Communications in Combinatorics and Optimization
Vol. xx, No. x (xxxx), pp. 1-23
https://doi.org/10.22049/cco.2026.30429.2465

CCO
Commun. Comb. Optim.

Research Article

Bi-objective resource allocation for cloud service providers: A dichotomic approach to pareto optimization

A.H.S. Razavi[1,†], M. Zaferanieh[1,*], S. Sobati-Moghadam[2,3]

[1] Department of Mathematics and Computer Sciences, Hakim Sabzevari University, Iran
[†] a.h.s.razavi@hsu.ac.ir
[*] m.zaferanieh@hsu.ac.ir

[2] Department of Computer, Faculty of Electrical and Computer Engineering, Hakim Sabzevari University, Iran
s.sobati@hsu.ac.ir

[3] Computer Engineering Department, Ferdowsi University of Mashhad, Mashhad, Iran
ssobati@um.ac.ir

Received: 18 March 2025; Accepted: 5 May 2026
Published Online: 30 May 2026

Abstract: Cloud computing refers to a paradigm where users request necessary computing resources through the Internet, and now there are numerous cloud service providers offering various resources and services at affordable costs. However, finding a provider that adequately caters to both commercial and operational needs is becoming increasingly challenging. This research proposes a *bi*-objective virtual machine resource allocation problem, which includes payment cost and execution time as the primary objective criteria. The proposed solution approach involves presenting a *bi*-objective mixed integer problem formulation followed by a two-phase method based on combinatorial optimization techniques to discover all Pareto optimal solutions. In phase 1, the utilized two-phase combinatorial technique locates all supported Pareto optimal solutions, while in phase 2, it obtains the inner non-supported Pareto optimal solutions. Additionally, suitable weights for payment cost and execution time objectives are determined corresponding to all existing Pareto optimal solutions.

Keywords: cloud computing, bi-objective programming, virtual machine assignment, pareto optimality, scientific workflow, resource allocation.

AMS Subject classification: 90-XX, 90Cxx, 90C27

1. Introduction

In this section, we give a table of acronyms to simplify the text reading, see Table (1). The popularity of *cloud computing*, a model where computer resources are provided

* *Corresponding Author*

Table 1. **List of Acronyms**

Abbr.	Description	Abbr.	Description
BO-CSP	Bi-objective cloud service providing	ILP	Integer linear programming
CRCP	Computer resource and cloud production	ML	Machine learning
CRMP	Cloud resource management problem	MLR	Multiple linear regression
CSP	Cloud system provider	MOOP	Multi-objective optimization problem
FC	Federated cloud	PO	Pareto optimal
GP	Graph processor	VM	Virtual machine
GRASP	Greedy randomized adaptive random search		
GRASP-FC	GRASP for federated clouds		

to customers based on their demand and priced on a pay-per-use basis, has soared in recent years. Cloud platforms typically host data centers that manage large-scale Virtual Machines (VMs), which are especially useful for computationally intensive tasks. Users only pay for the VMs they actually use, as these resources are dynamically allocated to user workloads based on application requirements. In today's business landscape, every cloud service provider offers an extensive range of VM options at varying price points. However, users must still carefully select the best resources based on their specific needs in order to complete their tasks efficiently. Similar services are offered by various Cloud System Providers (CSPs), each with its own set of additional options, features, pricing levels, and quality standards. It is possible that a CSP may be expensive for computing yet inexpensive for storage. With so many cloud service options available, the customers face challenges to choose the optimal CSPs for their requirements. Evaluating several CSPs using different assessment criteria is necessary to determine whether a CSP is an ideal candidate for a cloud user's needs. Therefore, providing clients with the most significant resources and CSP services is a fundamental research paradigm, as seen in Sun et al. [24]. For a comprehensive analysis of VM placement schemes in cloud computing, refer to Chaudhary and Kumar [7], Masdari et al. [13].

