B}_i$. A BoT $b \in \mathcal{B}_i$ includes δ_{ib} independent tasks and one task requires ET_{ib} execution time in a VM with a performance of 1 CCU. Therefore, the execution time of each task of BoT $b \in \mathcal{B}_i$ on an instance type $j \in J_k$, $k \in K$ is computed as:

$$\tau_{ibkj} = \frac{ET_{ib}}{ccu_{kj}}$$

Given a deadline D, we define the set $T = \{1, 2, \cdots, D\}$. Even if Abdi et al. (2017) consider an hour as the time unit, the models are valid for any time unit, such as minutes or seconds. A higher resolution in the time unit increases the cardinality of T. Without loss of generality we next use the term time unit. The number of tasks of BoT $b \in \mathcal{B}_i$, $i \in I$ executed by an instance type $j \in J_k$, $k \in K$ during exactly $d \in T$ time units is computed as follows:

$$w^d_{ibkj} = \left\lfloor \frac{d}{\tau_{ibkj}} \right\rfloor$$

The decision variables are:

$$y_{ik} = \begin{cases} 1 \text{ if application } i \text{ is submitted to cloud } k, \\ 0 \text{ otherwise,} \end{cases} \quad i \in I,\ k \in K$$

$N^d_{ibkj} =$ Number of VMs of instance type j in cloud k working on BoT b during exactly d time units, $k \in K$, $j \in J_k$, $i \in I$, $b \in \mathcal{B}_i$, $d \in T$

The Multi-Cloud Bot Scheduling (**MCBS**) problem for a fixed deadline D is formulated as follows:

$$\min_{y,N} \sum_{k\in K} \sum_{j\in J_k} \sum_{i\in I} \sum_{b\in\mathcal{B}_i} \sum_{d\in T} (P_{kj} \times d) N^d_{ibkj} \tag{1a}$$

$$\text{s.t.:} \sum_{k\in K} y_{ik} = 1, \quad i \in I \tag{1b}$$

$$N^d_{ibkj} \le U_{kj} y_{ik}, \quad i \in I,\ b \in \mathcal{B}_i,\ k \in K,\ j \in J_k,\ d \in T \tag{1c}$$

$$\sum_{i\in I} \sum_{b\in\mathcal{B}_i} \sum_{j\in J_k} \sum_{d\in T} N^d_{ibkj} \le U_k, \quad k \in K \tag{1d}$$

$$\sum_{i\in I} \sum_{b\in\mathcal{B}_i} \sum_{d\in T} N^d_{ibkj} \le U_{kj}, \quad k \in K,\ j \in J_k \tag{1e}$$

$$\sum_{k\in K} \sum_{j\in J_k} \sum_{d\in T} w^d_{ibkj} N^d_{ibkj} \ge \delta_{ib}, \quad i \in I,\ b \in \mathcal{B}_i \tag{1f}$$

$$y_{ik} \in \{0, 1\}, \quad i \in I,\ b \in \mathcal{B}_i,\ k \in K,\ j \in J_k,\ d \in T \tag{1g}$$

$$N^d_{ibkj} \ge 0, \ \text{integer}, \quad k \in K,\ j \in J_k,\ d \in T \tag{1h}$$

The objective function (1a) consists in minimizing the cost. Constraints (1b) guarantee that each application is submitted to only one cloud in the federation. If an application is not allocated to a cloud, none of the instances belonging to that cloud are chosen to run tasks from that application by constraints (1c). Constraints (1d) and (1e) are cloud and type capacity constraints. Constraints (1f) guarantee that the whole set of tasks are executed.

The optimal value of model **MCBS** is lower or equal to the one of model **IP-NDS** since the feasible set of model **IP-NDS** is included in the feasible region of model **MCBS**. Indeed the optimal solution of **IP-NDS** is always feasible for **MCBS**. Moreover as shown in Example 1 at the end of this section, the inequality can be strict, which means that model **MCBS** can provide a better scheduling than model **IP-NDS**.

Starting from **MCBS** we can define a model **MCBS-M** including a makespan criteria in place of a fixed deadline. The makespan is the length of time needed for executing all BoTs. In order to compute it, a binary variable, z^d, is added for each $d \in T$ to determine whether there is an instance of any type running at that time d. In that case, the variable is equal to 1:

$$z^d = \begin{cases} 1 \text{ if any } N^d_{ibkj} > 0,\ k \in K,\ j \in J_k,\ i \in I,\ b \in \mathcal{B}_i, \\ 0 \text{ otherwise} \end{cases}$$

Then, the objective function of model **MCBS-M** consists of minimizing the maximum value of $d \in T$ in which some instance type j is used, $j \in J_k$, $k \in K$:

$$\min_{y,z,N} \max_{d \in T} dz^d \tag{2}$$

In order to avoid the non-linearity of the objective function (2), we introduce a continuous decision variable θ and we include a set of constraints in model **MCBS-M** to guarantee that θ is greater than or equal to dz^d, for all $d \in T$. Therefore, the model **MCBS-M** is linear and it is formulated as follows:

$$\textbf{MCBS-M}: \min_{y,z,N,\theta} \theta \tag{3a}$$

$$\text{s.t.:} \sum_{k \in K} y_{ik} = 1, \quad i \in I \tag{3b}$$

$$N^d_{ibkj} \leq U_{kj} y_{ik}, \quad i \in I,\ b \in \mathcal{B}_i,\ k \in K,\ j \in J_k,\ d \in T \tag{3c}$$

$$\sum_{i \in I} \sum_{b \in \mathcal{B}_i} \sum_{j \in J_k} \sum_{d \in T} N^d_{ibkj} \leq U_k, \quad k \in K \tag{3d}$$

$$\sum_{i \in I} \sum_{b \in \mathcal{B}_i} \sum_{d \in T} N^d_{ibkj} \leq U_{kj}, \quad k \in K,\ j \in J_k \tag{3e}$$

$$\sum_{k \in K} \sum_{j \in J_k} \sum_{d \in T} w^d_{ibkj} N^d_{ibkj} \geq \delta_{ib}, \quad i \in I,\ b \in \mathcal{B}_i \tag{3f}$$

$$N^d_{ibkj} \leq U_{kj} z^d, \quad i \in I,\ b \in \mathcal{B}_i,\ k \in K,\ j \in J_k,\ d \in T \tag{3g}$$

$$\theta \geq dz^d, \quad d \in T \tag{3h}$$

$$y_{ik} \in \{0, 1\}, \quad i \in I, k \in K \tag{3i}$$

$$N^d_{ibkj} \geq 0, \text{ integer} \quad i \in I, b \in \mathcal{B}_i, k \in K, j \in J_k, d \in T \tag{3j}$$

$$\theta \geq 0 \tag{3k}$$

Note that if there exists a variable $N^d_{ibkj} > 0$ then at least one VM is active during d time units. The corresponding constraint (3g) guarantees that $z^d = 1$ and by constraint (3h) $\theta \geq d$. It results that the makespan will be more than d time units provided that there are VMs working d time units. The minimum makespan, denoted by θ^*, is thus implicitly computed. Constraint (3k) is a sign constraint on the decision variable θ. Constraints (3b–3f) and (3i–3j) are the same as in **MCBS** problem.

Finally, both cost and deadline minimization can be addressed simultaneously from a biobjective perspective. In a biobjective problem, it is generally not possible to compute a single solution generating the best value for both objectives. In this context, reducing the makespan leads to an increase in cost. A solution is efficient for a biobjective problem if there is no other feasible solution which a lower value for both objective functions, one of which is strictly lower. The set of efficient solutions provides different trade-offs between cost and makespan. In particular, a lexicographic approach can be used to compute an efficient solution (Ehrgott 2005). The lexicographic approach is a nonscalarizing method for finding efficient solutions based on the ordering of the objectives according to some priorities. Model **MCBS-1** prioritizes cost over makespan and the **MCBS-2** model takes the reverse order. Thus, this approach allows either i) to minimize the makespan in which the tasks are executed at minimum cost with a given deadline D:

$$\textbf{MCBS-1} : Lex \min_{y,z,N,\theta} \left(\sum_{k\in K} \sum_{j\in J_k} \sum_{i\in I} \sum_{b\in\mathcal{B}_i} \sum_{d\in T} (P_{kj} \times d) N^d_{ibkj},\ \theta \right),$$

or ii) to minimize the cost at the lowest feasible makespan:

$$\textbf{MCBS-2} : Lex \min_{y,z,N,\theta} \left(\theta,\ \sum_{k\in K} \sum_{j\in J_k} \sum_{i\in I} \sum_{b\in\mathcal{B}_i} \sum_{d\in T} (P_{kj} \times d) N^d_{ibkj} \right).$$

In the next example, we compare the solutions provided by the models **MCBS** and **MCBS-M** to the solution resulting from the **IP-NDS** model.

Example 1 Table 2 displays the input data with one cloud ($|K| = 1$), three instance types ($|J_1| = 3$) and one application ($|I| = 1$) with two BoTs ($|\mathcal{B}_1| = 2$). Let VM_j denote a VM of instance type j.

