Future Generation Computer Systems 162 (2025) 107466

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Deadline-constrained security-aware workflow scheduling in hybrid cloud architecture

Somayeh Abdi [a,b,*], Mohammad Ashjaei [a], Saad Mubeen [a]

[a] Department of Networked and Embedded Systems, Mälardalen University, Västerås, Sweden
[b] Assistant Professor, Department of Computer Engineering, Eslam Abad-E-Gharb Branch, Islamic Azad University, Eslam Abad-E-Gharb, Iran

ARTICLE INFO

Keywords:
Hybrid cloud
Workflow scheduling
Mixed integer linear programming
Cost minimization
Sensitivity analysis

ABSTRACT

A hybrid cloud is an efficient solution to deal with the problem of insufficient resources of a private cloud when computing demands increase beyond its resource capacities. Cost-efficient workflow scheduling, considering security requirements and data dependency among tasks, is a prominent issue in the hybrid cloud. To address this problem, we propose a mathematical model that minimizes the monetary cost of executing a workflow and satisfies the security requirements of tasks under a deadline. The proposed model fulfills data dependency among tasks, and data transmission time is formulated with exact mathematical expressions. The derived model is a Mixed-integer linear programming problem. We evaluate the proposed model with real-world workflows over changes in the input variables of the model, such as the deadline and security requirements. This paper also presents a post-optimality analysis that investigates the stability of the assignment problem. The experimental results show that the proposed model minimizes the cost by decreasing inter-cloud communications for dependent tasks. However, the optimal solutions are affected by the limitations that are imposed by the problem constraints.

1. Introduction

Cloud computing has received considerable attention in the business and research communities because it offers many advantages, including the provision of pay-per-use, reliable, and elastic resources [1]. From the deployment perspective, cloud computing models are classified as public cloud, private cloud, hybrid cloud, community cloud [2,3] and multi-cloud [4]. A public cloud (e.g., Amazon EC2, GoGrid, Microsoft Azure, etc.) provides virtually unlimited resources that any subscriber can access. A private cloud provides an infrastructure that is owned by an organization with more security and privacy concerns, and its services are accessible only to its dedicated users.

The hybrid cloud model utilizes resources from both private and public clouds to address the problem of insufficient resources in a private cloud when it deals with the peak demand of its users' requests. Although using hybrid resources of a private cloud and a public cloud imposes some challenges, such as traffic routes, communication latency, etc, hybrid clouds offer the cost and scale benefits of public clouds, with the security and control of private clouds. With a hybrid cloud, organizations can scale to the public cloud and use the additional computing power when needed, resulting in cost savings [2,5–10]. Indeed, when a private cloud cannot satisfy the quality of service (QoS) requirements of tasks, such as a deadline, outsourcing tasks to a public cloud is an efficient solution to achieve the desired QoS and provide scalable services. Since security is commonly the main concern of an enterprise, using a hybrid cloud, tasks with high-security concerns execute in the private cloud, whereas tasks with low-security concerns execute in the public cloud.

Different applications in both scientific and industrial domains can be modeled as workflows [11]. A workflow refers to a set of dependent tasks that perform a dedicated function. Workflow scheduling is one of the prominent issues in cloud computing. It is a well-known NP-hard problem [12] and it aims to optimize resource allocation to complete tasks considering the data dependency between them and satisfying QoS requirements, such as a deadline or security constraints. Allocating proper resources for workflow execution, particularly, in a hybrid cloud has been a challenge. Such resource allocation has a significant impact on meeting QoS requirements and minimizing the monetary cost of workflow execution. This paper proposes a mathematical programming model to address the problem of security-aware workflow scheduling in a hybrid cloud model. The main objective is to minimize the monetary cost of executing a workflow under deadline and security constraints. The derived model is a Mixed-Integer Linear Programming (MILP) problem. Since both the objective and the constraints of the model are linear, it can be solved by efficient algorithms in a reasonable time [13].

* Corresponding author.
E-mail addresses: somayeh.abdi@mdu.se (S. Abdi), Mohammad.ashjaei@mdu.se (M. Ashjaei), saad.mubeen@mdu.se (S. Mubeen).

https://doi.org/10.1016/j.future.2024.07.044
Received 26 April 2023; Received in revised form 23 February 2024; Accepted 29 July 2024
Available online 31 July 2024

With the improvements in parallelization and computing power, the application of mathematical programming models has gained considerable attention in scientific and engineering fields to solve optimization problems [14,15]. Mathematical optimization is capable of generating a globally optimal solution to many real-world business problems [16–19]. With mathematical optimization, we can capture the key features, i.e., decision variables, constraints, and the objective function of our problems in an optimization model. Once an optimization model is formulated, it can be solved using optimization solvers, such as CPLEX[1] and Gurobi.[2] Mathematical solvers are used in many fields to solve complex optimization problems and make data-driven decisions. Their capability to find globally optimal solutions makes them powerful tools in modern problem-solving, providing the basis for optimal decisions. In addition, with a mathematical model, it is very easy to determine the infeasibility of an optimization problem. Solving a problem is deemed feasible if all constraints of the problem can be satisfied.

To propose and evaluate the mathematical model, we define the objective function and constraints of the model. Moreover, we perform a post-optimality analysis to study the behavior of the proposed model. Post-optimality analysis is a technique used in mathematical programming to study the sensitivity of the optimal solution to changes in the input variable of the model after the optimal solution has been obtained.

The main contributions of this paper are as follows.

- We formulate the workflow scheduling problem in a hybrid cloud model as an MILP problem. The proposed model considers both the execution and data transfer costs of a workflow and fulfills constraints related to the data dependency among tasks, the workflow deadline, and security requirements.
- We solve the proposed mathematical model using the CPLEX solver and develop it with OPL,[3] which is an algebraic modeling language.
- We evaluate the solution on several real-world workflows with respect to changes in input variables of the model such as the workflow deadline and security requirements. Moreover, we perform a post-optimality analysis to determine whether the optimal solution remains stable or changes significantly with the changes in the input variables of the model. Experimental results demonstrate that the proposed MILP model generally performs better than the three existing algorithms to optimize monetary costs and find feasible solutions.

The rest of this paper is organized as follows. Section 2, reviews the related works in the area of workflow scheduling in cloud computing. Section 3 introduces the system model of the hybrid cloud and workflow model. Section 4 presents the proposed mathematical model for the workflow scheduling problem. The experimental results of the model are reported in Section 5, and finally, Section 6 concludes the paper.

2. Related work

Workflow scheduling has been extensively studied in cloud computing. Optimizing makespan [20] and monetary cost [21] are the regular objectives of this research. Minimizing the monetary cost is a critical requirement in the cloud environment. Therefore, real-world workflows require scheduling that minimizes the monetary cost while meeting the QoS requirements. Various QoS requirements, such as deadline [22], security [23], budget [24], and reliability [25] are addressed in the related works. In this section, we review some related works that consider the deadline and security requirements for workflow scheduling in cloud computing.

Meeting the deadline of a workflow with a cost-optimization objective has been a challenge. To tackle this problem, deadline distribution is one of the methods that have been widely used. The work in [26] introduces the concept of prioritizing tasks in the workflow and assigning resources to the tasks based on their priorities. The main idea of this method is to propose an algorithm that ranks tasks in the workflow based on the minimum execution time and the longest path from the first task in the workflow. In the task assignment phase, it assigns each task to a node that allows completing the task at the earliest time. This work proposes two algorithms which are called the Heterogeneous Earliest-Finish-Time (HEFT) algorithm and the Critical-Path-on-a-Processor (CPOP) algorithm. The HEFT algorithm selects the task with the highest upward rank value, while the CPOP algorithm uses the summation of upward and downward rank values for prioritizing tasks. The work in [27] uses the concept of the deadline distribution method to schedule a deadline-constrained workflow in a hybrid cloud with the objective of cost minimization. It makes the initial schedule in the private cloud using the HEFT algorithm and checks whether the makespan is within the deadline of the workflow. If the makespan is higher than the deadline, it uses rescheduling policies and based on that schedules some tasks in the public cloud by finding the minimum monetary cost VMs that complete the execution of the tasks within their sub-deadlines. The work in [28] uses the deadline distribution method based on task prioritization. it proposes an algorithm that ranks tasks in the workflow based on the minimum execution time and the longest path from the first task in the workflow. Then, the workflow deadline is distributed to a sub-deadline for each task based on its rank. In the resource selection step, each task is assigned to the resource that meets its sub-deadline and minimizes the cost. The work in [22] proposes a deadline-constrained workflow scheduling in a single cloud based on the deadline distribution method. It proposes an ant colony algorithm that distributes the deadline to a sub-deadline for each task and selects a computation resource for each task. Generally, the efficiency of the deadline distribution methods depends on accurate deadline estimates for each task. Therefore, inaccurate estimates can lead to missed deadlines and reduced efficiency in the scheduling algorithm, which is a potential drawback of this method.

In addition to the deadline distribution method, meta-heuristic algorithms are also applied to address the workflow scheduling problem. These algorithms find an approximation of solutions in a reasonable time, but they do not guarantee to find the optimal solution. Furthermore, they cannot infer whether an optimization problem is infeasible, nor do they guarantee to find a solution if one exists. The work in [29] uses a genetic algorithm to schedule a workflow in a hybrid cloud. The proposed algorithm schedules a workflow under a specific deadline and a budget. It finds solutions that satisfy the workflow deadline and minimize the cost to meet the budget. The work in [30] proposes a genetic algorithm that allocates resources of a hybrid cloud to tasks in a workflow to minimize costs. It assumes each task of the workflow has a deadline. In the first phase of this algorithm, resources of the private cloud are selected for executing tasks to reduce task offloading to the public clouds. Then the tasks that cannot be scheduled in the private cloud are re-allocated to the public cloud. The work in [31] also uses the same idea for scheduling a workflow in the hybrid cloud with a cost-minimization objective. It makes an initial schedule in the private cloud using path clustering Heuristic algorithm [32] and creates clusters of tasks that are on the same path in the DAG. If the deadline of the workflow cannot be met, it selects a cluster of tasks in the workflow and assigns them to the more powerful VMs in the public cloud to satisfy the workflow deadline.

The work in [33] presents a combination of particle swarm optimization and genetic algorithm that reduces the data transmission for executing a workflow in the hybrid cloud. It considers security

[1] https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer

[2] https://www.gurobi.com/solutions/gurobi-optimizer, which produce the optimal solution.

[3] Optimization Programming Language

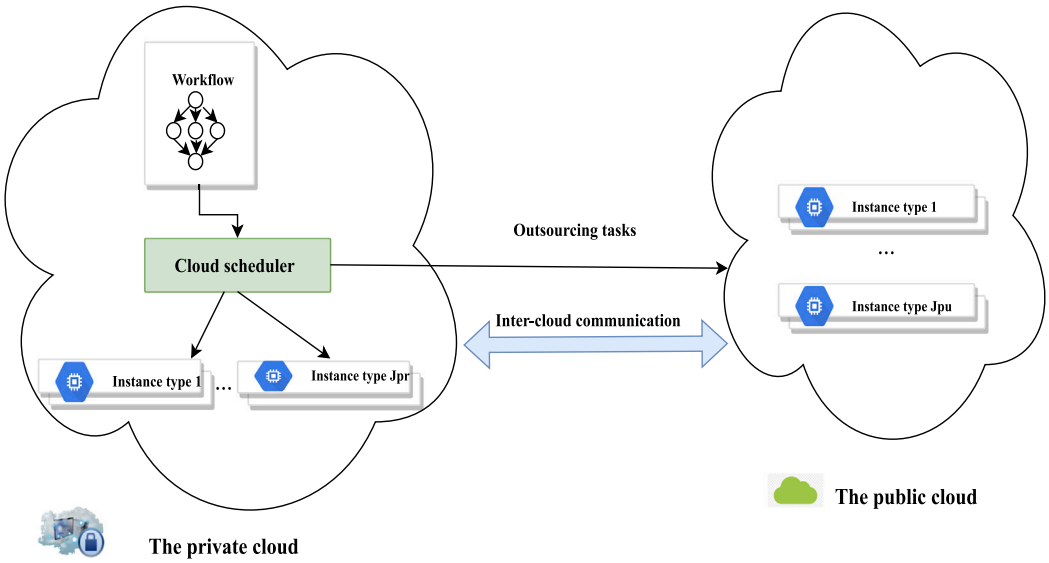


Fig. 1. Overview of the system model.

requirements for datasets, and the private dataset should be located in a private cloud data center. However, it does not consider workflow scheduling and how it could be cost-effective when combined with its approach. The work in [23] formulates the problem of task scheduling in a hybrid cloud as a non-linear programming model. It proposes a heuristic scheduling method that concerns the deadline and security requirements in a hybrid cloud. It assumes that each task has a deadline. The proposed algorithm maximizes the number of tasks that meet their deadlines and minimizes the cost with the basic idea of assigning tasks to VMs of the public cloud with the best price/performance ratio when the private cloud has insufficient resources. However, this work does not consider data dependency among tasks.