Various CSPs offer different classes of VMs, each with unique properties such as CPU, memory, disk storage, and pricing. Overestimating or underestimating these requirements can result in poor performance or high costs. Therefore, identifying the features of available VMs and conducting a proper cost study across different CSPs is crucial. The main challenge lies in selecting a CSP that can meet the user's task at minimum cost and execution time. However, solving the Constrained Multi-Objective Optimization Problem (MOOP) using a parametric single-objective optimization problem may be less effective than *dichotomic* methods. Adopting a parametric approach to MOOP requires predetermined weights for objective criteria to compromise the set of all efficient optimal solutions known as Pareto-Optimal (PO) solutions, as noted by Ehrgott [10]. In this paper, we present a *bi*-objective mathematical model aimed at selecting the optimal VMs from CSPs that satisfy both payment cost and execution time criteria. To achieve this goal, we utilize a dichotomic method to solve the proposed *bi*-objective model. The dichotomic method is an exact two-phase combinatorial approach originally introduced by Teghem [26] for solving combinatorial *bi*-objective problems. We evaluate the proposed method on a set of instances provided by well-known CSPs and find that it can generate a collection of PO solutions

with their respective weights, which facilitates the selection of more efficient CSPs. Based on the research by Coutinho et al. [8], Cloud Resource Management Problem (CRMP) is a multi-criteria optimization model that takes into account users' preferences for both cost and performance. The authors propose an integer problem formulation that provides the resource requirements of applications, cost limitations, and different types of VMs. In addition, Coutinho et al. [9] suggest an integer programming formulation and a Greedy Randomized Adaptive Search Procedure (GRASP) to select the best CSP in federated clouds (FCs). The authors use a weighted algorithm in three primary ways, limiting the results to these specific sets of weights only. Specifically, the variables α_1 and α_2 corresponding to weights of first and second objectvies are divided into three pairs of weights: $(0, 1), (0.5, 0.5)$, and $(1, 0)$, while other potentially existing POs are not considered. Our proposed method offers a set of different weights, i.e., α_1, and α_2, based on the user's constraints, to find all POs. To this aim, a *bi*-objective mixed integer mathematical model was explored, which provides all PO solutions with the best performance in terms of paid cost and execution time criteria. The added value of the proposed model and its solution approach are verified by some numerical examples with some reliable data taken from a list of CSPs, see Table (6). The significant contributions of this paper are listed as follows:

- A *bi*-objective cloud system providing (BO-CSP) model, inspired from Coutinho et al. [8], is given to select all efficient sets of VMs from CSPs where both execution time and monetary cost are considered as two distinct objective criteria.
- To solve the BO-CSP, an accurate dichotomic two-phase technique, first proposed by Teghem [26] then extended by Visée et al. [28], is used.
- The BO-CSP model, along with its solution approach, provides the complete set of all PO solutions. This is in contrast to other models, like those proposed by Coutinho et al. [8] and Coutinho et al. [9] which provided some but not all PO solutions.
- Furthermore, the intervals for the weights of the objective criteria are provided to characterize the PO solutions.
- Some real-world CSPs' selections based on distinct selective factors are provided to demonstrate the feasibility and effectiveness of the proposed framework.

The remainder of this paper is organized as follows. In section (2), some related works to the CSP resource allocation problem on single and FCs are reviewed. In section (3), the mathematical problem formulation BO-CSP, and some basic definitions are given. Next, in section (4), a dichotomic two-phase algorithm is illustrated and its details are discussed. In section (5), some numerical examples have been provided to compare the efficiency of some basic VMs for three different CSPs.

2. Related works

In order to make decisions regarding pricing, timing, and Quality of Service (QoS) when purchasing cloud services, a CSP selector method should be able to identify variations in demand and service activities. It is important to have an effective strategy for increasing system reliability while meeting user requirements for computing, storage, and CPUs within the constraints of a limited budget and suitable execution time. The cost of purchasing VMs is closely tied to the cloud provider's pricing structure. Pay-as-you-go pricing, also known as on-demand pricing, is the most commonly used vendor selection method in public clouds. In this model, the cost of using VMs is a set fee per unit of time, such as an hour. This fee varies based on the types of VMs' capacity and deployment zones. Many related works use actual prices from at least one vendor and model on-demand pricing (e.g., from Amazon EC2). It is important to consider these factors when making decisions about cloud purchases.