Auxiliary parameters are required to formulate **IP-NDS** model: η_{bj} is the maximum number of tasks of BoT $b \in \mathcal{B}_1$ solved by one VM_j, $j \in J_1$ for a given deadline D; although a master VM is run until the deadline is met, T_{bj} is the exact running time ot the master VMs and ν_{bj} is the number of required master VMs of instance type VM_j for executing tasks of BoT $b \in \mathcal{B}_1$. If $\tau_{bj}\eta_{bj} < \delta_b$ the master VM_js cannot carry out the computation of all tasks of BoT $b \in \mathcal{B}_1$. In this case, one extra VM_j is

Table 2 Instance types specifications and BoTs of the Example 1

One cloud with limit $U = 20$				BoTs	Number of tasks (δ_b)	ET_b
Instance type	Price(\$/ time unit)	CCU	Limit			
VM_1	0.6	10	10	BoT_1	600	0.9
VM_2	0.4	6	10	BoT_2	100	1.2
VM_3	0.2	2	10			

required during TR_{bj} in time units. These auxiliary paramaters are mathematically defined in the Appendix. For the example the corresponding values are defined in Table 3. The first column gives the name of the instance; columns 2 to 7 (resp. 8 to 13) present the corresponding data to BoT_1 (resp. BoT_2).

The optimal solution of **IP-NDS** is displayed in bold in Table 3. The minimum cost is equal to 40.8\$. Both BoTs are executed by VM_1: 5 master and 1 extra (5 time units) for BoT_1 and 1 master and 1 extra (3 time units) for BoT_2.

Table 4 shows the value of the auxiliary parameters $\{w_{bj}^d\}$ involved in problem **MCBS**. For each number of time units shown at the first row, the number of tasks executed by each VM is computed. As an example, a VM_1 instance executes during 5 time units 41 tasks of BoT_2.

Table 3 Auxiliary parameters in the **IP-NDS** model for a deadline $D = 10$ and the optimal solution with cost=40.8\$

Instance	BoT_1						BoT_2					
	τ_{1j}	η_{1j}	ν_{1j}	T_{1j}	TR_{1j}	Cost	τ_{2j}	η_{2j}	ν_{2j}	T_{2j}	TR_{2j}	Cost
VM_1	0.09	111	5	10	5	**33**	0.12	83	1	10	3	**7.8**
VM_2	0.15	66	9	10	1	36.4	0.2	50	2	10	0	8
VM_3	0.45	22	27	10	3	54.6	0.6	16	6	10	3	12.6

Table 4 Auxiliary parameters of the **MCBS** model for Example 1

	BoT_1									
	w_{1j}^1	w_{1j}^2	w_{1j}^3	w_{1j}^4	w_{1j}^5	w_{1j}^6	w_{1j}^7	w_{1j}^8	w_{1j}^9	w_{1j}^{10}
VM_1	11	22	33	44	55	66	77	88	100	111
VM_2	6	13	20	26	33	40	46	53	60	66
VM_3	2	4	6	8	11	13	15	17	20	22
	BoT_2									
	w_{2j}^1	w_{2j}^2	w_{2j}^3	w_{2j}^4	w_{2j}^5	w_{2j}^6	w_{2j}^7	w_{2j}^8	w_{2j}^9	w_{2j}^{10}
VM_1	8	16	25	33	41	50	58	66	75	83
VM_2	5	10	15	20	25	30	35	40	45	50
VM_3	1	3	5	6	8	10	11	13	15	16

In the solution proposed by **MCBS** model, there is an intensive use of VM_1 instances for both, BoT_1 and BoT_2: 6 instances executing tasks for BoT_1 during 9 time units, solving 600 tasks with a cost of 32.4\$; 1 instance executing BoT_2 tasks for 9 time units and another one for 3 time units, solving 75 and 25 tasks at a cost of 5.4\$ and 1.8\$, respectively. The total cost is 39.6\$ which is lower than 40.8\$. Moreover, the makespan is 9 time units, one time unit less than the deadline $D = 10$ assumed in **IP-NDS** model.

Let us define upper bounds on the number of VMs of each instance type: $U_1 = U_2 = 4$ and $U_3 = 2$. With these constraints, **IP-NDS** problem is no feasible any more, although **MCBS** problem finds a solution for a deadline $D = 10$. The optimal cost is 42.2\$. Tasks of BoT_1 are executed by 4 VM_1 instances during 10 time units (444 tasks), 2 VM_2 during 10 time units (132 tasks) and 1 VM_3 during 10 time units (22 tasks) and another one during 1 time unit (2 tasks); BoT_2 is executed by 2 VM_2 during 10 time units (100 tasks). The constraints on the number of resources are satisfied since 4 VM_1, 4 VM_2 and 2 VM_3 have been rented.

Finally by solving model **MCBS**, the smallest feasible deadline within which the related assignment problem is feasible given the upper bounds: $U_1 = U_2 = U_3 = 10$ and $U = 20$ is $D = 5$ with a cost of 42.8\$. For level constraints of the provider more restrictive, $U_1 = U_2 = 4$ and $U_3 = 2$, the smallest deadline threshold is 10 time units. If the constraints on the number of resources are $U_1 = 4$, $U_2 = 7$ and $U_3 = 4$ and the deadline is $D = 10$, model **IP-NDS** is not feasible any more and the optimal solutions for the models defined in this section are displayed in Table 5. Note that the lexicographic approach provides two efficient points with both objective function values, the cost and the makespan, equal to (41.6, 9) and (43, 8) by solving models **MCBS-1** and **MCBS-2**, respectively.

4 Global and instantaneous capacity constraints

From a practical point of view cloud providers limit the number of VMs available to a user at a given time even if from a theoretical point of view clouds could be able to provision an unlimited number of resources. Of course defining bounds on the number of resources strongly impacts the total cost and the feasible makespan since it directly affects the set of feasible allocations. The limits can be defined globally or instantaneously.

For example a global limit on the number of instances, independently of their types (50 instances per cloud service) is required by cloud Microsoft Azure. In the case of Amazon EC2, until September 2019 there were a limit of 20 VMs. Since October 2019, EC2 defines vCPU-based (virtual CPU) instance limits. Each instance has a (different) number of vCPUs, depending on its type. The total number of instances depends on their types, and any combination is allowed while the total number of vCPUs does not exceed the limit. This is also the case of cloud Google Cloud (GC). The bound on the number of CPUs allowed refers to the total number of virtual CPUs in all VM instances in a region. Although global capacity constraints are common in the literature, the attention has been focused more recently on instantaneous bounds limiting the number of resources that can be used at a given time. Capacity constraints can be

Table 5 Optimal solutions and optimal values for Example 1 with upper bounds $U_1 = 4$, $U_2 = 7$ and $U_3 = 4$ and deadline $D = 10$

BoT	**MCBS**: Minimum cost= 41.6$					**MCBS-M**: Minimum makespan θ^*=8				
	VMs	N	d	Tasks	Cost($)	VMs	N	d	Tasks	Cost($)
1	1	4	10	440	24	1	4	8	352	19.2
	2	2	9	120	7.2	2	4	8	212	12.8
	2	1	6	40	2.4	3	4	8	68	6.4
2	2	2	9	90	7.2	2	3	8	120	9.6
	2	1	2	10	0.8					
Total cost 41.6$						Total cost: 48$				
	MCBS-1: Minimum makespan θ^*=9					**MCBS-2**: Minimum cost=43$				
BoT	VMs	N	d	Tasks	Cost($)	VMs	N	d	Tasks	Cost($)
1	1	4	9	400	21.6	1	3	8	264	14.4
	2	2	9	120	7.2	2	6	8	318	19.2
	2	2	6	80	4.8	3	1	8	17	1.6
						3	1	1	2	0.2
2	2	2	9	90	7.2	1	1	8	66	4.8
	2	1	2	10	0.8	2	1	7	35	2.8
Total cost 41.6$						Total cost: 43$				

more constraining in the case of using private clouds, which usually consist of a more restrictive set of resources. Also the fact of having a set of previously hired instances that could be used for solving the problem along the whole makespan is a source for constraints.

In this section we extend models **MCBS** and **MCBS-M** to integrate instantaneous resource constraints in place of global resource ones. The decision variable N^d_{ibkj} is replaced by $N^{dt_0}_{ibkj}$, $k \in K$, $j \in J_k$, $i \in I$, $b \in \mathcal{B}_i$, $d \in T$, $t_0 \in T$ to point out the starting time t_0 of the VMs of instance type j in cloud k for execution of tasks of BoT b during exactly d time units. The new model **MCBS-ins** formulation is:

$$\min_{y,N} \sum_{k\in K} \sum_{j\in J_k} \sum_{i\in I} \sum_{b\in\mathcal{B}_i} \sum_{d,\, t_0\in T} (P_{kj} \times d) N^{dt_0}_{ibkj} \tag{4a}$$

$$\text{s.t.:} \sum_{k\in K} y_{ik} = 1, \quad i \in I \tag{4b}$$

$$N^{dt_0}_{ibkj} \leq U_{kj} y_{ik}, \quad i \in I, B \in \mathcal{B}_i, k \in K, j \in J_k, d, t_0 \in T \tag{4c}$$

$$\sum_{i\in I}\sum_{b\in\mathcal{B}_i}\sum_{j\in J_k}\sum_{t_0=1}^{t}\sum_{d=t+1-t_0}^{D} N_{ibkj}^{dt_0} \le U_k, \quad k\in K,\, t\in T \tag{4d}$$

$$\sum_{i\in I}\sum_{b\in\mathcal{B}_i}\sum_{t_0=1}^{t}\sum_{d=t+1-t_0}^{D} N_{ibkj}^{dt_0} \le U_{kj}, \quad k\in K,\, j\in J_k,\, t\in T \tag{4e}$$

$$\sum_{k\in K}\sum_{j\in J_k}\sum_{d\in T}\sum_{t0=1}^{D-d+1} w_{ibkj}^{d} N_{ibkj}^{dt_0} \ge \delta_{ib}, \quad i\in I,\, b\in\mathcal{B}_i \tag{4f}$$

$$y_{ik}\in\{0,1\} \quad i\in I,\, k\in K \tag{4g}$$

$$N_{ibkj}^{dt_0}\ge 0 \text{ integer} \quad i\in I,\, b\in\mathcal{B}_i,\, k\in K,\, j\in J_k,\, d,t_0\in T \tag{4h}$$