Applications of mathematical programming have already been of great importance in solving optimization problems. With the increase in computing power and the development of efficient algorithms used by solvers, the conditions for applying mathematical programming models have improved significantly. In this paper, the problem of workflow scheduling is formulated as an MILP model which can be solved in a reasonable time. The model fulfills data dependency among tasks in the workflow under a deadline. All possible cases for data transfer time among dependent tasks, in a hybrid cloud, are explicitly considered. In addition, the proposed model considers the execution cost of a workflow according to the common public cloud billing policies.

3. System model

The proposed hybrid-cloud model is depicted in Fig. 1, which comprises a private cloud and a public cloud. The public cloud provides virtually unlimited computing and storage resources and the private cloud provides limited resources. In this model, it is assumed that the scheduling algorithm is run on the cloud scheduler that resides in the private cloud. Workflows are submitted to the cloud scheduler and it decides to execute the tasks in a workflow on the private or the public cloud to optimize the objective function and satisfy the problem constraints. This decision is based on the workflow's requirements and the characteristics of available resources in the hybrid cloud.

In this section, the resource and workflow models are explained. The resource model describes the specification of virtual machine (VM) instances provided by the clouds. The workflow model describes the structure of a workflow and its requirements. Notations used in the system model are described in Table 1.

3.1. Resource model

Consider a hybrid cloud comprising a private cloud and a public cloud, we assume that our resource model is similar to Amazon's Elastic Compute Cloud (EC2)[4] and each cloud provides instance types with different prices and performance levels. Specifications of cloud providers and instance types are described in Table 1. A hybrid cloud at least consists of one private cloud and one public cloud. The number of clouds within the hybrid cloud is represented by m, and in Fig. 1, m is equal to 2. The notation CP indicates the set of cloud providers, and $CP = \{CP_{pu} \cup CP_{pr}\}$, where CP_k denotes cloud k. Thus, the notation CP_{pu} indicates the public cloud and CP_{pr} indicates the private cloud. Since we propose a security-aware task scheduling model, we consider a security tag for each cloud provider. In this paper, two security tags are considered; public security tag (S_{pu}) and private security tag (S_{pr}). The notation $RS_k \in \{S_{pu}, S_{pr}\}$ indicates the security tag of cloud k. Moreover, CP_k provides a set of instance types J_k with different specifications. Here, all VMs of the public cloud are considered public VMs, and all VMs of the private cloud are considered private VMs. The notation I_{kj} $(j \in J_k)$ indicates instance type j on cloud k. CP_k provides a pool of p VMs for instance type j indicated by the notation $L_j \implies \{1, 2, \ldots, p\}$. VMs of the same instance type are provisioned with the same characteristics. The notation vm_{kjl} indicates lth VM of instance type I_{kj}. Specification of instance type I_{kj} is as follows:

$$I_{kj} = \langle pr_{kj}, ram_{kj}, bw_{kj}, pf_{kj}, core_{kj} \rangle$$

The notation pr_{kj} denotes the price of running a VM of instance type I_{kj} per unit of time (hour). In this paper, we consider the cost of running VMs based on the common pricing model in cloud computing, called 'pay-as-you-go'. In this pricing model, the price of running an instance type is fixed, and it is billed by the hour. For instance, if a VM has been allocated to the tasks for less than one hour, it must be paid for the whole hour. The notation ram_{kj} denotes the RAM capacity of I_{kj}. The intra-cloud communication bandwidth of a VM of instance type I_{kj} is denoted by bw_{kj}. Instance type I_{kj} has $core_{kj}$ cores and the performance of its core(s) is denoted by pf_{kj}. The performance of an instance type can be quantified with several optional performance metrics, such as FLOPS, MIPS, CCU,[5] etc. In this work, we use the CCU metric as the assessment metric, and our proposed model is compatible

[4] https://aws.amazon.com/ec2/instance-types/
[5] Cloud Harmony Compute Unit (CCU)

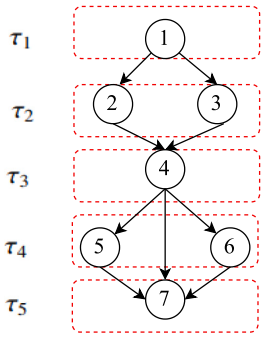


Fig. 2. A sample DAG representing a workflow.

with other performance metrics. CCU metric was developed by CloudHarmony[6] to provide a uniform metric for performance evaluation of instance types belonging to different cloud providers. A value of 1 CCU is approximately equal to 1 ECU[7] (e.g. 1 ECU = 1.0–1.2 GHz 2007 Xeon).

The notation bw_c indicates the communication bandwidth between the private and the public clouds. Moreover, it is assumed that the data transfer between VMs of the same cloud is free, and only data transfer between the private and the public cloud is charged. The notation dtc indicates the cost of transferring data per unit data size, between the private and the public clouds.

3.2. Workflow model

Applications in different fields are usually composed of dependent tasks [34]. These applications are called workflows and their structure can be modeled as a DAG (direct acyclic graph) $W = (T, E)$, depicted in Fig. 2. Workflows of different problems such as tomographic reconstruction [35], image analysis and processing [36], video analysis [37], and some scientific workflows, e.g., Epigenome, Montage, CyberShake, and LIGO [38,39] have a bag of tasks (BoT) model. Research on the real-world characteristics of parallel workloads indicates that between 34% and 89% of applications running on parallel systems have a BoT model [40,41]. Thus, in this work, we consider BoT workflows. A BoT workflow contains BoT stages in which each bag consists of parallel homogeneous tasks. The sample workflow contains five BoT stages, depicted in Fig. 2 by dashed boxes.

Since different workflows of wide-ranging domains are frequently executed by different research groups or companies on cluster or cloud systems, their characteristics (e.g., structure, task execution time, data dependencies, memory, input/output data size) are known beforehand [37,39,42,43]. For instance, the work in [39] developed a set of workflow profiling tools called wfprof to provide detailed information about the various computational tasks present in a workflow. This work provides the characteristics of workflows from diverse scientific applications, such as astronomy and bioinformatics. Moreover, information about different workflows from a wide range of domains, such as image processing, physics, astronomy, and bioinformatics, are available from the Pegasus workflow generator.[8]

The characteristics of a workflow are described in Table 1. It is assumed that workflow W consists of n BoTs, $T = \{\tau_1, \tau_2, \dots \tau_n\}$. The workflow has a deadline of Dl and all BoTs in the workflow should meet this deadline. Specification of BoT τ_i is as follows:

$$BoT\ \tau_i = \langle \mu_i, CS_i, TS_i, mem_i, pre_i \rangle$$

The notation μ_i indicates the number of tasks contained in BoT τ_i. If $\mu_i = 1$, BoT τ_i contains a single task; otherwise it contains μ_i parallel homogeneous tasks that can be executed at the same time, i.e., no data dependencies among them exist. In this work, we assume that when a task runs on a VM, it will finish without any interruption, i.e., there is no preemption enabled for the tasks. The computation size of BoT τ_i, denoted by CS_i indicates its execution time on a VM with the performance of 1 CCU. Tasks in a bag have the same security requirements considering they are homogeneous. Therefore, each BoT τ_i has a security requirement that is indicated by $TS_i \in \{S_{pu}, S_{pr}\}$. Here, S_{pu} indicates a public security tag, and S_{pr} indicates a private security tag. If $TS_i = S_{pu}$, BoT τ_i can be executed on VMs of the private or the public cloud. If $TS_i = S_{pr}$, BoT τ_i can only be executed on VMs of the private cloud. The minimum required ram for executing BoT τ_i is denoted by mem_i.

The data dependencies among BoTs are indicated by a dependency matrix $D \in \mathbb{R}_{|T|\times|T|}$. D is an upper triangular matrix and all elements in the diagonal are zero. Element d_{ij} $(i < j)$ represents data dependency between BoTs τ_i, τ_j. If $d_{ij} = 0$, there is no data dependency between τ_i, τ_j. Otherwise, d_{ij} indicates the size of data that must be transferred from BoT τ_i to τ_j, and BoT τ_j can start after BoT τ_i transmits its output data to it. Here, BoTs τ_i and τ_j are called parent (predecessor) and child (successor) BoTs, respectively. Based on the dependency matrix D, we identify the predecessor BoTs of each BoT, and the notation pre_i indicates the predecessor BoTs of BoT τ_i. BoT τ_1 is the first BoT of the workflow, and it does not have any predecessor BoT. The notation ids_1 indicates its input data size. The last BoT of the workflow does not have any successor BoT, and the notation ods_n denotes its output data size. Furthermore, it is assumed that the input data for τ_1 is stored in the distributed file system (DFS) on the private cloud, and after the completion of the last BoT the output data will be stored on DFS.

3.3. An illustrative example

To explain the resource model and workflow model, we present an illustrative example in this section. It is assumed that a public cloud (CP_{pu}) and a private cloud (CP_{pr}) participate in the hybrid cloud. The security tags of the clouds are $RS_{pr} = S_{pr}$ and $RS_{pu} = S_{pu}$. We assume that the private cloud provides two different instance types: $J_{pr} = \{I_1, I_2\}$, and the public cloud provides two different instance types $J_{pu} = \{I_1, I_2\}$. Each instance type has its specification as follows:

$$I_{pr1} = \langle pr_{pr1}, ram_{pr1}, bw_{pr1}, pf_{pr1}, core_{pr1} \rangle$$

$$I_{pr2} = \langle pr_{pr2}, ram_{pr2}, bw_{pr2}, pf_{pr2}, core_{pr2} \rangle$$

$$I_{pu1} = \langle pr_{pu1}, ram_{pu1}, bw_{pu1}, pf_{pu1}, core_{pu1} \rangle$$

$$I_{pu2} = \langle pr_{pu2}, ram_{pu2}, bw_{pu2}, pf_{pu2}, core_{pu2} \rangle$$

For instance, the notation vm_{pr1} indicates instance type 1th of the private cloud (CP_{pr}). The notation pr_{pr1} indicates the price of I_{pr1}, and ram_{pr1} indicates its memory capacity. The notations bw_{pr1}, pf_{pr1}, and $core_{pr1}$ indicate the communication bandwidth, performance, and the number of cores of I_{pr1}, respectively.

Moreover, the sample workflow, shown in Fig. 2, contains five BoTs, $T = \{\tau_1, \tau_2, \dots, \tau_5\}$. Each BoT has its specifications as follows:

$$BoT\ \tau_1 = \langle \mu_1, CS_1, TS_1, mem_1, pre_1 \rangle$$

$$BoT\ \tau_2 = \langle \mu_2, CS_2, TS_2, mem_2, pre_2 \rangle$$

$$BoT\ \tau_3 = \langle \mu_3, CS_3, TS_3, mem_3, pre_3 \rangle$$

$$BoT\ \tau_4 = \langle \mu_4, CS_4, TS_4, mem_4, pre_4 \rangle$$

[6] https://blog.cloudharmony.com/what-is-ecu-cpu-benchmarking-in-the-cloud/

[7] EC2 Compute Unit (ECU) is a unit that Amazon created to measure the relative CPU performance of an instance type.

[8] https://pegasus.isi.edu/

Table 1
Model parameters and their description.

Notation	Description
Notations of a workflow	
W	Workflow W.
$T = \{\tau_1, \tau_2, \dots \tau_n\}$	A set of n BoTs in workflow W.
$D \in \mathbb{R}_{\lvert T\rvert\times\lvert T\rvert}$	Element d_{ij} in dependency matrix D indicates the data size that must be transferred from BoT τ_i to τ_j .
μ_i	Number of tasks contained in BoT i.
ids_1	Input data size of BoT τ_1 in data unit size.
ods_n	Output data size of BoT τ_n in data unit size.
pre_i	Set of predecessor BoT(s) of BoT τ_i.
$TS_i \in \{S_{pu}, S_{pr}\}$	Security tag of BoT τ_i; S_{pu} indicates public security tag and S_{pr} indicates private security tag.
CS_i	The execution time of task(s) of BoT τ_i on a VM with performance of 1 CCU.
mem_i	The minimum required RAM for executing task(s) of BoT τ_i.
Dl	Deadline of the workflow.
Notation of cloud resources	
m	The number of participating clouds in the hybrid cloud.
$CP = \{CP_{pu} \cup CP_{pr}\}$	A set of cloud providers including the public cloud and the private cloud which participate in the hybrid cloud.
$RS_k \in \{S_{pu}, S_{pr}\}$	Security tag of cloud k ; S_{pu} indicates public security tag and S_{pr} indicates private security tag.
J_k	The set of instance types provided by CP_k.
I_{kj}	jth instance type provided by CP_k.
L_j	The set of VMs provided for jth instance type.
pr_{kj}	The fee of running a VM of instance type I_{kj} in dollar per hour.
$core_{kj}$	Number of computing cores of I_{kj}.
pf_{kj}	The performance of each core of I_{kj} in CCU metric.
bw_{kj}	Communication bandwidth of I_{kj}.
ram_{kj}	The memory capacity (RAM) of I_{kj}.
bw_c	The communication bandwidth between the private and the public clouds.
dtc	The price of data transfer between the private and the public clouds.

$$BoT\ \tau_5 = \langle \mu_5, CS_5, TS_5, mem_5, pre_5 \rangle$$

For instance, the notation μ_1 denotes the number of tasks contained in BoT τ_1, and the notation CS_1 indicates the computation size of BoT τ_1. The notation TS_1 denotes the security tag of BoT τ_1, and mem_1 indicates the minimum required RAM for executing BoT τ_1. The notation pre_1 indicates the set of predecessor BoT(s) of BoT τ_1.

$$D = \begin{bmatrix} 0 & d_{12} & 0 & 0 & 0 \\ 0 & 0 & d_{23} & 0 & 0 \\ 0 & 0 & 0 & d_{34} & d_{35} \\ 0 & 0 & 0 & 0 & d_{45} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (1)$$

Dependency matrix D, in Relation , indicates the data dependencies among BoTs in the sample workflow, shown in Fig. 2. Since this workflow has 5 BoTs, $D \in \mathbb{R}_{|5|\times|5|}$. Each element $d_{ti} \neq 0$ indicates data dependency between BoT τ_t, τ_i. For instance, element d_{12} indicates that there is data dependency between BoT τ_1 and τ_2. Indeed, d_{12} indicates the data size that must be transferred from BoT τ_1 to τ_2. Generally, the first BoT does not have any predecessor BoT, thus $pre_1 = \{\}$. Based on this matrix, the predecessor BoT(s) of each BoT are identified as follows:

$$pre_2 = \{\tau_1\},\ pre_3 = \{\tau_2\},\ pre_4 = \{\tau_3\},\ pre_5 = \{\tau_3, \tau_4\}$$

For instance, $pre_2 = \{\tau_1\}$ means that BoT τ_1 is predecessor of BoT τ_2.

4. Proposed mixed-integer linear programming (MILP) model

Mathematical programming has been widely used to solve decision-making problems. It is the process of finding the optimal solution to a problem within a set of constraints, where the solution is expressed in mathematical expressions. Mathematical programming models have four main elements: objective function, constraints, decision variables, and input data. The objective function is a mathematical expression that describes the goal of the optimization problem, which is either maximizing or minimizing some quantity. Constraints of a mathematical programming model represent the practical or physical requirements of the problem and they limit the values that the decision variables can take. In a decision-making problem, decision variables indicate the quantities that the decision-makers would like to determine such as doing some assignments. Values of decision variables are unknown in an optimization problem, and they are computed during the problem-solving procedure. The purpose of solving a mathematical programming model is to find values for decision variables that satisfy all constraints and optimize a specific objective function [14].

In this paper, we formulate the problem of workflow scheduling in a hybrid cloud as a mixed-integer linear programming model. It is assumed that workflow W is submitted to the cloud scheduler. As shown in Fig. 1, a scheduling algorithm is executed in the cloud scheduler, and it selects appropriate instance types from the hybrid cloud to execute the workflow. The proposed mathematical model fulfills constraints related to the workflow requirements. These requirements include considering data dependency among BoTs, security requirements of BoTs, required memory for executing BoTs, and the deadline of the workflow. Based on these constraints, the cloud scheduler selects proper instance types for BoTs to minimize the monetary cost of executing the workflow. The proposed model considers both the execution and data transfer costs of the workflow. Since the decision variables of the proposed model contain binary, non-negative integers, and real variables, the proposed model is an MILP model. Notations and descriptions of decision variables used in the proposed model are shown in Table 2. Moreover, input data for the mathematical model are described in Table 1. In this section, the formulation of the constraints and the objective function of the proposed mathematical model are presented in detail.

4.1. Task assignment

In this model, each BoT τ_i is assigned to a VM that is provided by the public or the private cloud in the hybrid cloud system. To formulate this constraint, we define decision variable $y_{ikjl} \in \{0, 1\}$ as follows:

$$y_{ikjl} = \begin{cases} 1 & if\ BoT\ \tau_i\ is\ assigned\ to\ vm_{kjl} \\ 0 & otherwise \end{cases} \quad (2)$$

The value of decision variable y_{ikjl} is delineated by Eq. (3) as follows:

Table 2
Decision variables and their description.

Notation	Description
$y_{ikjl} \in \{0,1\}$	1 iff BoT τ_i is assigned to vm_{kjl}.
$z_{ikj} \in \{0,1\}$	1 iff BoT τ_i is assigned to a VM of instance type I_{kj}.
$x_{ik} \in \{0,1\}$	1 iff BoT τ_i is assigned to cloud k.
$w_{itkjl} \in \{0,1\}$	1 iff BoTs τ_i and τ_t are assigned to vm_{kjl} ; otherwise 0.
$v_{it} \in \{0,1\}$	0 iff BoTs τ_i and τ_t are assigned to the same VM; otherwise 1.
$u_{itk} \in \{0,1\}$	1 iff BoTs τ_i and τ_t are assigned to cloud k; otherwise 0.
$c_{it} \in \{0,1\}$	0 iff BoTs τ_i and τ_t are assigned to the same cloud; otherwise 1.
$\theta_{it} \in \{0,1\}$	1 iff BoTs τ_i and τ_t run on different VMs in the same cloud ; otherwise 0.
$\gamma_{itkj} \in \{0,1\}$	1 iff $\theta_{it} = 1$ and $z_{ikj} = 1$; otherwise 0.
$ST_i \in \mathbb{R}^+$	The start time of BoT τ_i.
$RT_i \in \mathbb{R}^+$	The run time of BoT τ_i.
$M_i \in \mathbb{R}^+$	The completion time of BoT τ_i.
$DTT_{ti} \in \mathbb{R}^+$	Data transfer time between BoTs τ_t and τ_i.
$VS_{kjl} \in \mathbb{R}^+$	The start-timestap of running vm_{kjl}.
$VF_{kjl} \in \mathbb{R}^+$	The finish-timestap of running vm_{kjl}.
$TP_{kjl} \in \mathbb{Z}^+$	The period of running vm_{kjl} in unit time.

$$\sum_{k\in CP}\sum_{j\in J_k}\sum_{l\in L_j} y_{ikjl} = 1 \quad \forall i \in T \tag{3}$$

Eq. (3) states that each BoT can only be assigned to exactly one VM. If BoT τ_i is assigned to vm_{kjl} then decision variable y_{ikjl} takes value 1; otherwise, it is set to 0.

4.2. Minimum required RAM for executing tasks

Satisfying the minimum required RAM for executing BoTs is one of the constraints in our model. To formulate this constraint, decision variable $z_{ikj} \in \{0,1\}$ is defined as follows:

$$z_{ikj} = \begin{cases} 1 & if\ BoT\ \tau_i\ is\ assigned\ to\ a\ VM\ of\ instance\ type\ I_{kj} \\ 0 & otherwise \end{cases} \tag{4}$$

The value of decision variable z_{ikj} is delineated by Eq. (5). In this equation, the expression $\sum_{l\in L_j} y_{ikjl} = 1$, if BoT τ_i is assigned to a VM of instance type I_{kj}. Conversely, when this summation equals 0, it indicates that BoT τ_i is not assigned to any VM of instance type I_{kj}. As a direct consequence of this constraint, if $\sum_{l\in L_j} y_{ikjl} = 1$, the decision variable z_{ikj} takes a value of 1; otherwise, it is set to 0.

$$z_{ikj} = \sum_{l\in L_j} y_{ikjl} \quad \forall i \in T,\ \forall k \in CP,\ \forall j \in J_k \tag{5}$$

This constraint encapsulates the relationship between the decision variable z_{ikj} and the assignment of BoT τ_i to VMs of different instance types. Eq. (6) ensures that BoT τ_i is assigned to an instance type that satisfies the minimum required RAM for executing its task(s).

$$z_{ikj} \cdot mem_i \le ram_{kj} \quad \forall i \in T,\ \forall k \in CP,\ \forall j \in J_k \tag{6}$$

4.3. Security requirements

Since we propose a security-aware task scheduling model in the hybrid cloud, security tags are considered for BoTs and cloud resources. As mentioned in Section 3.1, we consider two security tags; public security tag (S_{pu}) and private security tag (S_{pr}), and $S_{pr} < S_{pu}$, where smaller security tag indicates that the BoT or cloud resource has higher security concern. Thus, we consider $S_{pr} = 1, S_{pu} = 2$, where $S_{pr} < S_{pu}$. To formulate the security constraint, we define decision variable $x_{ik} \in \{0,1\}$ as follows:

$$x_{ik} = \begin{cases} 1 & if\ BoT\ \tau_i\ is\ assigned\ to\ cloud\ k \\ 0 & otherwise \end{cases} \tag{7}$$

The value of decision variable x_{ik} is delineated by Eq. (8). In this equation, the expression $\sum_{j\in J_k}\sum_{l\in L_j} y_{ikjl} = 1$ if BoT τ_i is assigned to a VM in cloud k. Conversely, if this summation equals 0, it indicates that BoT τ_i is not assigned to any VM in cloud k. Therefore, if $\sum_{j\in J_k}\sum_{l\in L_j} y_{ikjl} = 1$, the decision variable x_{ik} is set to 1. This indicates that BoT τ_i is assigned to cloud k; otherwise, x_{ik} is set to 0, indicating that BoT τ_i is not assigned to cloud k.

$$x_{ik} = \sum_{j\in J_k}\sum_{l\in L_j} y_{ikjl} \quad \forall i \in T,\ \forall k \in CP \tag{8}$$

This constraint encapsulates the relationship between the decision variable x_{ik} and the allocation of BoT τ_i to the kth cloud. Eq. (9) ensures that the security requirement of BoTs is satisfied. Accordingly, BoTs with a public security tag can be executed in the public cloud or the private cloud, whereas private BoTs can only be executed in the private cloud.

$$x_{ik} \cdot RS_k \le TS_i \quad \forall i \in T,\ \forall k \in CP \tag{9}$$

If BoT τ_i is private then $TS_i = 1$, and according to Eq. (9), inequality $x_{ik} \cdot RS_k \le 1$ should be satisfied. Considering $RS_{pr} = 1, RS_{pu} = 2$, only x_{ipr} can take value 1, and it means that BoT τ_i can only be assigned to the private cloud. If BoT τ_i is public then $TS_i = 2$, and inequality $x_{ik} \cdot RS_k \le 2$ should be satisfied. In this case, either x_{ipr}, or x_{ipu} can take value 1 and it means that BoT τ_i can be assigned to either the public or the private cloud.

4.4. Data dependency among BoTs

As mentioned in Section 3.2, we consider a workflow consisting of n BoTs. There are data dependencies among BoTs, and the notation pre_i indicates the predecessor BoT(s) of BoT τ_i. To fulfill data dependency among BoTs in a workflow, each BoT starts only when all its predecessor BoTs have been completed and their output data have been sent to it. Eqs. (10)–(11) state the data dependencies among BoTs as follows:

$$ST_1 = 0 \tag{10}$$

$$ST_i = \max(M_t + DTT_{ti}) \quad \forall i \in T - \{\tau_1\},\ \forall t \in pre_i \tag{11}$$

Since the first BoT in the workflow does not have any predecessor BoT(s), according to Eq. (10), the start time of the first BoT is zero, i.e., $ST_1 = 0$. For other BoTs, a BoT can only start after it has received all the input data from its predecessor BoTs, according to Eq. (11). In this equation, the decision variable M_t indicates the completion time of BoT τ_t, and the decision variable DTT_{ti} indicates the data transfer time from BoT τ_t to τ_i. Eq. (12) determines the value of decision variable M_i as follows:

$$M_i \ge ST_i + RT_i \quad \forall i \in T \tag{12}$$

In this equation, the decision variable ST_i indicates the start time of BoT τ_i and the decision variable RT_i denotes the run time of BoT τ_i.