Several constraints from **MCBS** have been adapted. Constraints (4d) and (4e) ensure that for each time period $t\in T$, the number of active VMs is lower or equal to the corresponding upper bound. Note that a VM started at time t_0 for d time units is running at time t if and only if $t_0+d-1\ge t$. An instance $j\in J_k$, $k\in K$ executes w_{ibkj}^{d} tasks of BoT $b\in\mathcal{B}_i$ if and only if $t_0+d-1\le D$. Hence the index t_0 of the last sum in constraints (4f) goes from 1 to $D-d+1$. Since an instance running beyond D does not make sense, we can add the following set of constraints:

$$N_{ibkj}^{dt_0}=0,\ i\in I,\, B\in\mathcal{B}_i,\, k\in K,\, j\in J_k,\, d\in T, t_0\ge D-d+2 \tag{4i}$$

Model **MCBS-ins** is less restrictive than **MCBS** and can result in an optimal solution with lower cost since the upper bound is applied to each time period.

Let $v(\mathbf{M})$ denote the optimal value of a model $\mathbf{M}$.

Proposition 4.1 $v(\mathbf{MCBS\text{-}ins})\le v(\mathbf{MCBS})$ *and the inequality can be strict.*

Proof Let $(\overline{y},\overline{N})$ be an optimal solution of model **MCBS**. We define:

$$\widehat{y}_{ik}=\overline{y}_{ik}, \qquad i\in I,\, k\in K$$

$$\widehat{N}_{ibkj}^{dt_0}=\begin{cases}\overline{N}_{ibkj}^{d} & \text{if } t_0=1\\ 0 & \text{otherwise}\end{cases} \qquad i\in I,\, b\in\mathcal{B}_i,\, k\in K,\, j\in J_k,\, d,t_0\in T$$

It easy to see that $\widehat{y}$ satisfies constraints (4b) and (4g). Since $\overline{N}$ meets constraints (1c) and (1h), then $\widehat{N}$ satisfies constraints (4c) and (4h). Moreover, for each $k\in K$ and $t\in T$:

$$\sum_{i\in I}\sum_{b\in\mathcal{B}_i}\sum_{j\in J_k}\sum_{t_0=1}^{t}\sum_{d=t+1-t_0}^{D}\widehat{N}_{ibkj}^{dt_0}=\sum_{i\in I}\sum_{b\in\mathcal{B}_i}\sum_{j\in J_k}\sum_{d=t}^{D}\overline{N}_{ibkj}^{d}\leq\sum_{i\in I}\sum_{b\in\mathcal{B}_i}\sum_{j\in J_k}\sum_{d=1}^{D}\overline{N}_{ibkj}^{d}\leq U_k$$

on the basis of constraints (1d). Therefore, constraints (4d) are satisfied. In a similar way, constraints (1e) ensure that $\widehat{N}$ satisfies constraints (4e). Constraint (4f) for $i \in I$ and $b \in \mathcal{B}_i$:

$$\sum_{k\in K}\sum_{j\in J_k}\sum_{d\in T}\sum_{t0=1}^{D-d+1}w_{ibkj}^{d}\widehat{N}_{ibkj}^{dt_0}=\sum_{k\in K}\sum_{j\in J_k}\sum_{d\in T}w_{ibkj}^{d}\overline{N}_{ibkj}^{d}\geq\delta_{ib}$$

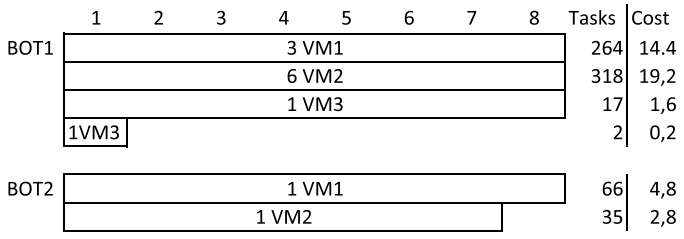
where the constraints (1f) have been applied in the last inequality.

In conclusion, any feasible solution for **MCBS** provides a feasible solution for model **MCBS-ins**. Moreover, since the objective function (4a) only depends on the duration while the machine is working on some BoT:

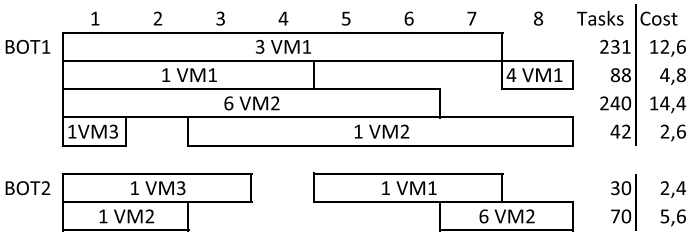
$$\sum_{k\in K}\sum_{j\in J_k}\sum_{i\in I}\sum_{b\in\mathcal{B}_i}\sum_{d,t_0\in T}(P_{kj}\cdot d)\widehat{N}_{ibkj}^{dt_0}=\sum_{k\in K}\sum_{j\in J_k}\sum_{i\in I}\sum_{b\in\mathcal{B}_i}\sum_{d\in T}(P_{kj}\cdot d)\overline{N}_{ibkj}^{d}$$

Hence, v(**MCBS-ins**) $\leq v$(**MCBS**). Example 1 (see below) provides an example for which the inequality is strict. □

Example 1 (continued). For **MCBS-2** model, the optimal resources allocation, where 4 VM_1, 7 VM_2 and 2 VM_3 are used, is displayed in Figure (1a). To reach this solution, first, the model **MCBS-M** with $U_1 = 4$, $U_2 = 7$, $U_3 = 4$, $U = 20$ is solved



(a) Resource allocation provided by **MCBS-2**



(b) Resource allocation provided by **MCBS-ins**

Fig. 1 Optimal resources allocation for $D = 8$ time units and $U_1 = 4$, $U_2 = 7$, $U_3 = 4$, $U = 20$ VMs

to determine the minimum makespan, $\theta^* = 8$. Then the model **MCBS** for $D = 8$ provides the solution with the lowest cost, 43$, displayed in Table 5.

The optimal solution of model **MCBS-ins** has a cost equal to 42.4$ which is strictly lower than v(**MCBS**). Figure (1b) displays the optimal resources allocation. This solution, where 9 VM_1, 14 VM_2 and 2 VM_3 are allocated, is not a feasible solution for **MCBS**. However, the number of instances working in any time period is at most 4 for VM_1, 7 for VM_2 and 2 for VM_3.

The next proposition allows to fix N-variables to zero in order to reduce the size of the problem.

Proposition 4.2 *Any optimal solution of* **MCBS-ins** *satisfies that for* $i \in I$, $b \in \mathcal{B}_i$, $k \in K$, $j \in J_k$, $t_0 \in T$:

$$N_{ibkj}^{dt_0} = 0,\ d \in T,\ d \neq \left\lceil m \cdot \tau_{ibkj} \right\rceil,\ m = 1, 2, \ldots$$

Proof Let us assume that there exists an optimal solution $(\overline{y}, \overline{N})$ of **MCBS-ins** and a variable $\overline{N}_{ibkj}^{d_0 t_0} > 0$ with $d_0 \neq \left\lceil m \cdot \tau_{ibkj} \right\rceil$ for a $m \in \mathbb{N}$. Suppose that $\left\lceil m \cdot \tau_{ibkj} \right\rceil < d_0 < \left\lceil (m+1) \cdot \tau_{ibkj} \right\rceil$. We can build a feasible solution $(\overline{y}, \widehat{N})$, with $\widehat{N}_{ibkj}^{dt_0} = \overline{N}_{ibkj}^{dt_0}$, except for $\widehat{N}_{ibkj}^{d_0 t_0} = 0$ and $\widehat{N}_{ibkj}^{\lceil m \cdot \tau_{ibkj} \rceil t_0} = \overline{N}_{ibkj}^{d_0 t_0}$. This feasible solution corresponds to a better objective function value:

$$w_{ibkj}^{d_0} = w_{ibkj}^{\lceil m \cdot \tau_{ibkj} \rceil} \text{and} (P_{kj} \cdot \left\lceil m \cdot \tau_{ibkj} \right\rceil) \widehat{N}_{ibkj}^{\lceil m \cdot \tau_{ibkj} \rceil t_0} < (P_{kj} \cdot d_0) \overline{N}_{ibkj}^{d_0 t_0}$$

This is in contradiction with the optimality of $(\overline{y}, \overline{N})$. □

In proposition 4.2, we consider tasks requiring more than one time period to be solved by an instance type. Any optimal solution will use that instance type during a time which is a multiple of the time required for a task. The same arguments hold for **MCBS**.

Lemma 4.3 *Any optimal solution of* **MCBS** *satisfies that for* $i \in I$, $b \in \mathcal{B}_i$, $k \in K$, $j \in J_k$, $d \in T$:

$$N_{ibkj}^{d} = 0,\ d \neq \left\lceil m \cdot \tau_{ibkj} \right\rceil,\ m = 1, 2, \ldots$$

The next proposition aims to reduce the symmetry introduced by model **MCBS-ins**. More precisely, each feasible solution leads to alternative feasible solutions with the same objective function value by splitting the time during a machine is working as long as the number of tasks executed is not improved when the length of working time is increased. This result is not valid for model **MCBS** since the upper bound refers to the number of instances.

Proposition 4.4 *An optimal solution of* **MCBS-ins** *exists such that for* $i \in I$, $b \in \mathcal{B}_i$, $k \in K$, $j \in J_k$, $t_0 \in T$ *such that* $\tau_{ibkj} < 1$:

$$N_{ibkj}^{dt_0} = 0, d \in T \text{ such that } d \geq 2 \text{ and } w_{ibkj}^{d} - w_{ibkj}^{d-1} \leq w_{ibkj}^{1}$$

Proof Suppose there exists an optimal solution $(\overline{y}, \overline{N})$ of **MCBS-ins** which does not satisfy the statement conditions.

Since $(\overline{y}, \overline{N})$ does not satisfy the statement conditions, there exists a variable $\widehat{N}_{ibkj}^{d_0 t_0} > 0$ with $d_0 \geq 2$ and $w_{ibkj}^{d_0} - w_{ibkj}^{d_0-1} \leq w_{ibkj}^{1}$ for some d_0. Notice that w_{ibkj}^{d} defines the number of tasks of BoT $b \in \mathcal{B}_i$ executed by a instance type j during exactly d time units, $k \in K$, $j \in J_k$, $i \in I$, $b \in \mathcal{B}_i$, $d \in T$. When $\tau_{ibkj} < 1$, the number of tasks of BoT $b \in \mathcal{B}_i$ executed by an instance type j during one time unit is at least one:

$$w_{ibkj}^{d} = \left\lfloor \frac{d}{\tau_{ibkj}} \right\rfloor \Rightarrow d = w_{ibkj}^{d} \tau_{ibkj} + \epsilon_d \text{ with } \epsilon_d < \tau_{ibkj}$$

For $d \geq 2$:

$$\begin{aligned} d &= (d-1) + 1 = w_{ibkj}^{d-1} \tau_{ibkj} + \epsilon_{d-1} + 1 = w_{ibkj}^{d-1} \tau_{ibkj} + \epsilon_{d-1} + w_{ibkj}^{1} \tau_{ibkj} + \epsilon_1 \\ &= (w_{ibkj}^{d-1} + w_{ibkj}^{1}) \tau_{ibkj} + (\epsilon_{d-1} + \epsilon_1) \end{aligned}$$

Moreover,

$$0 \leq \epsilon_{d-1} + \epsilon_1 < 2\tau_{ibkj}$$

Therefore, for $d \geq 2$:

$$w_{ibkj}^{d} = \begin{cases} w_{ibkj}^{d-1} + w_{ibkj}^{1} & \text{if } \epsilon_{d-1} + \epsilon_1 < \tau_{ibkj} \\ w_{ibkj}^{d-1} + w_{ibkj}^{1} + 1 & \text{if } \epsilon_{d-1} + \epsilon_1 \geq \tau_{ibkj} \end{cases}$$

Since $w_{ibkj}^{d_0} - w_{ibkj}^{d_0-1} \leq w_{ibkj}^{1}$ and $d_0 \geq 2$, we conclude that $w_{ibkj}^{d_0} = w_{ibkj}^{d_0-1} + w_{ibkj}^{1}$.

We can build a feasible solution $(\overline{y}, \widehat{N})$, with $\widehat{N}_{ibkj}^{dt_0} = \overline{N}_{ibkj}^{dt_0}$, except for:

$$\begin{aligned} \widehat{N}_{ibkj}^{d_0 t_0} &= 0 \\ \widehat{N}_{ibkj}^{(d_0-1)t_0} &= \overline{N}_{ibkj}^{d_0 t_0} + \overline{N}_{ibkj}^{(d_0-1)t_0} \\ \widehat{N}_{ibkj}^{1(t_0+d_0-1)} &= \overline{N}_{ibkj}^{d_0 t_0} \end{aligned}$$

The new feasible solution has the same objective value function than $(\overline{y}, \overline{N})$. In the case of $\widehat{N}_{ibkj}^{(d_0-1)t_0} > 0$ such that $d_0 - 1 \geq 2$ and $w_{ibkj}^{d_0-1} - w_{ibkj}^{d_0-2} \leq w_{ibkj}^{1}$, we repeat the process for $d_0 - 1$. □

Model **MCBS-M** can be extended to the **MCBS-Mins** problem to include instantaneous resource constraints. The mathematical formulation is written as follows:

$$\min_{y,N,\theta,z} \theta \tag{5a}$$

$$\text{s.t.:}(4b) - (4h) \tag{5b}$$

$$N_{ibkj}^{dt_0} \leq U_{kj} z_{t_0+d-1}, i \in I, \quad b \in \mathcal{B}_i, k \in K, j \in J_k, \\ d,\ t_0 \in T, \quad t_0 + d - 1 \leq D \tag{5c}$$

$$\theta \geq d z^d, \quad d \in T \tag{5d}$$

$$z^d \in \{0, 1\}, \quad d \in T \tag{5e}$$

5 Computational experiments

In this Section, we present the results of some experiments carried out on a synthetic dataset generated from real data. Table 6 provides a summary of the above proposed models. The goal is to analyse in a realistic case the differences between the models and discuss their advantages and disadvantages. All models have been solved by using CPLEX 12.9.0.0. in a Intel(R) Core(TM) i7-9700 CPU 3.00GHz 32.0 GB RAM with Windows 10.

5.1 Dataset

The test instances have been defined in accordance with the features of instances defined by well-known public cloud providers. They usually offer some specific VMs better adapted for intensive computing. This is the case of the C4, C5 and C5n families for Amazon EC2, the Fsv2, Fs and the F series for MS-Azure and the C2 Compute Optimized family for Google Cloud. Table 7 shows the synthetic list of cloud providers and instances used for the experiments. We are considering three virtual cloud providers, namely C_1, C_2, C_3, offering, respectively, 6, 4 and 5 VMs types. Let VM_{kj} denotes a VM of instance type j in cloud C_k.

Table 6 Summary of the proposed models for Multi-Cloud Bot Scheduling

Single objective otpimization models	
MCBS:	Cost optimization with a fixed deadline
MCBS-M:	Makespan optimization
Lexicographic approaches to a bi-objective optimization models	
MCBS-1:	First: cost minimization. Second: makespan optimization
MCBS-2:	First: makespan minimization. Second: cost optimization
Models with capacity constraints	
MCBS-ins:	Cost optimization with instantaneous constraints
MCBS-Mins:	Makespan optimization with instantaneous constraints

Table 7 Instance types specifications of the three clouds

Cloud	Instance	Virtual CPUs	Ram (Gb)	Price($/time unit)	CCU	Limit (U=100)
C_1	VM_{11}	2	4	10.2	90	50
	VM_{12}	4	8	20.4	170	25
	VM_{13}	36	72	183.6	1410	2
	VM_{14}	48	96	244.8	1880	2
	VM_{15}	72	144	367.2	2810	1
	VM_{16}	96	192	489.6	3750	1
C_2	VM_{21}	1	2	6.56	60	100
	VM_{22}	2	4	13.08	90	50
	VM_{23}	4	8	26.28	170	25
	VM_{24}	16	32	105	640	6
C_3	VM_{31}	4	16	25.06	240	25
	VM_{32}	8	32	50.11	460	12
	VM_{33}	16	64	100.22	900	6
	VM_{34}	30	120	187.92	1670	3
	VM_{35}	60	240	375.85	3300	1

Like EC2, we define the resource limit constraints in terms of vCPUs. Let U_k be the CPU quota for cloud $k \in K$ and vc_{kj} be the number of vCPU of the instance type $j \in J_k$, $k \in \{C_1, C_2, C_3\}$. Notice that, for a given upper bound U_k, an upper bound on the number of VMs of instance type j is computed as follows:

$$U_{kj} = \left\lfloor \frac{U_k}{vc_{kj}} \right\rfloor, k \in K, j \in J_k$$

Therefore, the total number of VMs depends on their types and any combination is allowed while the total number of vCPUs does not exceed the limit. Last column of Table 7 shows the resource limit constraint for each instance type when the upper bound on the total number of vCPUs of each cloud is 100. Regarding the applications and BoTs, they have not established in an arbitrary way, but they have been chosen from the literature (Juve et al. 2013) so as to have a variety in terms of running times and number of tasks. We consider three applications $I = \{A_1, A_2, A_3\}$ and three BoTs in each application. BoT in application A_i are denoted by B_{ib} with $b \in \{1, 2, 3\}$. As in Abdi et al. (2017), we define the computational size of an application (CSA) as the execution time of its tasks on a VM with a performance of 1 CCU. The data are summarized in Table 8.