To compute the run time of BoT τ_i, it is assumed that its run time on a VM with performance 1 CCU is known in advance, and it is denoted

by CS_i in Table 1. Eq. (13) determines the run time of BoT τ_i on a VM of instance type I_{kj}.

$$RT_i \geq z_{ikj} \cdot \frac{CS_i}{pf_{kj} \cdot minimum(core_{kj}, \mu_i)} \quad \forall i \in T,\ \forall k \in CP,\ \forall j \in J_k \tag{13}$$

The formula involves the division of the computation size of BoT τ_i, i.e., CS_i by the product of the performance factor associated with the instance type I_{kj}, i.e., pf_{kj} and the number of tasks that can be executed in parallel on a VM of instance type I_{kj}, i.e., $minimum(core_{kj}, \mu_i)$. It should be noted that BoT τ_i contains μ_i homogeneous parallel tasks and instance type I_{kj} has $core_{kj}$ which pf_{kj} indicates the performance of each core of the instance type I_{kj}. If the number of cores of instance type I_{kj} is less than the number of parallel tasks in BoT τ_i then $core_{kj}$ tasks can be executed in parallel on a VM of the instance type I_{kj}. Conversely, when the number of tasks in BoT τ_i is less than the number of core then μ_i tasks can be executed in parallel on a VM of instance type I_{kj}. Therefore, the expression $minimum(core_{kj}, \mu_i)$ in the formula indicates the number of tasks in BoT τ_i that can be executed in parallel on a VM of instance type I_{kj}. This constraint imposes a positive run time of BoT τ_i, i.e., RT_i, if and only if BoT τ_i is assigned to a VM of instance type I_{kj}, i.e., $z_{ikj} = 1$.

Although the use of a hybrid cloud presents some challenges, such as traffic routing and DNS changes, we do not model the impact of these technical aspects and only model the data transfer time between dependent BoTs. To formulate data transfer time between BoT τ_i and its predecessor BoT τ_t, we consider three possible cases in Eq. (14). If BoT τ_i and τ_t run on the same VM then $DTT_{ti} = 0$. In the case that BoTs τ_i and τ_t run on the same cloud but different VMs, data transfer time is computed by considering the data transfer bandwidth of I_{kj} that BoT τ_t is assigned to it, and $DTT_{ti} = z_{tkj} \cdot (d_{ti}/bw_{kj})$. Here, d_{ti} indicates the output data size of BoT τ_t that must be transferred to τ_i. If BoTs τ_i and τ_t run on different clouds, $DTT_{ti} = d_{ti}/bw_c$, where bw_c denotes communication bandwidth between the private and public clouds.

$$DTT_{ti} = \begin{cases} 0 & v_{it} = 0 \\ z_{tkj} \cdot (d_{ti}/bw_{kj}) & \theta_{it} = 1 \\ d_{ti}/bw_c & c_{it} = 1 \end{cases} \tag{14}$$

To formulate DTT_{ti}, we define decision variables v_{it}, c_{it} and θ_{it}. Decision variable $v_{it} \in \{0, 1\}$ indicates whether BoTs τ_i and τ_t are assigned to the same VM or not as follows:

$$v_{it} = \begin{cases} 0 & if\ BoTs\ \tau_i\ and\ \tau_t\ are\ assigned\ to\ the\ same\ VM \\ 1 & otherwise \end{cases} \tag{15}$$

Eq. (16) determines the value of decision variable v_{it}. If BoTs τ_i and τ_t are assigned to lth VM of instance type I_{kj} then $y_{ikjl} = 1$, $y_{tkjl} = 1$, and according to this constraint v_{it} takes value 0; otherwise $v_{it} = 1$.

$$v_{it} = 1 - \sum_{k \in CP} \sum_{j \in J_k} \sum_{l \in L_j} y_{ikjl} \cdot y_{tkjl} \quad \forall i \in T - \{\tau_1\},\ \forall t \in pre_i \tag{16}$$

Due to the quadratic term $y_{ikjl} \cdot y_{tkjl}$, this constraint is non-linear. The use of non-linear constraints increases the computational complexity of the optimization model and the time required to solve it. Many non-linear problems cannot be solved in a reasonable time with existing solvers. Therefore, transformation and linearization techniques are used to replace certain non-linear equations or functions with an exact equivalent linear programming formulation to create valid inequalities. It should be noted that the linearization of the problem does not change the result of the problem, but only reduces the time required to solve it [44,45]. Therefore, we linearize non-linear constraints using common transformation and linearization techniques [44]. To linearize the non-linear constraint in Eqs. (16), we can use a binary variable w_{itkjl} to replace $y_{ikjl} \cdot y_{tkjl}$, where $w_{itkjl} = 1$ if and only if BoTs τ_i and τ_t are assigned to the same VM on a cloud. Eqs. (17)–(18) determine the value of decision variable w_{itkjl}, and Eq. (19) which is a linear constraint determines the value of decision variable v_{it}.

$$w_{itkjl} \geq y_{ikjl} + y_{tkjl} - 1 \tag{17}$$

$$w_{itkjl} \leq \frac{1}{2} \cdot (y_{ikjl} + y_{tkjl}) \tag{18}$$

$$\forall i \in T - \{\tau_1\},\ \forall t \in pre_i,\ \forall k \in CP,\ \forall j \in J_k,\ \forall l \in L_j$$

$$v_{it} = 1 - \sum_{k \in CP} \sum_{j \in J_k} \sum_{l \in L_j} w_{itkjl} \quad \forall i \in T - \{\tau_1\},\ \forall t \in pre_i \tag{19}$$

Decision variable $c_{it} \in \{0, 1\}$ indicates whether BoTs τ_i and τ_t run on the same cloud or not as follows:

$$c_{it} = \begin{cases} 0 & if\ BoTs\ \tau_i\ and\ \tau_t\ are\ assigned\ to\ the\ same\ cloud \\ 1 & otherwise \end{cases} \tag{20}$$

Eq. (21) determines the value of decision variable c_{it}. If BoTs τ_i and τ_t are assigned to cloud k then $x_{ik} = 1$, $x_{tk} = 1$, and according to this constraint c_{it} takes value 0; otherwise $c_{it} = 1$.

$$c_{it} = 1 - \sum_{k \in CP} x_{ik} \cdot x_{tk} \quad \forall i \in T - \{\tau_1\},\ \forall t \in pre_i \tag{21}$$

To linearize non-linear constraint in Eqs. (21), we can use a binary variable u_{itk} to replace $x_{ik} \cdot x_{tk}$, where $u_{itk} = 1$ if and only if BoTs τ_i and τ_t are assigned to the same cloud. Eqs. (22)–(23) determine the value of decision variable u_{itk}, and Eq. (24) which is a linear constraint determines the value of decision variable c_{it}.

$$u_{itk} \geq x_{ik} + x_{tk} - 1 \tag{22}$$

$$u_{itk} \leq \frac{1}{2} \cdot (x_{ik} + x_{tk}) \tag{23}$$

$$\forall i \in T - \{\tau_1\},\ \forall t \in pre_i,\ \forall k \in CP$$

$$c_{it} = 1 - \sum_{k \in CP} u_{itk} \quad \forall i \in T - \{\tau_1\},\ \forall t \in pre_i \tag{24}$$

We also define decision variable $\theta_{it} \in \{0, 1\}$ as follows:

$$\theta_{it} = \begin{cases} 1 & if\ BoTs\ \tau_i\ and\ \tau_t\ are\ assigned\ to \\ & different\ VMs\ in\ the\ same\ cloud \\ 0 & otherwise \end{cases} \tag{25}$$

Eq. (26) determines the value of decision variable θ_{it}. $\theta_{it} = 1$ if and only if BoTs τ_i and τ_t run on different VMs in the same cloud. In this case, decision variable $v_{it} = 1$ and $c_{it} = 0$ and decision variable θ_{it} takes value 1; otherwise it sets to 0.

$$\theta_{it} = v_{it} - c_{it} \tag{26}$$

$$\forall i \in T - \{\tau_1\},\ \forall t \in pre_i$$

Eq. (27) that is a non-linear constraint indicates data transfer time between BoT τ_i and its predecessor BoT τ_t. Based on the defined constraints, if τ_i and τ_t run on the same VM then values of decision variables θ_{it} and c_{it} attain 0, and $DTT_{ti} = 0$. In the case that τ_i and τ_t run on different clouds then decision variables $\theta_{it} = 0$, $c_{it} = 1$ and $DTT_{ti} = d_{ti}/bw_c$. If τ_i and τ_t run on different VMs of the same cloud, decision variables $\theta_{it} = 1$, $c_{it} = 0$ and $DTT_{ti} = z_{tkj} \cdot (d_{ti}/bw_{kj})$.

$$DTT_{ti} \geq \theta_{it} \cdot z_{tkj} \cdot \frac{d_{ti}}{bw_{kj}} + c_{it} \cdot \frac{d_{ti}}{bw_c} \tag{27}$$

$$\forall i \in T - \{\tau_1\},\ \forall t \in pre_i,\ \forall k \in CP,\ \forall j \in J_k$$

To linearize non-linear constraint in Eqs. (27), we can use a binary variable γ_{itkj} to replace $\theta_{it} \cdot .z_{tkj}$, where $\gamma_{itkj} = 1$ if and only if BoTs τ_i and τ_t are assigned to different VMs on the same cloud, and BoT τ_t is assigned to a VM of instance type I_{kj}. Eqs. (28)–(29) determine the value of decision variable γ_{itkj}. Eq. (30), a linear constraint, indicates data transfer time between BoTs τ_i and τ_t.

$$\gamma_{itkj} \geq \theta_{it} + z_{tkj} - 1 \tag{28}$$

$$\gamma_{itkj} \leq \frac{1}{2} \cdot (\theta_{it} + z_{tkj}) \tag{29}$$

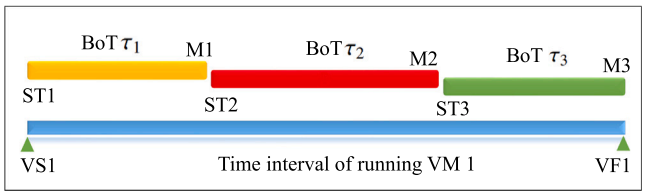


Fig. 3. Example of the time interval of running a VM.

$$DTT_{ti} \geq \gamma_{itkj} \cdot \frac{d_{ti}}{bw_{kj}} + c_{it} \cdot \frac{d_{ti}}{bw_c} \tag{30}$$
$$\forall i \in T - \{\tau_1\},\ \forall t \in pre_i,\ \forall k \in CP,\ \forall j \in J_k$$

4.5. Deadline of the workflow

Meeting the deadline of a workflow is one of the constraints of our model. Since a workflow comprises a set of n dependent BoTs, the completion time of the last BoT, BoT τ_n, must be less or equal to the deadline. Eq. (31) guarantees that the completion time of BoT τ_n is less or equal to the deadline.

$$M_n \leq Dl \tag{31}$$

4.6. Objective function

The objective function of the proposed model is to minimize the monetary cost of executing a workflow. This cost includes the cost of running VMs in dollars per hour and the data transfer costs. Eq. (32) indicates the objective function.

$$\text{Cost} = \text{minimize (Execution cost+ Data transfer cost)} \tag{32}$$

Subject to:

Constraint set A: (3), (5), (6), (8), (9), (10), (11), (12), (13), (17), (18), (19), (22), (23), (24), (26), (28), (29), (30), (31), (37), (38), (39), (40), (41)

where Eq. (33) indicates the cost of running VMs and Eq. (34) denotes the data transfer cost.

$$\text{Execution Cost} = \sum_{k \in CP} \sum_{j \in J_k} \sum_{l \in L_j} TP_{kjl} \cdot pr_{kj} \tag{33}$$

$$\text{Data transfer Cost} = \sum_{i \in T-\{t1\}} \sum_{t \in pre_i} c_{it} \cdot d_{ti} \cdot dtc + (x_{1pu} \cdot ids_1 + x_{npu} \cdot ods_n) \cdot dtc \tag{34}$$

As mentioned in Section 3.1, our model considers the cost of running VMs based on the 'pay-as-you-go' model which is the common pricing model in cloud computing. According to this pricing model, the cost of running a VM is charged based on the number of time intervals (hourly) that it has been allocated to the BoTs.