5.2 Computing the minimum makespan for a set of given resource limits

In the resource allocation problem, once the availability for each cloud provider is known, we compute the minimum makespan, that is, a lower bound on the deadline

Table 8 Tasks in the BoT workflows of each application

Application	BoT	Count	Runtime	CSA
A_1	B_{11}	17	282.37	
	B_{12}	2102	1.73	
	B_{13}	6172	0.66	12510.27
A_2	B_{21}	96	3.06	
	B_{22}	96	84.92	
	B_{23}	96	196.40	27300.48
A_3	B_{31}	1	55.95	
	B_{32}	7	34.32	
	B_{33}	8	11.01	384.27

Table 9 Minimum makespan in time units and CPU times in seconds for each upper bound on the number of instances

U	MCBS-M		MCBS-Mins	
	θ^* time units	CPU time (seconds)	θ^* time units	CPU time (seconds)
25	20	3.28	20	45.69
50	10	2.75	10	16.74
75	7	4.86	7	48.63
100	5	2.76	5	75.38
125	5	3.28	4	47.44
150	4	3.5	4	30.33
175	3	2.58	3	19.51
200	3	3.03	3	25.5
225	3	3.77	3	24.69

needed to deal with all BoTs. For the sake of clarity, we have run the experiments with the same resource limits for the three clouds, that is, $U_k = U, k \in \{C_1, C_2, C_3\}$. The experiments are carried out for different values of $U \in \{25, 50, , \ldots, 200, 225\}$. By solving **MCBS-M** and **MCBS-Mins** models we determine the minimum makespan ensuring that all the jobs could be finished.

Table 9 shows in the second and third columns, the minimum makespan θ^* and the CPU times required to solve model **MCBS-M**, respectively. The fourth and fifth columns include the same values for model **MCBS-Mins**.

Model **MCBS-M** requires a smaller computation time than **MCBS-Mins**. Since the feasible region of **MCBS-Mins** includes the feasible region of **MCBS-M**, this model may provide a worse solution than the one of **MCBS-Mins** model as for $N = 125$. This behaviour is described in Fig. 2. When the resource limits are lower, the results given by **MCBS-Mins** are sharply better than those of **MCBS-M**. However, the maximum difference is 3 time units. As the availability of resources increases, the reduction of the deadline is slower, resulting in the same solution for both models.

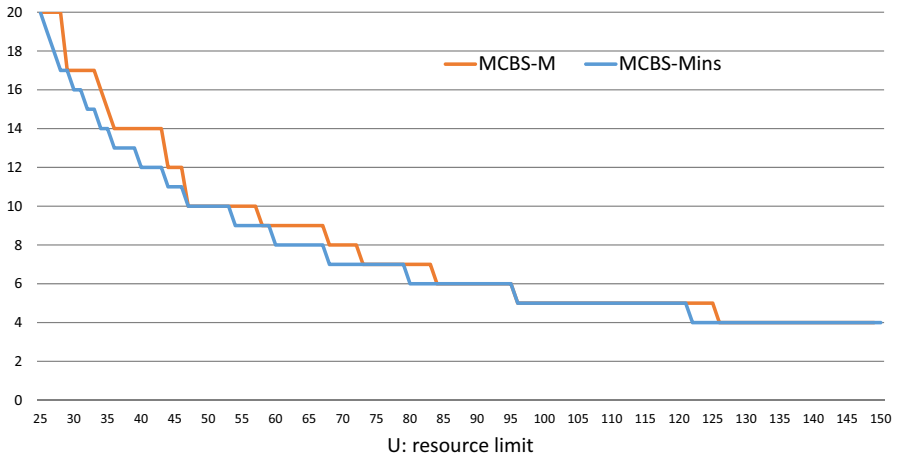


Fig. 2 Minimum makespan in time units vs Upper resources limit on VMs

5.3 Effect of the deadline in the cost

In Sect. 5.2 we have seen how an upper bound on the number of resources of each cloud provider allows to determine the minimum deadline to operate all the BoTs by solving models **MCBS-M** and **MCBS-Mins**. However, when the deadlines are extended, it is interesting to identify their impacts on the costs. In this subsection, we consider an upper bound $U = 50$ on the capacity of resources of the three cloud providers and compute the minimal cost for several values of D. We solve model **MCBS**, whose capacity constraint is globally imposed as the number of instances allowed along the deadline period, and model **MCBS-ins** in which the limits refer to the number of instances allowed to work in simultaneous way. We also compare these results with the value provided by model **IP-NDS**.

Table 10 displays the minimum cost and the computational time in seconds for the three models and four deadlines. **IP-NDS** model is a very simple allocation problem which is solved in less than one second. As in the previous subsection, model **MCBS** requires a much lower computation time than model **MCBS-ins**. The running time of CPLEX is limited to 300 s. When CPLEX is interrupted after

Table 10 Minimum cost ($) and CPU times (seconds) vs the deadline ($U = 50$ VMs)

D	IP-NDS		MCBS		MCBS-ins	
	Cost	CPU time	Cost	CPU time	Cost	CPU time
10	4468.212	0.01	4443.108	0.97	4387.248	7.88
15	4387.248	0.03	4387.248	47.58	4335.576	5.91
20	4335.576	0.02	4310.52	0.7	4310.52	300.66(0.014)
25	4335.576	0	4285.464	0.38	4285.464	300.63(0.010)

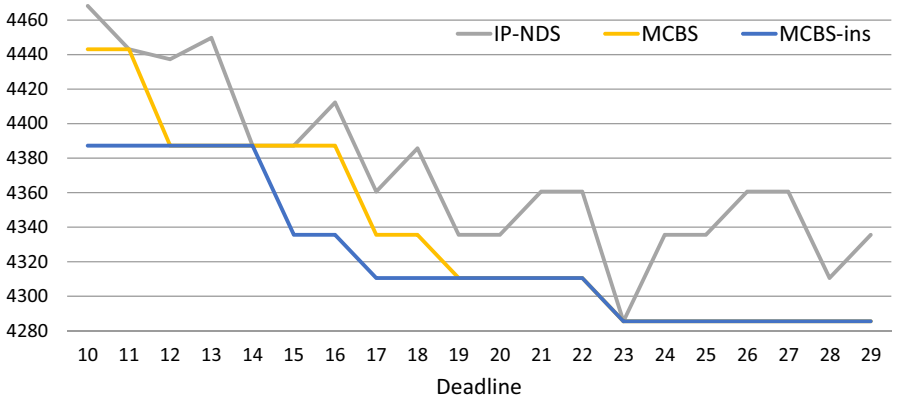


Fig. 3 Minimum cost ($) vs deadline (time units) ($U = 50$ VMs)

this limit, the best integer solution is provided and we include between parenthesis the MIP relative gap which refers to the quotient of the best integer objective minus the objective of the best node remaining between the best integer objective. For instance, for a deadline $D = 25$ both models **MCBS** and **MCBS-ins** provide the same objective value, 4285.464, however only model **MCBS** guarantees that this value is the optimal one. In the case of **MCBS-ins**, the relative gap of 0.01 means that CPLEX has found a feasible integer solution in the 1 percent of the optimal one.

Figure 3 displays the minimum cost behaviour as the deadline increases. Model **MCBS-ins** results in a lower cost value than **MCBS** for some deadlines. As expected, with a global upper bound on the number of vCPUs (model **MCBS**), a higher deadline is required to generate a cost similar to the one provided by **MCBS-ins** with shorter deadlines. As shown in Fig. 3 both **MCBS** and **MCBS-ins** are non-increasing monotone functions. However, **IP-NDS** makes use of the full deadline to allocate the instances which may result in a non monotone cost function. For **MCBS-ins**, the optimal value displayed is guaranteed to be the optimal only for $D \in \{10, 11, 12, 13, 23, 27, 28\}$. The remaining values refer to the best integer solution value. The difference between the costs of the solutions provided by **MCBS** and **MCBS-ins** ranges from 0 ($D \geq 19$) to 1.26% ($D = 10, 11$).

5.4 Results from a biobjective perspective

Models **MCBS-1** and **MCBS-2** integrate jointly the cost and the makespan. In Sect. 5.2, the minimum makespan has been computed for a given set of resource limits, while in Sect. 5.3 the minimum cost is obtained for several values of deadline. The solutions are optimal when only one of the criteria is considered. Following, models **MCBS-1** and **MCBS-2** are applied for computing the corresponding efficient solutions.

Table 11 shows in the first and second columns the bounds considered for the number of resources and the deadline, respectively. The third and fourth columns

Table 11 Efficient solutions for the biobjective problem in which the cost and the makespan are minimized

U	D	MCBS-1		MCBS-2	
		Minimum cost ($)	Minimum makespan (time units)	Minimum makespan (time units)	Minimum cost ($)
25	25	4403.772	23	20	4430.028
50	10	4443.108	10	10	4443.108
50	15	4387.248	12	10	4443.108
50	20	4310.52	19	10	4443.108
50	25	4285.464	23	10	4443.108
75	25	4285.464	23	7	4587.564
100	25	4285.464	23	5	4387.248
125	25	4285.464	23	5	4387.248
150	25	4285.464	23	4	4597.92
175	25	4285.464	23	3	4622.976
200	25	4285.464	23	3	4622.976
225	25	4285.464	23	3	4622.976

include the cost and the makespan for the efficient solution computed by model **MCBS-1** which first optimizes the cost. Model **MCBS-2** first optimizes the makespan which is displayed at the fifth column, while the sixth column shows the minimum cost given that optimal makespan. Each row provides two efficient solutions for a given resource limit and a deadline. For instance, for $U = 25$ and $D = 25$, the minimum cost is 4403.772$ and to achieve this cost at least 23 units of time are required. The only way to reduce the makespan and reach the minimum value of 20

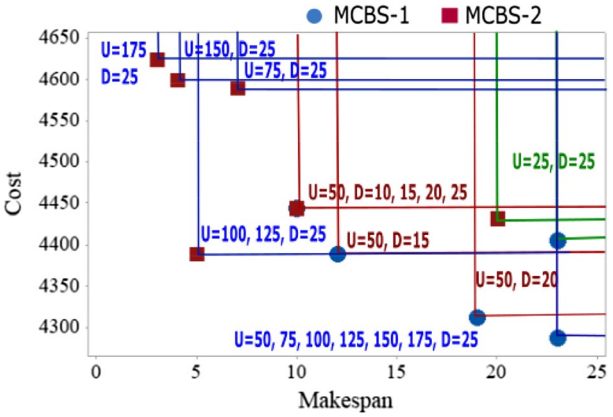


Fig. 4 Cost ($) and makespan (time units) of efficient solutions for several values of U (limit on the number of VMs) and deadlines D (time units) obtained by solving models **MCBS-1** and **MCBS-2**

time units is to increase the cost. The minimum cost required to perform BoTs in 20 time units is 4430.028$.

The values of the cost and the makespan of the efficient solutions are displayed in the scatterplot displayed in Fig. 4. With a deadline $D = 25$, increasing the resource limits above $U = 175$ does not provide better solutions neither in terms of the cost nor the makespan. Therefore the figure does not include the values corresponding to the $U \in \{200, 225\}$. Model **MCBS-1** first optimize the cost, therefore the corresponding efficient solutions provides lower values than the ones provided by model **MCBS-2**. On the contrary, the lower values of the makespan are obtained by solving model **MCBS-2**.

Notice that, for $U = 50$ and deadlines $D \in \{10, 15, 20, 25\}$, the minimum makespan is 10 time units and the minimum cost with $D = 10$ is 4443.108$. For this reason, model **MCBS-2** provides the same values of both objectives for all deadlines. However, model **MCBS-1** provides better cost values as the deadline increases, while the makespan gets worse. For deadline $D = 25$, the efficient solution obtained by solving **MCBS-2** with $U = 100$ is better than the one with $U = 75$. In this case, increasing the upper resources limit allow to decrease the cost and the makespan simultaneously. However, the efficient solution obtained by solving **MCBS-2** with $U \in 100, 150, 175$ are not comparable.

5.5 Analysis of the two approaches when introducing capacity constraints

The previous computational experience raises the question of the impact to consider instantaneous or global capacity constraints. Notice that, given a deadline D, to make the best possible use of resources, the instances will be hired as long as possible in order to reduce the idle time in each time unit. When the instances are used until the deadline is met, like master VM in **IP-NDS** model, the instantaneous and global capacity constraints would coincide.

From the above results, it is clear that the **MCBS-ins** and **MCBS-Mins** models can provide better results than the **MCBS** and **MCBS-M** models, respectively. Note that the feasibility sets of **MCBS** and **MCBS-M** are included in the corresponding ones of **MCBS-ins** and **MCBS-Mins**. This is in line with the intuitive idea that a capacity constraint on each time unit offers more flexibility than a global capacity constraint.

For illustrative purpose, we set $U = 50$ and $D = 17$ and we solve both models. Table 12 shows from the second to the fourth columns the optimal solution (S_1) provided by model **MCBS** with a cost 4335.576. The best integer solution (S_2) provided by model **MCBS-ins** when interrupted after 300 s with a cost 4310.52 is displayed from the fifth to eighth columns. As both solutions are very similar, we have highlighted the differences in bold. Solution S_1 uses 48 and 6 vCPUs from clouds C_3 and C_1, respectively. For solution S_2 the number of vCPUs from cloud C_3 goes from 16 at the 17th time period to 48 vCPUs instantaneously used at 6 out of 17 times periods. Notice that S_2 is not a feasible solution of **MCBS** since for cloud C_3 the total number of vCPUs are 80. On the other hand, the instantaneous capacity constraint makes

Table 12 Description of the solutions provided by models **MCBS** and **MCBS-ins** with $D = 17$ time units

BoT	MCBS			MCBS-ins			
	Instance	d	N	Instance	t_0	d	N
B_{11}	VM_{31}	5	1	VM_{31}	2	6	1
	VM_{31}	**16**	**1**	**VM_{31}**	**3**	**13**	**1**
				VM_{32}	**6**	**1**	**1**
B_{12}	VM_{31}	16	1	VM_{31}	11	2	1
				VM_{31}	13	3	1
				VM_{31}	7	11	1
B_{13}	VM_{31}	17	1	VM_{31}	2	1	1
				VM_{31}	2	16	1
B_{21}	VM_{31}	2	1	VM_{31}	1	1	1
				VM_{31}	2	1	1
B_{22}	VM_{31}	17	2	VM_{31}	1	17	2
B_{23}	**VM_{31}**	**15**	**2**	**VM_{31}**	**7**	**5**	**1**
	VM_{31}	**17**	**3**	**VM_{31}**	**12**	**5**	**1**
				VM_{31}	**1**	**14**	**5**
B_{31}	VM_{11}	1	1	VM_{11}	1	1	1
B_{32}	VM_{11}	3	1	VM_{11}	8	1	1
				VM_{11}	9	2	1
B_{33}	VM_{11}	1	1	VM_{11}	2	1	1

Table 13 Number of binary and integer variables in the models

	IP-NDS	MCBS MCBS-ins	MCBS-M MCBS-Mins	MCBS MCBS-M	MCBS-ins MCBS-Mins	MCBS-ins MCBS-Mins
D	Binary	Binary	Binary	Integer	Integer	Reduced
10	144	9	19	1350	13500	700
15	144	9	24	2025	30375	1015
20	144	9	29	2700	54000	1342
25	144	9	34	3375	84375	1654

possible to execute all the tasks in bag B_{23} paying for 80 time units of instances VM_{31} in solution S_2 instead of 81 time units needed in solution S_1.

From a mathematical point of view, a global capacity constraint only relies on the duration of each machine, which reduces the number of integer variables in the **MCBS** and **MCBS-M** models. However, to consider instantaneous capacity constraint requires to define the starting time of each instance to be able to evaluate at each time period then number of instances used from each cloud. Propositions 4.2 and 4.3 allow to reduce the number of integer variables. For illustrative purposes, we consider a capacity $U = 50$ for all the clouds and compute the number of binary and integer variables for each model. Table 13 displays these numbers for D varying

Table 14 Parameters of the uniform random distribution for each BoT

BoT	Lower bound	Average	Upper bound
B_{11}	254.13	282.37	310.61
B_{12}	1.56	1.73	1.90
B_{13}	0.59	0.66	0.73
B_{21}	2.75	3.06	3.37
B_{22}	76.43	84.92	93.41
B_{23}	176.76	196.40	216.04
B_{31}	50.36	55.95	61.54
B_{32}	30.89	34.32	37.75
B_{33}	9.91	11.01	12.11

Table 15 Generated running times (time units) for each BoT and each Set

BoT	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9	Set 10
B_{11}	309.5	260.6	259.0	273.9	306.9	261.8	296.1	266.2	268.4	304.5
B_{12}	1.7	1.8	1.6	1.9	1.7	1.6	1.8	1.6	1.8	1.8
B_{13}	0.7	0.6	0.6	0.7	0.7	0.6	0.7	0.6	0.6	0.7
B_{21}	3.2	3.0	3.2	3.3	3.2	3.2	2.8	3.1	3.3	2.9
B_{22}	88.2	83.3	84.9	76.7	77.6	85.8	80.8	78.3	89.3	90.3
B_{23}	211.1	200.0	194.5	203.7	203.5	214.0	187.1	186.3	192.5	199.1
B_{31}	55.5	55.5	55.2	53.0	52.7	54.0	57.4	58.8	58.5	60.1
B_{32}	35.4	34.2	32.1	36.2	36.0	33.7	34.2	36.6	33.1	35.7
B_{33}	9.9	11.1	10.3	11.5	10.1	11.1	11.0	10.4	11.6	10.7

from 10 to 25. The seventh column shows the number of integer variables different from 0 after applying both Propositions. Even if the number of integer variables is sharply reduced by up to 5%, the computational time is not decreased.

Bearing in mind previous results, it may occur that the improvements achieved are not large enough to justify the computational effort required to solve the **MCBS-Mins** and **MCBS-ins** models. Notice that the instantaneous capacity constraints rely on estimation of the execution time required for a task. Pham et al. (2020) introduce an efficient method to estimate the required execution time with only a few real executions, obtaining estimation errors below 10% and show that alternative methods display similar or worse results. We next illustrate these estimation errors impacts on the optimal value of **MCBS** and **MCBS-ins** models.

The running times displayed in Table 8 are considered as the Set 0. We define a uniform random distribution with a lower (respectively an upper bound) equal to 90% (respectively 110%) of the Set 0 running times. Thus, the average times of the uniform distribution are exactly the running times of the Set 0. The lower and upper bounds of the associated uniform distribution are shown in Table 14. From these distributions we generate 10 sets of running times for the nine BoTs.