To compute the cost of running VMs, decision variables VS_{kjl}, VF_{kjl}, and TP_{kjl} are defined. If BoTs in the workflow are assigned to VM lth of instance type I_{kj}, its running time must be determined. To this aim, the start-timestamp (VS_{kjl}) and finish-timestamp (VF_{kjl}) of running VMs should be determined. Decision variable VS_{kjl} denotes the start time of the earliest BoTs that are assigned to the VM and VF_{kjl} denotes the completion time of the latest BoT assigned to the VM. For instance, we assume that BoT τ_1, τ_2, τ_3 are assigned to VM_1, as shown in Fig. 3. In this case, the start-timestamp of VM_1 is the start time of the first BoT that runs on it, $VS_1 \leq ST_1$, and the finish-timestamp of VM_1 is the completion time of the last BoT that runs on it, which is $VF_1 \geq M_3$.

Eqs. (35)–(36) determine the value of these decision variables.

$$VS_{kjl} \leq ST_i \cdot y_{ikjl} \tag{35}$$

$$VF_{kjl} \geq M_i \cdot y_{ikjl} \tag{36}$$
$$\forall i \in T,\ \forall k \in CP,\ \forall j \in J_k,\ \forall l \in L_j$$

Due to the quadratic terms $ST_i \cdot y_{ikjl}$ in Eq. (35), and $M_i \cdot y_{ikjl}$ in Eq. (36), these constraints are non-linear. In these equations, ST_i and M_i are continuous variables, and y_{ikjl} is a binary variable. It should be noted that to linearize these constraints, we need to identify an upper bound for the continuous decision variables ST_i and M_i. Since the start time and the completion time of each BoT is less than the deadline of the workflow, we consider the deadline as the upper bound for these decision variables; $0 \leq ST_i \leq Dl$, $0 \leq M_i \leq Dl$. Eqs. (37)–(40), which are linear constraints determine the value of these decision variables.

$$VS_{kjl} \leq Dl \cdot y_{ikjl} \tag{37}$$

$$VS_{kjl} \leq ST_i + Dl \cdot (1 - y_{ikjl}) \tag{38}$$

$$VF_{kjl} \leq Dl \cdot y_{ikjl} \tag{39}$$

$$VF_{kjl} \geq M_i + Dl \cdot (y_{ikjl} - 1) \tag{40}$$
$$\forall i \in T,\ \forall k \in CP,\ \forall j \in J_k,\ \forall l \in L_j$$

If VM lth of instance type I_{kj} is not allocated to any BoTs, then $VS_{kjl} = 0, VF_{kjl} = 0$. Otherwise, any optimal solution will automatically use the largest possible value of VS_{kjl} and the smallest possible value of VF_{kjl} since the execution cost is a decreasing function of the time interval of running VMs ($VF_{kjl} - VS_{kjl}$). Therefore, VS_{kjl} indicates the start time of the first BoT that runs on VM l of instance type I_{kj}, and VF_{kjl} indicates the completion time of the last BoT that runs on the VM.

Since the cost of running VMs is billed hourly, we need to determine the number of time intervals (hourly) of running VM lth of instance type I_{kj}. For instance, if a VM has been allocated to the BoTs in the workflow for 0.6 h, the time interval must be rounded to 1 h. Thus, we use an integer variable TP_{kjl} to round up the time interval. Eq. (41) determines the value of this decision variable.

$$TP_{kjl} \geq VF_{kjl} - VS_{kjl} \tag{41}$$
$$\forall i \in T, \forall k \in CP,\ \forall j \in J_k,\ \forall l \in L_j$$

Finally, the mathematical expression $pr_{kj} \cdot TP_{kjl}$, in Eq. (33), indicates the cost of running VMs in dollars per hour.

Eq. (34) indicates the data transfer cost of the workflow. As mentioned in Section 3.1, only data transfer between clouds is charged. The mathematical expression $c_{it} \cdot d_{ti} \cdot dtc$ in Eq. (34), indicates the data transfer costs among dependent BoTs. If dependent BoTs are assigned to VMs of the same cloud, data transfer among them is free. In other words, if BoT τ_i and its predecessor BoT τ_t run on the same cloud then $c_{it} = 0$, and data transfer is free of charge; otherwise, $c_{it} = 1$ and data transfer between τ_t and τ_i is charged. In this expression, d_{ti} indicates the size of data that must be transferred from BoT τ_t to τ_i, and dtc denotes the data transfer cost per unit data size. Furthermore, we consider the data transfer cost for the first and the last BoTs in the workflow. If the first BoT in the workflow is assigned to the public cloud, then $x_{1pu} = 1$, and its data transfer cost is charged, according to Eq. (34). Similarly, if the last BoT in the workflow is assigned to the public cloud, then $x_{npu} = 1$, and its data transfer cost is charged.

After solving the proposed mathematical model and determining the values of the decision variables, the optimal assignment of BoTs to VMs is obtained. Algorithm 1 shows the procedure for implementing the proposed mathematical model and using the results obtained from the model to schedule a workflow. The parameters and data defined in Table 1 are the input data of the algorithm. The first step is to define the decision variables of the problem, which are listed in Table 2. Then the objective function and constraints of the problem are defined according to Eq. (32) and constraint set A, which are stated in Section 4.6. After solving the optimization problem, the optimal values of decision variables are determined. To schedule the workflow W, the value of the decision variable y is used, so that if $y_{ikjl} = 1$, then the BoT τ_i is assigned

Algorithm 1 :Workflow scheduling algorithm

Input: Defined parameters for Workflow $W = (T, D)$ and cloud resources in Table 1
Output: The schedule Sch
Define the matrix of decision variables according to Table 2
Define the objective function according to Equation (32)
Define the *constraints set* A stated in Section 4.6
Solve the model
$Sch = \emptyset$
for all $i \in T, k \in CP, j \in J_k, l \in L_j$ **do**
  if $y_{ikjl} == 1$ **then**
    Assign BoT τ_i to vm_{kjl} at start time ST_i
    $Sch{=}Sch \cup < \tau_i, vm_{kjl}, ST_i, M_i >$
  end if
end for

Table 3
Instance types specifications provided by the hybrid cloud.

Cloud	Instance type	Price ($/hour)	CCU	Core(s)	Memory
Public Cloud	Type1	0.169	2.2	4	4
	Type2	0.338	2	4	8
	Type3	0.676	3	8	16
	Type4	1.352	3.5	16	32
	Type5	2.704	5	32	64
	Type6	3.186	6.5	36	72
	Type7	4.056	7.5	48	96
	Type8	5.408	8.5	64	128
	Type9	8.112	9	96	192
	Type10	11.328	10	128	256
	Type11	7.9	8	64	512
	Type12	9.576	8	64	768
Private Cloud	Type1	0.15	1.5	2	8
	Type2	0.45	2.5	8	32
	Type3	0.75	2.7	16	64
	Type4	3.7	4	32	192
	Type5	6.3	8	32	256
	Type6	7.1	8	32	512

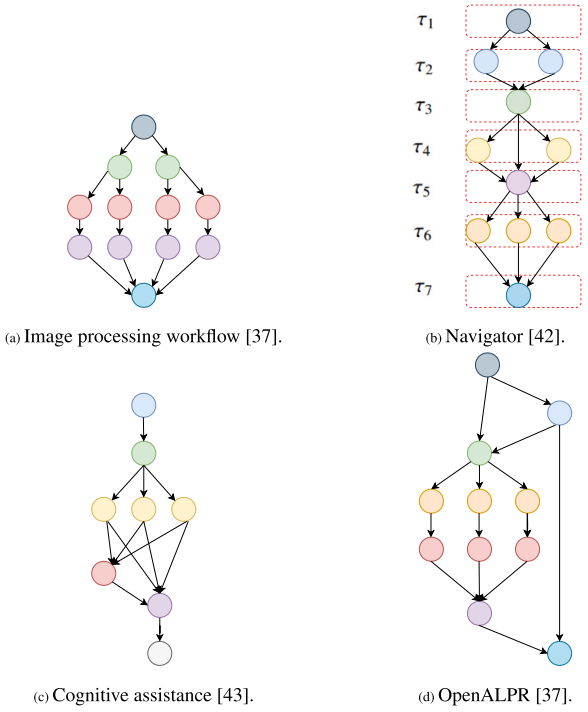
to vm_{kjl}. The value of the decision variable ST_i and M_i indicate the start time and completion time of BoT τ_i on the allocated VM. Thus, the assignment of BoT τ_i is represented as $\langle \tau_i, vm_{kjl}, ST_i, M_i \rangle$, and a schedule Sch contains the assignments of all BoTs in the workflow W.

5. Experimental results

To evaluate the performance of the proposed mathematical model, we use the CPLEX solver and implement the model as an OPL project in IBM ILOG CPLEX Optimization Studio version 22.1 with default settings. Specifically, experiments are performed on a PC with Intel Core i7 2.3 GHz, 32 GB RAM, and Windows 10 operating system. The average running time for solving the model is about 10 s.

5.1. Experimental settings

In the experiments, we use the specifications of compute-optimized instance types of Amazon EC2, and their specifications are listed in Table 3. Since there are costs associated with providing services to a private cloud, such as electricity and maintenance costs, it is assumed that running VMs of the private cloud is not completely free [46]. To investigate the behavior of the proposed mathematical model, we evaluate the optimal cost over changes in the input variables of the model such as the computation size of tasks, the data size of tasks, security requirements, and the workflow deadline.



(a) Image processing workflow [37]. (b) Navigator [42]. (c) Cognitive assistance [43]. (d) OpenALPR [37].

Fig. 4. The structure of some real-world workflows.

In the experiments, we use the structure of real-world workflows. The structures of these workflows are indicated in Fig. 4. These workflows have different structures, data-intensive (e.g. video analysis) and compute-intensive (e.g. image processing via CNNs[9]) characteristics. Since the workflows have various attributes, we normalized the execution cost of a workflow by dividing its actual execution cost by $cost_b$.

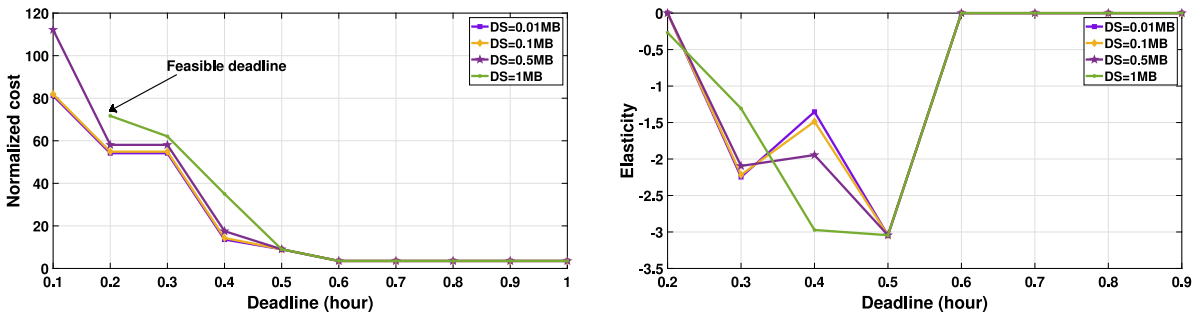
5.2. Post-optimality analysis

To fully analyze how the optimal solution of the proposed model is affected by the changes in the input variables, we perform a post-optimality analysis. This analysis indicates how sensitive the optimal solution is to changes in the input parameters or constraints. This is called sensitivity analysis. Sensitivity analysis in mathematical programming is a technique used to assess the stability of the optimal solution of a mathematical model with respect to changes in the model parameters. It helps to understand how changes in the model's input parameters affect the optimal value of the objective function [47]. In this paper, we use an Elasticity formula to perform sensitivity analysis. Elasticity indicates the percentage change in the optimal value of the objective function (e.g., y) on the percentage change in the input variable of the model (e.g., x) under a given set of assumptions [48], and it is expressed as follows:

$$E_{y,x} = \frac{\frac{\Delta y}{y}}{\frac{\Delta x}{x}} = \frac{dy}{dx} \cdot \frac{x}{y} \tag{42}$$

Overall, sensitivity analysis in mathematical programming is used to study the impact of variations in the model's parameters on the solution.