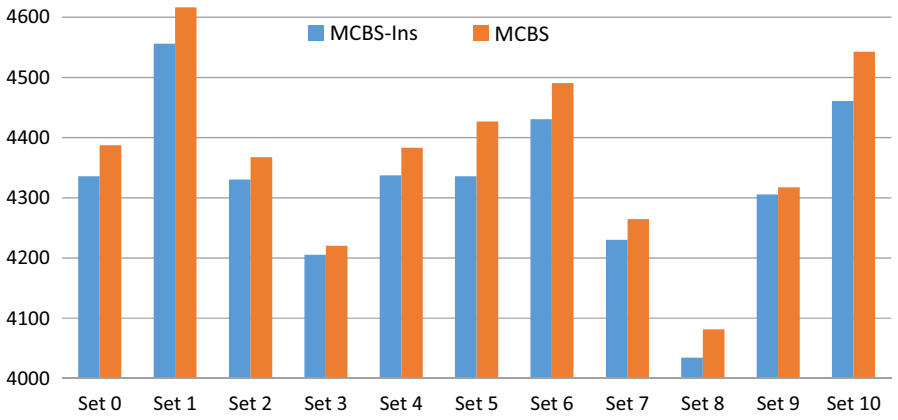


Fig. 5 Minimum cost ($) vs set of running times ($U = 50$ VMs, $D = 15$ time units)

Fig. 6 Paired T-Test of the minimum costs ($) obtained by **MCBS** and **MCBS-ins** models

μ_difference: mean of (MCBS - MCBS-ins)

Mean	StDev	SE Mean	95% Lower Bound for μ_difference
48.80	24.36	7.35	35.49

Test

Null hypothesis	H_0: μ_difference = 0
Alternative hypothesis	H_1: μ_difference > 0

T-Value	P-Value
6.64	0.000

Table 15 displays for each BoT the value of the generated running time for each set.

MCBS and **MCBS-ins** models with $U = 50$ and $D = 15$ are solved for each set of running times.

Figure 5 shows the minimum cost obtained for both models for each set. The Set 0 consists in the running times defined in Table 8. The difference between the minimum costs for each set ranges from 0.285% (Set 9) to 2.1% (Set 5). We also compare the minimum cost for each set with the cost of Set 0. For **MCBS-ins** model, the difference ranges from −6.93% (Set 8) to 5.08% (Set 1). Similar values for **MCBS** model, from −6.96% (Set 8) to 5.22% (Set 1).

We have performed a paired sample t-test to evaluate whether **MCBS-ins** model provides better solutions than **MCBS** in terms of cost. In this statistical test, each set of running times gives a pair of observations, one for each model. As Fig. 6 displays a p-value equals to 0, the minimum cost obtained by **MCBS** model is significantly greater than the one obtained by **MCBS-ins**. In fact, with a confidence level of 95% the mean of the differences will be at least

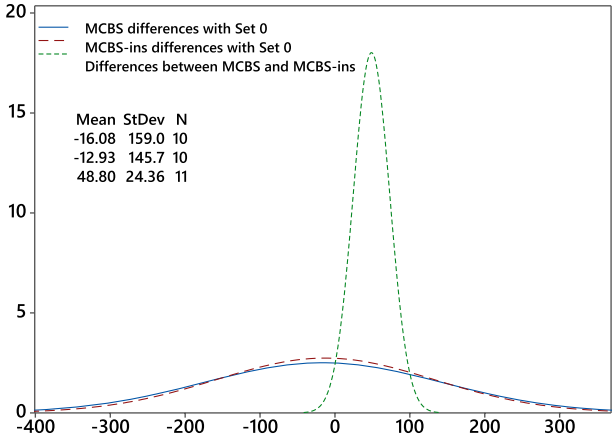


Fig. 7 Differences among the minimum costs obtained by **MCBS** and **MCBS-ins** models for each set of running times

equal to 35.49. As expected, the mean of the differences between the minimum cost obtained with each set of running times is not significantly different from the corresponding minimum cost obtained with Set 0 since these differences are positive and negative. All of these differences are shown in Fig. 7. In conclusion the range of the difference to the minimum cost corresponding to Set 0 due to the estimation error associated with the running times is much larger than the differences between solutions obtained by **MCBS** and **MCBS-ins** models. The results suggest that it is not worth the computational effort of solving model **MCBS-ins** compared to model **MCBS** in terms of cost improvement.

6 Conclusions and future research

The paper defines new optimization models integrating the VM type selection, resource scaling and workload allocation in a multi-cloud environment. These models only deal with one objective, the minimization of cost considering a deadline and the minimization of the makespan to run all the BoTs, respectively. In addition, we propose two models which involve jointly the costs and makespan objective functions. Two efficient solutions of the biobjective problem are computed by using a lexicographic approach. A first efficient solution is computed by prioritizing cost over makespan and the second one takes the reverse order. Capacity constraints with respect to the number of allowed resources imposed by the cloud providers in the case of public clouds or the cloud system itself in the case of private clouds are also considered.

The limits in the number of VM instances are defined according to two different paradigms. First, a global upper bound is considered, regardless of the time the instances are used. This approach is in accordance with those models assuming that VMs work as long as possible to complete the deadline. In the literature, most of the mathematical models looking for exact solutions assume this constraint. In contrast, we consider in this work the time VMs are working for executing the BoTs,

extending the models and increasing the feasibility space. A second approach establishes an instantaneous bound on the number of VMs. This approach increases the number of feasible solutions, leading to better solutions in terms of cost, but increasing significantly the computation time as shown in the computational experience.

Finally, experiments carried out on synthetic realistic data put into highlight the cost improvements of the proposed solutions compared to the solutions of previous models and the effect of the model parameters in the cost and the makespan. A first analysis shows how the minimum makespan required to execute all BoTs decreases as the number of available resources increases, and stabilizes when a certain value is reached. Secondly, we consider fixed the resource limit in order to compute the minimum cost. The minimum values obtained by solving the proposed models are non-increasing monotone functions of the deadline. In addition to the individual analysis of cost and makespan, efficient solutions have been calculated for a bi-objective problem using a lexicographic approach for several values of resource limits and deadline. There is a great impact on the computational time for solving the model when instantaneous capacity constraints are considered, which is not accompanied by a noticeable improvement in the values of the cost or the makespan. In fact, we have seen that the range of the differences in the minimum cost due to the estimation error associated with the running times is much larger than the differences between solutions obtained with instantaneous or global capacity constraints.

As future work we aim at extending the mathematical models to consider more general applications, such as acyclic graphs of BoTs or even more general workflow structures that besides the parallel execution of tasks in a BoT must also consider some precedence relations among the different BoTs. In addition, the proposed models only consider on–demand instances and it is also assumed that the execution time of each task of a BoT remains constant. Interesting future work will be to see how to include reserved VMs and how to remove the assumption on the running times. Finally, we also aim at reducing the computational cost of the considered solutions by strengthening the formulations and defining specific solution methods that could take advantage of the specific structure of the problem formulation. The use of fine grained time units may result in decreased savings and raises the question of whether the computation time required is justified. As a result, reducing computation time by utilizing the model's specific properties, similar to those examined in Propositions 4.2, 4.4 and Lemma 4.3, is considered a crucial future task.

Appendix

Abdi et al. (2017) propose a binary formulation for solving the resource allocation problem for the execution of BoTs in intercloud systems. The execution time of each task of BoT $b \in \mathcal{B}_i$, $i \in I$ with an instance type j in cloud k, $k \in K$, $j \in J_k$ is computed as follows:

$$\tau_{ibkj} = \frac{ET_{ib}}{ccu_{kj}}$$

Once it is known, the maximum number of tasks of BoT $b \in \mathcal{B}_i$ solved by one VM of instance type j in cloud k for a given deadline D is:

$$\eta_{ibkj} = \min\left(\delta_{ib}, \left\lfloor \frac{D}{\tau_{ibkj}} \right\rfloor\right), k \in K, j \in J_k, i \in I, b \in \mathcal{B}_i$$

where $\lfloor \cdot \rfloor$ denotes the floor function. Notice that, $\eta_{ibkj} = 0$ states that the assignment of a VM of instance type j in cloud k to BoT $b \in \mathcal{B}_i$ is infeasible because it cannot meet specified deadline D. Otherwise, the number of required VMs of instance type j for executing tasks of BoT $b \in \mathcal{B}_i$ is:

$$\nu_{ibkj} = \left\lfloor \frac{\delta_{ib}}{\eta_{ibkj}} \right\rfloor, \ k \in K, j \in J_k, i \in I, b \in \mathcal{B}_i$$

These VMs are used until the deadline D is met and are called master VMs. Notice that, the running time ot the master VMs is computed as follows:

$$T_{ibkj} = \left\lceil \tau_{ibkj}\eta_{ibkj} \right\rceil, k \in K, j \in J_k, i \in I, b \in \mathcal{B}_i$$

and it is supposed to be approximately the deadline D. It is possible that $\tau_{ibkj}\eta_{ibkj} < \delta_{ib}$. That means that the master VMs of the instance type j in cloud k cannot carry out the computation of all tasks of BoT $b \in \mathcal{B}_i$. In this case, an extra VM of the same instance type is required during the following time in hours:

$$TR_{ibkj} = \left\lceil (\delta_{ib} - (\nu_{ibkj}\eta_{ibkj}))\tau_{ibkj} \right\rceil, k \in K, j \in J_k, i \in I, b \in \mathcal{B}_i$$

If $TR_{ibkj} > 0$, then an extra VM should be used and XR_{ibkj} is set to 1. Otherwise, $XR_{ibkj} = 0$. The makespan of BoT $b \in \mathcal{B}_i$ on instance type j in cloud k is computed as:

$$M_{ibkj} = \max\{T_{ibkj}, TR_{ibkj}\}, k \in K, j \in J_k, i \in I, b \in \mathcal{B}_i$$

When $\eta_{ibkj} = 0$ and the allocation of instance type j in cloud k to tasks of BoT $b \in \mathcal{B}_i$ is not feasible, the coefficients T_{ibkj} and TR_{ibkj} are set to ∞.