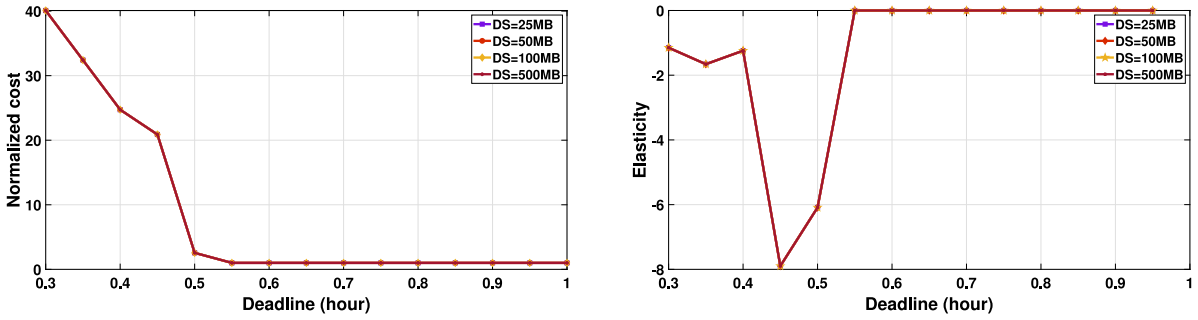
[9] Convolutional neural networks.



(a) Normalized cost over changes in the deadline for different DS.

(b) Elasticity of optimal cost over changes in the deadline for different DS.

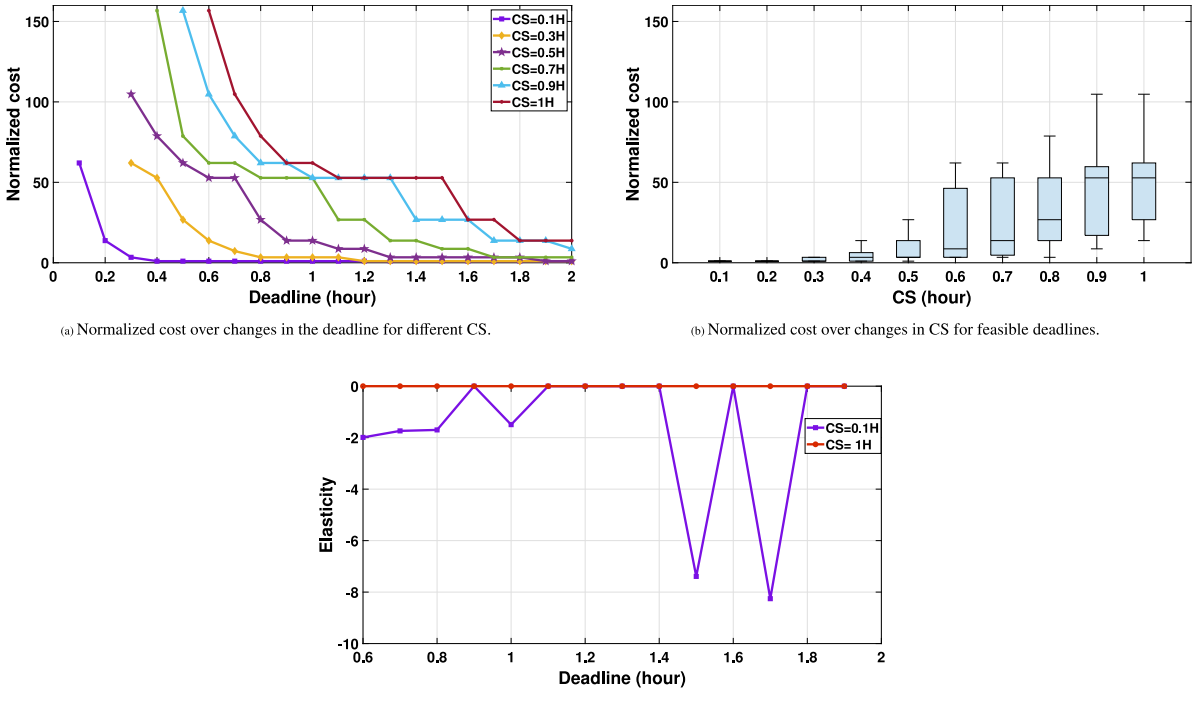
Fig. 5. Normalized cost and its elasticity over changes in the deadline and the data sizes (DS) for the image processing workflow.



(a) Normalized cost over changes in the deadline for different DS.

(b) Elasticity of optimal cost over changes in the deadline for different DS.

Fig. 6. Normalized cost and its elasticity over changes in the deadline and the data sizes (DS) for Navigator workflow.



(a) Normalized cost over changes in the deadline for different CS.

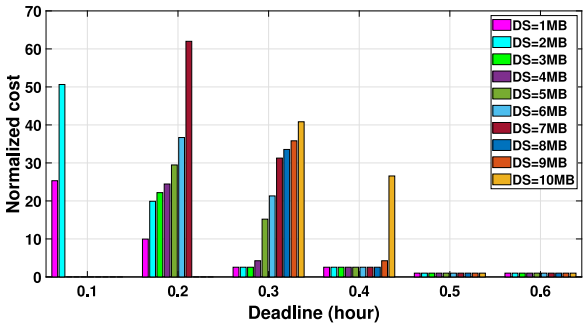
(b) Normalized cost over changes in CS for feasible deadlines.

(c) Elasticity of optimal cost over changes in the deadline.

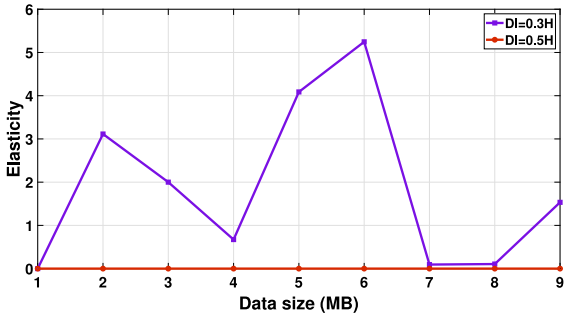
Fig. 7. Normalized cost and its elasticity over changes in the deadline and computation size (CS) for cognitive assistance workflow.

5.3. Results

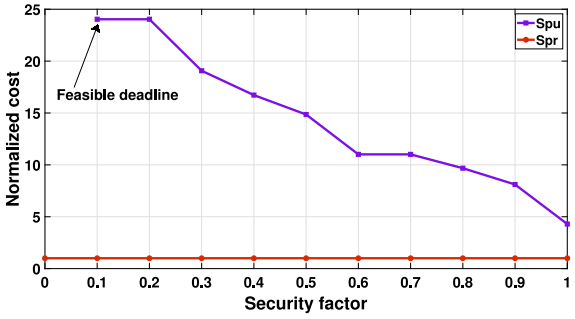
Figs. 5(a)–5(b) indicate the normalized cost and its sensitivity over changes in the deadline for an image processing workflow, respectively. This workflow is compute-intensive and its structure is depicted in Fig. 4(a). In a large-scale image processing workflow, there are some BoT stages for various operations such as resizing, cropping, filtering, and enhancement of images. The input data of this workflow is an image that is processed by the tasks in the workflow. To efficiently evaluate the performance of the proposed model, we fix the computation size of BoTs to 0.5 (hour) and increase the deadline from 0.1 to 1 (hour) with a step of 0.1. The image size is used as the input size of data



(a) Normalized cost over changes in the deadline for different DS.



(b) Elasticity of optimal cost over changes in DS.



(c) Normalized cost over changes in the security factor(α).

Fig. 8. Normalized cost and elasticity over changes in the data size (DS) and security factor for OpenALPR workflow.

for the BoTs and we performed the experiments with different values for the image size, namely DS = $\{0.01, 0.1, 0.5, 1\}$ MB. As it is shown in Fig. 5(a), generally the normalized cost decreases as the deadline increases. Since the BoTs in this workflow are compute-intensive, for tight deadlines, the model selects instance types from the public cloud which are more powerful and expensive to meet the deadline of the workflow. As the deadline increases, the model selects cost-efficient instance types that meet the deadline. For deadlines greater than Dl = 0.6, the model selects the most cost-efficient instance types and the cost does not change over changes in the deadline for different DS.

Moreover, we can observe that the costs vary for different data sizes when the deadline is tight, and the problem is not feasible for deadline = 0.1 (hour) when DS = 1 MB. When DS increases, the data transfer time increases. Therefore, the model selects more powerful instance types to meet the deadline of the workflow.

Fig. 5(b) depicts the sensitivity of the optimal cost to changes in the deadline for the image processing workflow. We perform this analysis for different DS for feasible deadlines $[0.2, \ldots, 1]$ (hour) with a step of 0.1. As this figure indicates, the value of elasticity changes significantly for tight deadlines. These changes in the elasticity, with increasing the deadline, indicate situations in which the model selects more cost-efficient instance types for the workflow. Moreover, when Dl = 0.6, the deadline is extended enough to select the most cost-efficient instance types from the private cloud. Thus, for deadlines greater than Dl = 0.6, the cost is not sensitive to the changes in the deadline. This analysis can help determine whether the optimal solution is stable to changes in the input variables, and it can also identify critical constraints that should be monitored.

To evaluate how the proposed model considers inter-cloud communications, we performed experiments for navigator workflow over changes in the deadline. This workflow is compute-intensive, and its structure is depicted in Fig. 4(b). This workflow processes a map and the map size can be different. The map is the output data of BoT τ_3 and input data for BoTs τ_4 and τ_5, depicted in Fig. 4(b). We performed the experiments with different values for the map size, namely DS $=\{25, 50, 100, 500\}$ MB. Figs. 6(a)–6(b) illustrate the normalized cost and its elasticity over changes in the deadline and DS for this workflow. As Fig. 6(a) indicates, overall, the normalized cost decreases when the deadline increases. For tight deadlines, powerful instance types from the public cloud which are more expensive are selected. For long deadlines, instance types from the private cloud that minimize the cost and meet the deadline are selected. However, the feasible deadlines and costs do not vary for different map sizes (DS). This is due to that our model submits dependent data-intensive BoTs to the same cloud to decrease the data transfer time and cost. In other words, our model decreases inter-cloud communications for dependent data-intensive BoTs. Fig. 6(b) shows that the optimal cost is sensitive to the deadline and, for long deadlines, the optimal solution is stable. Moreover, this analysis confirms that the optimal solution is not affected by the changes in the data size of inter-mediate BoTs.

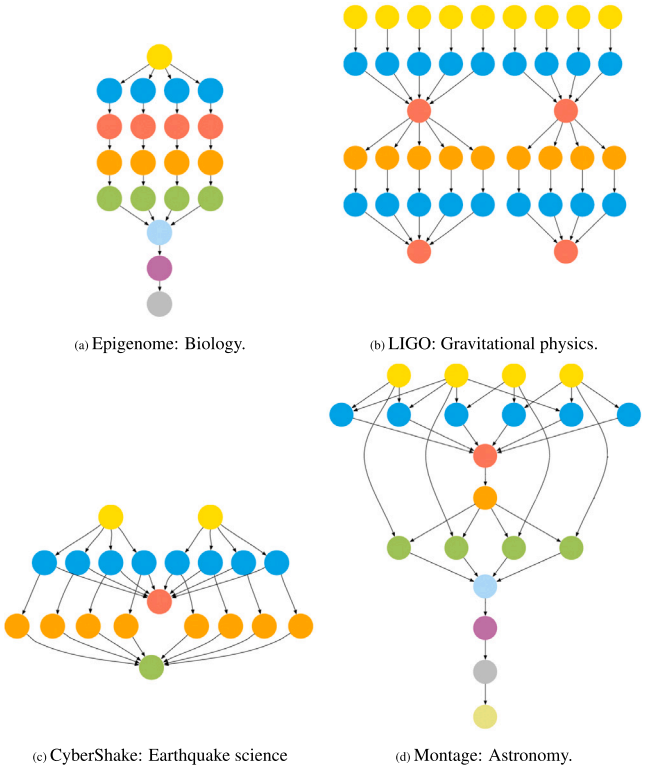
Figs. 7(a)–7(c) show the normalized cost and its elasticity over changes in the deadline for cognitive assistance workflow. This workflow is compute-intensive and its structure is depicted in Fig. 4(c). We performed experiments with different values for the computation size of BoTs. As mentioned in Section 5.1, the computation size of a BoT indicates its execution time on a VM with 1 CCU performance in unit time (hour). In this experiment, we fix the input data size of BoTs to 0.1 MB and increase the computation size from 0.1 to 1 (hour) with a step of 0.1.

As it can be seen in Fig. 7(a), the feasible deadlines vary for different CSs. For example, the problem is not feasible for deadlines less than Dl = 0.6 (hour) when CS = 1 (hour). Moreover, the cost decreases as the deadline increases, but the difference between normalized costs for small and large values of CSs, particularly for tight deadlines, is considerable. To examine how the normalized cost is dispersed for different values of CS, the results for deadline interval $[0.6, \ldots, 2]$ (hour) are indicated in Fig. 7(b). This boxchart indicates that the median of the normalized cost increases with the increase of CS. It is due to that when BoTs are more compute-intensive, more powerful instance types from the public cloud are allocated to the BoTs to meet the deadline. In contrast, for less compute-intensive BoTs, instance types from the private cloud are selected which leads to decreasing the execution and

data transfer costs. In Fig. 7(c), the elasticity of the optimal cost for small and large values of CS is compared. This comparison depicts that for compute-intensive BoTs, the elasticity fluctuates with the changes in the deadline. The significant changes in elasticity, with the increase of the deadline, indicate the situations in which the model selects more cost-efficient instance types to meet the deadline. When the computation size of BoTs is small, the optimal cost does not change over changes in the deadline, and the optimal solution is stable.

Figs. 8(a)–8(b) show the normalized cost and its elasticity over changes in the deadline and the data size for a vide analysis workflow, such as OpenALRP. A video analytics workflow typically involves processing and analyzing video content to extract useful information, and includes various stages of tasks such as frame Extraction, object detection, tracking, classification and recognition. This workflow is data-intensive and its structure is depicted in Fig. 4(d). We performed experiments for this workflow over changes in the data size, namely DS $= \{1, 2, \ldots, 10\}$ MB and we fixed the computation size of BoTs to 0.1 h. It can be seen from Fig. 8(a), when the deadline is too tight, only the problem is feasible for smaller values of DS. For example, when the deadline is Dl = 0.1 (hour), the problem is only feasible for executing the workflow with DS = 1 MB and DS = 2 MB. For tight deadlines, the model selects powerful instance types from the public cloud to meet the deadline. But as DS increases, due to increasing the data transfer time, the deadline cannot be met even though the model selects the most powerful instance types. As the deadline increases the problem is feasible for larger values of DS, and when Dl = 0.3 (hour) the problem is feasible for all considered DS. However, the normalized costs vary for different DS. For deadlines greater than Dl = 0.5 (hour), the cost does not change over changes in DS, because instance types from the private cloud are allocated to BoTs and the cost is not affected by data transmission. Moreover, we compared the elasticity of the model for Dl = 0.3 (hour) and Dl = 0.5 (hour) over changes in DS. When Dl = 0.3 (hour), the problem is feasible for all DS but the workflow is executed in the public cloud. When Dl = 0.5 (hour), the deadline is extended enough to execute the workflow in the private cloud. As expected, Fig. 8(b) indicates that when the workflow is executed in the private cloud, the cost is not affected by the changes in the data size, in contrast to executing the workflow in the public cloud.

We evaluated the impact of security requirements on the optimal cost for OpenALRP workflow. As mentioned in Section 4.3, when a BoT is private, it can only be executed in the private cloud. However, a public BoT can be executed on the private or public cloud. To perform this experiment, we considered two scenarios. In one scenario, all BoTs in the workflow can be executed in the private cloud, called *Spr* scenario. In the second scenario, the workflow is more compute-intensive and executing all BoTs in the workflow in the private cloud is not feasible to meet the deadline, and the model outsources some or all BoTs to the public cloud to meet the deadline, called *Spu* scenario. The normalized costs over changes in the security factor (α) for *Spr* and *Spu* scenarios are shown in Fig. 8(c). In this experiment, when the security factor $\alpha = 0$, all the BoTs are private while the security factor $\alpha = 1$ indicates that all BoTs are public, and for example, $\alpha = 0.1$ indicates that 10 percent of BoTs are public. As depicted in Fig. 8(c), in *Spr* scenario, the normalized cost is not affected by changes in the security factor α. Since in this scenario executing all the BoTs in the private cloud is feasible, the model selects appropriate instance types from the private cloud to execute the workflow. As mentioned, in *Spu* scenario, executing all BoTs in the private is not feasible. Therefore, when $\alpha = 0$, the normalized cost is not shown. In this scenario, the optimal cost is affected by the changes in the security factor, and with the increase of α, the cost decreases so that when all the BoTs are public, $\alpha = 1$, the optimal cost is minimum. Because, when $\alpha < 1$, instance types from both public and private clouds are rented to meet the security requirements of tasks. Moreover, the data transfer cost among dependent BoTs increases the monetary cost of executing the workflow. When $\alpha = 1$, the model selects appropriate instance types from the public cloud to meet the deadline and minimizes the data transfer cost among dependent BoTs.



(a) Epigenome: Biology. (b) LIGO: Gravitational physics. (c) CyberShake: Earthquake science (d) Montage: Astronomy.

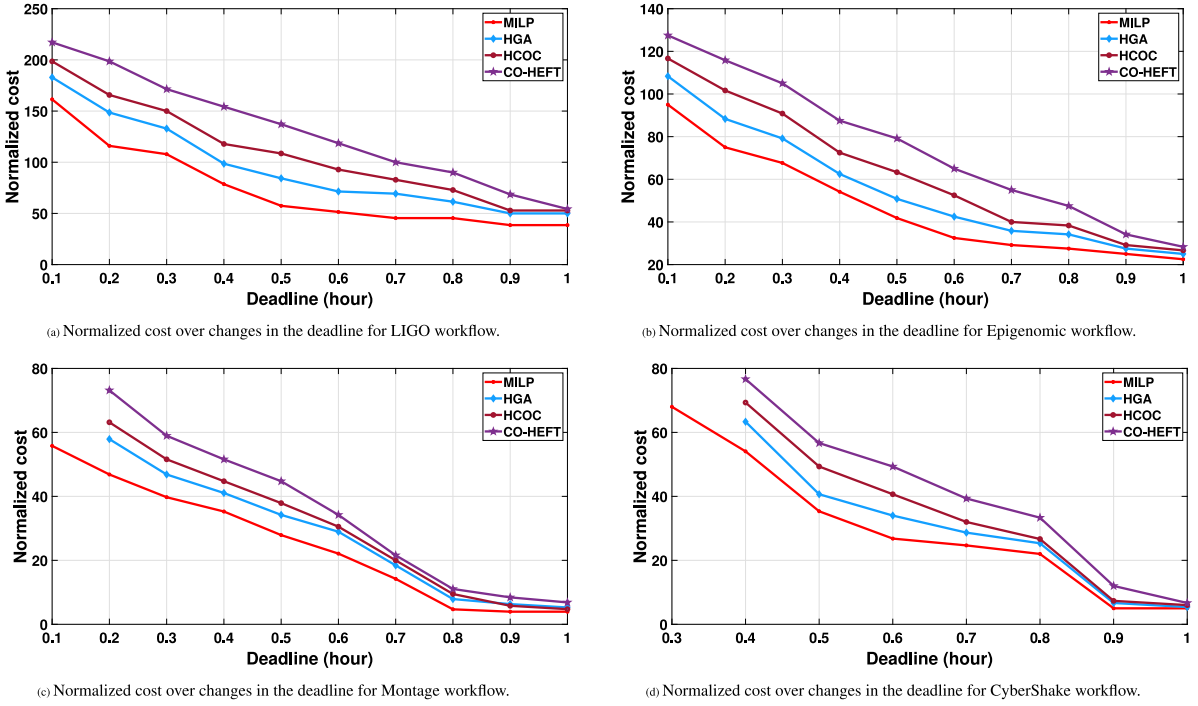
Fig. 9. The structure of some scientific workflows [39].

5.4. Cost comparison

In this section, we compare the performance of the proposed mathematical programming model (MILP) with CO-HEFT [27], HGA [29], and HCOC [31] approaches. We selected these algorithms since they schedule a deadline-constraint workflow with a cost-minimization objective in cloud computing. In the following, the algorithms CO-HEFT, HCOC, and HGA are briefly described.

- CO-HEFT uses the deadline distribution method to schedule a workflow in a hybrid cloud. It creates the initial schedule in the private cloud using the HEFT [26] algorithm and checks that the makespan is within the workflow deadline. If the makespan is greater than the deadline, it iteratively selects a task with the maximum missed deadline time and reschedules it to a VM in the public cloud that meets its sub-deadline.
- HCOC intends to schedule deadline-based workflows in a hybrid environment to minimize cost. It makes an initial schedule in the private cloud and creates clusters of tasks that are on the same path in the DAG. If the deadline of the workflow cannot be met, it selects a cluster of tasks in the workflow and assigns them to the more powerful VMs in the public cloud to satisfy the workflow deadline.
- HGA uses a genetic algorithm to schedule a workflow in a cloud environment. The proposed algorithm schedules a workflow under a specific deadline and a budget. It finds solutions that satisfy the workflow deadline and minimize the cost to meet the budget. However, we ignore the budget constraint since our problem focuses on cost minimization. We extended the genetic algorithm to schedule a workflow in a hybrid cloud.

For the comparison fairly, for all approaches, we assume that both VMs of the public and private clouds are charged and only the data transfer between clouds is charged. Since HGA and HCOC approaches evaluated their algorithms on scientific workflows, in the experiments, we used



(a) Normalized cost over changes in the deadline for LIGO workflow.

(b) Normalized cost over changes in the deadline for Epigenomic workflow.

(c) Normalized cost over changes in the deadline for Montage workflow.

(d) Normalized cost over changes in the deadline for CyberShake workflow.

Fig. 10. Normalized cost over changes in the deadline of a workflow.

the information of some scientific workflows that have the BoTs model, shown in Fig. 9. The specifications of these scientific workflows are available on the Pegasus workflow generator.[10]

The Epigenome is a compute-intensive scientific workflow used to map the epigenetic state of human cells on a genome-wide scale. This workflow contains several BoT stages to analyze the sequencing, aligning, and mapping of DNA, shown in Fig. 9(a). The LIGO is a compute-intensive scientific workflow used in the field of gravitational wave detection. This workflow contains six BoT stages for data preprocessing, signal search, parameter estimation, statistical significance assessment, follow-up observations, and data archiving [49], as shown in Fig. 9(b). The CyberShake is a data-intensive workflow used by the Southern California Earthquake Center (SCEC) to characterize earthquake hazards in a region. The Montage is a data-intensive workflow used in the field of astronomy and most tasks in it are not compute-intensive. In the experiments, we considered LIGo, CyberShake, Epigenome, and Montage workflows with 500 tasks.

Fig. 10 shows the cost achieved by the MILP, HGA, CO-HEFT, and HCOC approaches for the scientific workflows over changes in the deadline. To efficiently evaluate the cost comparison, we increase the deadline of workflows from 0.1 to 1 (hour) with a step of 0.1. As shown in Fig. 10, the normalized costs generally decrease as the deadline of the workflows increases. For tight deadlines, all approaches select instance types that are more powerful and expensive to meet the workflow deadline. When the deadline is extended, the solutions obtained by the different approaches give results that are almost close to each other. In general, MILP outperforms the other algorithms, especially for tighter deadlines, because it finds globally optimal solutions and minimizes both the cost of renting VMs and the cost of data transfer between dependent BoTs. HGA achieves the lowest cost after MILP and outperforms HCOC. CO-HEFT achieves the highest execution costs because it distributes the deadline of a workflow to sub-deadlines for tasks and it aims for a fast schedule that meets the tasks' sub-deadlines. However, due to the different workflow structures and characteristics, the execution costs of the workflows by the mentioned approaches are different. For Epigenome and LIGO, all the mentioned approaches find feasible solutions with different deadlines. By contrast, for Montage and CyberShake, only the proposed MILP model obtains feasible solutions when the deadline is very tight. For example, for CyberShake workflow, the problem is feasible for deadlines greater than 0.3 h, and when the deadline is 0.3 h, only MILP finds a feasible solution. Therefore, we can see that MILP performs better than the other approaches in terms of monetary cost reduction and finding feasible solutions.

6. Conclusion

This paper studies the problem of resource allocation in the hybrid cloud, in which BoTs in a workflow are assigned to instance types with the cost-minimizing objective. The main contribution of this paper is proposing a mixed-integer linear programming model for the assignment problem. The proposed model is formulated with three key insights: (1) considering QoS requirements including the workflow deadline and security requirements of BoTs; (2) formulating data transfer time and cost among dependent BoTs which leads to reducing inter-cloud and inter-VM communications; (3) considering the cost of workflow execution based on the common pricing model in cloud computing. The proposed model was solved with the CPLEX solver and the sensitivity analysis was performed to study the behavior of the model over changes in the input variables such as the security requirements, and the workflow deadline. Experimental results show that the proposed model can efficiently deal with the data dependency among BoTs, and applying the Elasticity formula as a stability criterion shows that the optimal decisions in the private cloud are more stable than those in the public cloud. The experimental results show that the proposed MILP model outperforms the three existing algorithms for workflow scheduling in hybrid clouds in terms of monetary cost reduction and finding feasible solutions.