The decision variables are:

$$y_{ik} = \begin{cases} 1 \text{ if application } i \text{ is submitted to cloud } k, \\ 0 \text{ otherwise,} \end{cases} \quad i \in I, k \in K$$

$$x_{ibkj} = \begin{cases} 1 \text{ if instance type } j \text{ in cloud } k \text{ is selected for BoT } b \in \mathcal{B}_i, \\ 0 \text{ otherwise,} \end{cases}$$
$$k \in K, j \in J_k, i \in I, b \in \mathcal{B}_i$$

The allocation model, denoted by **IP-NDS**, is written as follows:

$$\min_{y,x} \sum_{k\in K}\sum_{j\in J_k} P_{kj} \sum_{i\in I}\sum_{b\in\mathcal{B}_i} x_{ibkj}(v_{ibkj}T_{ibkj} + TR_{ibkj}) \tag{6a}$$

$$\text{s.t.:} \sum_{k\in K} y_{ik} = 1, i \in I \tag{6b}$$

$$\sum_{b\in\mathcal{B}_i}\sum_{j\in J_k} x_{ibkj} = y_{ik}|\mathcal{B}_i|, i \in I,\ k \in K \tag{6c}$$

$$x_{ibkj} \leqslant y_{ik}, i \in I,\ b \in \mathcal{B}_i\, k \in K,\ j \in J_k \tag{6d}$$

$$\sum_{k\in K}\sum_{j\in J_k} x_{ibkj} = 1, i \in I,\ b \in \mathcal{B}_i \tag{6e}$$

$$\sum_{i\in I}\sum_{b\in\mathcal{B}_i} x_{ibkj}(v_{ibkj} + XR_{ibkj}) \leq U_{kj}, k \in K, j \in J_k \tag{6f}$$

$$\sum_{j\in J_k}\sum_{i\in I}\sum_{b\in\mathcal{B}_i} x_{ibkj}(v_{ibkj} + XR_{ibkj}) \leq U_k, k \in K \tag{6g}$$

$$\sum_{k\in K}\sum_{j\in J_k} x_{ibkj}M_{ibkj} \leq D, i \in I,\ b \in \mathcal{B}_i \tag{6h}$$

$$y_{ik},\ x_{ibkj} \in \{0,1\} i \in I,\ b \in \mathcal{B}_i,\ k \in K, j \in J_k \tag{6i}$$

The objective function (6a) includes two terms, the cost of the master VMs and the cost of an extra VM. Constraints (6b) guarantee that each application is submitted to only one cloud in the federation. Constraints (6c) and (6e) ensures that for each application and each cloud only one instance type of the chosen cloud is selected. Constraints on the resource limits of each cloud are imposed in (6g). Moreover, the number of instances of each instance type at each cloud cannot exceed the maximum given by constraints (6f). Finally, constraints (6i) are the binary constraints. We will denote by y and x the variables $\{y_{ik}\}_{i\in I,\, k\in K}$ and $\{x_{ibjk}\}_{i\in I,\, b\in\mathcal{B}_i,\, k\in K,\, j\in J_k}$, respectively.

Acknowledgements The research of Carmen Galé has been funded by the Spanish Ministry of Science and Innovation 447 under grants PID2019-104263RB-C43 and by the Gobierno de Aragón under grant E41-20R. The research of J. Ezpeleta was supported in part by the Aragonese Government under Programa de Proyectos Estratégicos de Grupos de Investigación (DisCo research group, ref. T21-23R).

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

Abdi S, PourKarimi L, Ahmadi M et al. (2017) Cost minimization for deadline-constrained bag-of-tasks applications in federated hybrid clouds. Future Generat Comput Syst 71:113–128. https://doi.org/10.1016/j.future.2017.01.036

Abdi S, PourKarimi L, Ahmadi M et al. (2018) Cost minimization for bag-of-tasks workflows in a federation of clouds. The J Supercomput 74(6):2801–2822. https://doi.org/10.1007/s11227-018-2322-9

Abed-alguni BH, Alawad NA (2021) Distributed grey wolf optimizer for scheduling of workflow applications in cloud environments. Appl Soft Comput 102(107):113. https://doi.org/10.1016/j.asoc.2021.107113

Alkhanak EN, Lee SP, Khan SUR (2015) Cost-aware challenges for workflow scheduling approaches in cloud computing environments 50(C):3-21. https://doi.org/10.1016/j.future.2015.01.007,

Buyya R, Yeo CS, Venugopal S et al. (2009) Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Generat Comput Syst 25(6):599–616. https://doi.org/10.1016/j.future.2008.12.001

Chhabra A, Huang KC, Bacanin N et al. (2022) Optimizing bag-of-tasks scheduling on cloud data centers using hybrid swarm-intelligence meta-heuristic. The J Supercomput. https://doi.org/10.1007/s11227-021-04199-0

Díaz J, Entrialgo J, García M et al. (2017) Optimal allocation of virtual machines in multi-cloud environments with reserved and on-demand pricing. Future Generat Comput Syst 71:129–144. https://doi.org/10.1016/j.future.2017.02.004

Ehrgott M (2005) Multi Opt, 2nd edn. Springer, Berlin, Heidelberg

Foster I, Zhao Y, Raicu I, et al. (2008) Cloud computing and grid computing 360-degree compared. in: 2008 grid computing environments workshop, pp 1–10, https://doi.org/10.1109/GCE.2008.4738445

Genez T, Bittencourt L, Madeira E (2020) Time-discretization for speeding-up scheduling of deadline-constrained workflows in clouds. Future Generat Comput Syst 107:1116–1129. https://doi.org/10.1016/j.future.2017.07.061

Juve G, Chervenak A, Deelman E et al. (2013) Characterizing and profiling scientific workflows. Future Generat Comput Syst 29(3):682–692. https://doi.org/10.1016/j.future.2012.08.015

Karaja CAAASLM (2022) Efficient bi-level multi objective approach for budget-constrained dynamic bag-of-tasks scheduling problem in heterogeneous multi-cloud environment. Appl Intell. https://doi.org/10.1007/s10489-022-03942-1

Keivani A, Tapamo JR (2019) Task scheduling in cloud computing: A review. in: 2019 International conference on advances in big data, computing and data communication systems (icABCD), pp 1–6, https://doi.org/10.1109/ICABCD.2019.8851045

Kumar M, Sharma S, Goel A et al. (2019) A comprehensive survey for scheduling techniques in cloud computing. J Netw Comput Appl 143:1–33

Mann ZA (2015) Allocation of virtual machines in cloud data centers-a survey of problem models and optimization algorithms. ACM Comput Surv. https://doi.org/10.1145/2797211

Pham TP, Durillo JJ, Fahringer T (2020) Predicting workflow task execution time in the cloud using a two-stage machine learning approach. IEEE Transact Cloud Comput 8(1):256–268. https://doi.org/10.1109/TCC.2017.2732344

Teylo L, Arantes L, Sens P et al. (2021) Scheduling bag-of-tasks in clouds using spot and burstable virtual machines. IEEE Transact Cloud Comput. https://doi.org/10.1109/TCC.2021.3125426

Thai L, Varghese B, Barker A (2018) A survey and taxonomy of resource optimisation for executing bag-of-task applications on public clouds. Future Generat Comput Syst 82:1–11

Wang B, Song Y, Sun Y et al. (2016) Managing deadline-constrained bag-of-tasks jobs on hybrid clouds with closest deadline first scheduling. KSII Transact Int Inform Syst 10(7):2952–2971. https://doi.org/10.3837/tiis.2016.07.005

Wang B, Song Y, Cao J et al. (2019) Improving task scheduling with parallelism awareness in heterogeneous computational environments. Future Generat Comput Syst 94:419–429. https://doi.org/10.1016/j.future.2018.11.012

Yin L, Zhou J, Sun J (2022) A stochastic algorithm for scheduling bag-of-tasks applications on hybrid clouds under task duration variations. J Syst Softw 184(111):123

Authors and Affiliations

Luce Brotcorne[1] · Joaquín Ezpeleta[2] · Carmen Galé[3]

Luce Brotcorne
luce.brotcorne@inria.fr

Joaquín Ezpeleta
ezpeleta@unizar.es

[1] Lille Nord-Europe, INRIA, 40 Avenue Halley, 59000 Lille, France

[2] Department of Computer Science and Systems Engineering, Engineering Research Institute of Aragon, Universidad de Zaragoza, María de Luna 3, 50018 Zaragoza, Spain

[3] Statistical Methods Department, IUMA, Universidad de Zaragoza, María de Luna 3, 50018 Zaragoza, Spain