For future work, we are going to extend our model to consider workflow scheduling in edge-cloud computing, in which a computing continuum from edge servers to a hybrid cloud is combined to satisfy the QoS requirements of tasks. Moreover, optimization problems with inexact data can be used to consider variability in the parameters of workflows and virtual machines.

[10] https://pegasus.isi.edu/

CRediT authorship contribution statement

Somayeh Abdi: Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Validation, Visualization, Writing – original draft. **Mohammad Ashjaei:** Conceptualization, Investigation, Supervision, Writing – review & editing. **Saad Mubeen:** Investigation, Supervision, Writing – review & editing.

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.

Data availability

Data will be made available on request.

Acknowledgments

The work in this paper is supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA) through the AORTA projects and KKS foundation through the project SEINE. Also, the work is supported by XPRES project.

References

[1] L. Qian, Z. Luo, Y. Du, L. Guo, Cloud computing: An overview, in: IEEE International Conference on Cloud Computing, Springer, 2009, pp. 626–631.
[2] S. Goyal, Public vs private vs hybrid vs community-cloud computing: a critical review, Int. J. Comput. Network Inf. Secur. 6 (3) (2014) 20–29.
[3] B.P. Rimal, E. Choi, I. Lumb, A taxonomy and survey of cloud computing systems, in: 2009 Fifth International Joint Conference on INC, IMS and IDC, Ieee, 2009, pp. 44–51.
[4] A.N. Toosi, R.N. Calheiros, R. Buyya, Interconnected cloud computing environments: Challenges, taxonomy, and survey, ACM Comput. Surv. 47 (1) (2014) 1–47.
[5] Q. Li, Z.-y. Wang, W.-h. Li, J. Li, C. Wang, R. y. Du, Applications integration in a hybrid cloud computing environment: Modelling and platform, Enterprise Inf. Syst. 7 (3) (2013) 237–271.
[6] A. Srinivasan, M.A. Quadir, V. Vijayakumar, Era of cloud computing: A new insight to hybrid cloud, Procedia Comput. Sci. 50 (2015) 42–51.
[7] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, Proactive workload management in hybrid cloud computing, IEEE Trans. Netw. Serv. Manag. 11 (1) (2014) 90–100.
[8] J. Weinman, Hybrid cloud economics, IEEE Cloud Comput. 3 (1) (2016) 18–22.
[9] R. Balasubramanian, M. Aramudhan, Security issues: public vs private vs hybrid cloud computing, Int. J. Comput. Appl. 55 (13) (2012) 00.
[10] R. Sujay, Hybrid cloud: A new era, Int. J. Comput. Sci. Technol. 2 (2) (2011) 323–326.
[11] W. Van Der Aalst, K.M. Van Hee, K. van Hee, Workflow Management: Models, Methods, and Systems, MIT Press, 2004.
[12] L.P. Michael, Scheduling: Theory, Algorithms, and Systems, Springer, 2018.
[13] H.A. Taha, Integer Programming: Theory, Applications, and Computations, Academic Press, 2014.
[14] E. Castillo, A.J. Conejo, P. Pedregal, R. Garcia, N. Alguacil, Building and Solving Mathematical Programming Models in Engineering and Science, John Wiley & Sons, 2011.
[15] S. Abdi, M. Ashjaei, S. Mubeen, Cognitive and time predictable task scheduling in edge-cloud federation, in: 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation, ETFA, IEEE, 2022, pp. 1–4.
[16] O.V. Bisikalo, V.V. Kovtun, O.V. Kovtun, O.M. Danylchuk, Mathematical modeling of the availability of the information system for critical use to optimize control of its communication capabilities, Int. J. Sensors Wirel. Commun. Control 11 (5) (2021) 505–517.
[17] N. Buras, An application of mathematical programming in planning surface water storage 1, JAWRA J. Am. Water Resour. Assoc. 21 (6) (1985) 1013–1020.
[18] I.E. Grossmann, G. Guillén-Gosálbez, Scope for the application of mathematical programming techniques in the synthesis and planning of sustainable processes, Comput. Chem. Eng. 34 (9) (2010) 1365–1376.
[19] J. Samuels, Opportunity costing: an application of mathematical programming, J. Account. Res. (1965) 182–191.
[20] J. Sun, L. Yin, M. Zou, Y. Zhang, T. Zhang, J. Zhou, Makespan-minimization workflow scheduling for complex networks with social groups in edge computing, J. Syst. Archit. 108 (2020) 101799.
[21] J. Lei, Q. Wu, J. Xu, Privacy and security-aware workflow scheduling in a hybrid cloud, Future Gener. Comput. Syst. 131 (2022) 269–278.
[22] Q. Wu, F. Ishikawa, Q. Zhu, Y. Xia, J. Wen, Deadline-constrained cost optimization approaches for workflow scheduling in clouds, IEEE Trans. Parallel Distrib. Syst. 28 (12) (2017) 3401–3412.
[23] B. Wang, C. Wang, W. Huang, Y. Song, X. Qin, Security-aware task scheduling with deadline constraints on heterogeneous hybrid clouds, J. Parallel Distrib. Comput. 153 (2021) 15–28.
[24] N. Rizvi, D. Ramesh, Fair budget constrained workflow scheduling approach for heterogeneous clouds, Cluster Comput. 23 (4) (2020) 3185–3201.
[25] Q.-H. Zhu, H. Tang, J.-J. Huang, Y. Hou, Task scheduling for multi-cloud computing subject to security and reliability constraints, IEEE/CAA J. Autom. Sin. 8 (4) (2021) 848–865.
[26] 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.
[27] N. Chopra, S. Singh, Heft based workflow scheduling algorithm for cost optimization within deadline in hybrid clouds, in: 2013 Fourth International Conference on Computing, Communications and Networking Technologies, ICCCNT, IEEE, 2013, pp. 1–6.
[28] S. Abrishami, M. Naghibzadeh, D.H. Epema, Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds, Future Gener. Comput. Syst. 29 (1) (2013) 158–169.
[29] H. Aziza, S. Krichen, A hybrid genetic algorithm for scientific workflow scheduling in cloud environment, Neural Comput. Appl. 32 (2020) 15263–15278.
[30] A. Pasdar, Y.C. Lee, K. Almi'ani, Hybrid scheduling for scientific workflows on hybrid clouds, Comput. Netw. 181 (2020) 107438.
[31] L.F. Bittencourt, E.R.M. Madeira, Hcoc: a cost optimization algorithm for workflow scheduling in hybrid clouds, J. Internet Services Appl. 2 (2011) 207–227.
[32] L.F. Bittencourt, E.R. Madeira, A performance-oriented adaptive scheduler for dependent tasks on grids, Concurr. Comput.: Pract. Exper. 20 (9) (2008) 1029–1049.
[33] 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, IEEE, 2018, pp. 556–563.
[34] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, K. Vahi, Characterization of scientific workflows, in: 2008 Third Workshop on Workflows in Support of Large-Scale Science, IEEE, 2008, pp. 1–10.
[35] S. Smallen, H. Casanova, F. Berman, Applying scheduling and tuning to online parallel tomography, in: Proceedings of the 2001 ACM/IEEE Conference on Supercomputing, 2001, p. 12.
[36] H. Wu, W. Knottenbelt, K. Wolter, Y. Sun, An optimal offloading partitioning algorithm in mobile cloud computing, in: International Conference on Quantitative Evaluation of Systems, Springer, 2016, pp. 311–328.
[37] V. De Maio, I. Brandic, First hop mobile offloading of dag computations, in: 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID, IEEE, 2018, pp. 83–92.
[38] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P.J. Maechling, R. Mayani, W. Chen, R.F. Da Silva, M. Livny, et al., Pegasus, a workflow management system for science automation, Future Gener. Comput. Syst. 46 (2015) 17–35.
[39] 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.
[40] T.N. Minh, T. Nam, D.H. Epema, Parallel workload modeling with realistic characteristics, IEEE Trans. Parallel Distrib. Syst. 25 (8) (2013) 2138–2148.
[41] S. Abdi, L. PourKarimi, M. Ahmadi, F. Zargari, Cost minimization for deadline-constrained bag-of-tasks applications in federated hybrid clouds, Future Gener. Comput. Syst. 71 (2017) 113–128.
[42] L. Aceto, A. Morichetta, F. Tiezzi, Decision support for mobile cloud computing applications via model checking, in: 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, IEEE, 2015, pp. 199–204.
[43] F.A. Salaht, F. Desprez, A. Lebre, An overview of service placement problem in fog and edge computing, ACM Comput. Surv. 53 (3) (2020) 1–35.
[44] M. Asghari, A.M. Fathollahi-Fard, S. Mirzapour Al-e hashem, M.A. Dulebenets, Transformation and linearization techniques in optimization: A state-of-the-art survey, Mathematics 10 (2) (2022) 283.
[45] R.G. Jeroslow, J.K. Lowe, Modelling with Integer Variables, Springer, 1984.
[46] R. Birke, A. Podzimek, L.Y. Chen, E. Smirni, Virtualization in the private cloud: State of the practice, IEEE Trans. Netw. Serv. Manag. 13 (3) (2016) 608–621.
[47] S. French, Mathematical programming approaches to sensitivity calculations in decision analysis, J. Oper. Res. Soc. 43 (1992) 813–819.
[48] P.R. Thimmapuram, J. Kim, A. Botterud, Y. Nam, Modeling and simulation of price elasticity of demand using an agent-based model, in: 2010 Innovative Smart Grid Technologies, ISGT, IEEE, 2010, pp. 1–8.
[49] D.A. Brown, P.R. Brady, A. Dietz, J. Cao, B. Johnson, J. McNabb, A case study on the use of workflow technologies for scientific analysis: Gravitational wave data analysis, in: Workflows for E-Science: Scientific Workflows for Grids, 2007, pp. 39–59.

Somayeh Abdi is a PostDoc researcher in the Department of Network and Embedded Systems, Mälardalen University, Vasterås, Sweden. She received a B.S. degree in Software Engineering from Razi University in Kermanshah, Iran, an M.S. and a Ph.D. degree in the same course from Science and Research Branch, Islamic Azad University, Tehran, Iran. Her current research interests include mathematical programming, Edge-Cloud Computing, and workflow scheduling.

Mohammad Ashjaei is a senior lecturer in the Complex Real-Time Systems (CORE) and the Heterogeneous Systems - Hardware Software Co-design (HERO) research groups at Mälardalen University in Sweden. Mohammad has received his Ph.D. degree in Computer Science in November 2016 from Mälardalen University. His main research interests include real-time systems, real-time distributed systems, scheduling algorithms on networks and processors, schedulability analysis techniques, resource reservation and reconfiguration mechanisms for real-time networks. He is also giving lectures on various topics related to embedded systems and data communication networks. He is a PC member and referee for several international conferences and journals, including IEEE Transactions on Industrial Informatics (TII), IEEE Transactions on Industrial Electronics (TIE), Elsevier's Journal of Systems Architecture, IEEE Transactions on Network and Services, Journal of Cloud Computing, and ACM Computing Surveys. He has organized and chaired several special sessions and workshops at the international conferences such as ETFA.

Saad Mubeen is a Professor at Mälardalen University, Sweden. He has previously worked in the vehicle industry as a Senior Software Engineer at Arcticus Systems and as a Consultant for Volvo Construction Equipment, Sweden. He received his Ph.D. in Computer Science and Engineering from Mälardalen University Sweden in June 2014. He is a Senior Member of IEEE and a Co-chair of the Subcommittee on In-vehicle Embedded Systems within the IEEE IES Technical Committee on Factory Automation. His research focus is on model- and component-based development of predictable embedded software, modeling and timing analysis of in-vehicle communication, and end-to-end timing analysis of distributed embedded systems. Within this context, he has co-authored over 150 publications in peer-reviewed international journals, conferences and workshops. He has received several awards, including the IEEE Software Best Paper Award in 2017. He is a PC member and referee for several international conferences and journals respectively. He is a guest editor of Elsevier's Journal of Systems Architecture and Microprocessors and Microsystems, IEEE Transactions on Industrial Informatics (TII), ACM SIGBED Review, and Springer's Computing journal. He has organized and chaired several special sessions, tracks, and workshops at the international conferences such as IEEE's IECON, ICIT, ETFA, ISORC, DIVERSE, CRTS, to mention a few. For more information see http://www.es.mdh.se/staff/280-Saa_Mubeen.