Noshy et al. [16] evaluates some optimization methods for using VMs without delay on CSPs. Bajo et al. [3] proposed a decentralized resource allocation where a multi-agent system can be used to re-allocating resources in a cloud computing service. Chaisiri et al. [5, 6] offer a stochastic programming approach to support the provisioning of VMs, both on-demand and reservation plans, while demand and pricing limitations are taken into account. To find optimal solutions, the authors consider several approaches, including a deterministic equivalent formulation, sample-average approximation, and Benders' decomposition.

Coutinho et al. [9] introduce the Cloud Resource Management Problem (CRMP), a parametric binary model that takes user cost and performance preferences into account. The authors outlined a method for integer programming that considers application resource needs, cost restrictions, and various VM types. A Greedy Randomized Adaptive Search Procedure (GRASP) is proposed as a solution approach to the problem. Coutinho et al. [9] also introduced a heuristic GRASP approach for federated clouds (GRASP-FC) to reduce the execution time of each stage of the workflow. To evaluate the specific resources required for each task, the authors used the GRASP meta-heuristic. The proposed approach analyzes a selection of optimal cloud resources to fit the mapped activities in the minimum paid cost.

Abdi et al. [1] propose the GRASP-FC approach to reduce extra paid costs such as those related to maintaining VMs and moving data between different CSPs. Although their method is not intended to provide resource-optimized allocation, the authors claim that employing the GRASP-FC meta-heuristic reduces the amount of time spent searching significantly. Genez et al. [12] provided an integer linear programming (ILP) to generate low-cost scheduling for workflow execution across various CSPs. To reduce the search time for processes, the authors employed a granularity-based approach. Considering costs and execution times using a historical database, the authors noted that the algorithm's runtime might be longer than the workflow's execution time itself, making the use of ILP impractical.

To forecast the duration of workflow tasks for various cloud providers, Pham et al.

Table 2. **Comparing related works from different perspectives and criteria**

	[8]	[9]	[1]	[12]	[20]	[22]	dichotomic method
Tech.	CRMP	GRASP	GRASP-FC	ILP	ML	GRASP+MLR	BO-CSP
Multi Cloud		•	•	•	•	•	
Resources							
Mem.	•	•	•	•			•
Stor.	•	•	•	•			•
CPU	•	•	•	•			•
Objectives							
Cost	•	•					•
Time	•	•	•	•	•	•	•
Weight							•

[20] suggests an analytical modeling-based weather forecasting method. Two steps, including machine learning and a historical database, were used to give a predictive model. Based on the task information and the VMs, the first phase calculates the task execution time, and the second phase backs up the time estimation when the historical execution database is empty. However, the authors demonstrated that applying the two-phase method results in an average absolute inaccuracy and error for the group of procedures. To support more predictor variables that function with GRASP and a multiple linear regression method, Rosa et al. [22] introduced a Computational Resource and Cost Prediction service (CRCP). The CRCPs use a pre-processing stage to estimate costs based on how long the workflows take to complete. Because users' needs are not focused on such criteria as time, cost, or feedback from workflow execution, the users are allowed to submit the workflows for either a low-cost or high-performance execution time.

In Table (2), we compared the related works together and with our results under the column *dichotomic method*. We considered the problem formulation works on a single cloud or a set of FCs or cloud brokers. A cloud broker creates a plan for reserving VMs first, with the option to purchase additional machines if the authorized number is exceeded during the use phase. In first part of Table (2), the aforementioned mathematical methods in related work are compared with the BO-CSP model. In the second portion of Table (2), the customers' requirements including memory, disk storage, and computing power are compared. In the third portion of Table (2), the objective functions criteria, including execution time and paid cost, are compared. As reported in Table (2), only Coutinho et al. [8] and Coutinho et al. [9] are concerned with both execution time and paid cost criteria, and the other researchers considered execution time as the objective function. However, the proposed problem formulation by Coutinho et al. [8] and Coutinho et al. [9] are parametric models wherein the weights of objectives are given as known values, and a few but not all PO solutions are determined, while in the proposed dichotomic two-phase algorithm for the BO-CSP model, all PO solutions and their related weights are obtained. Furthermore, the intervals of weights are obtained and all PO solutions are determined.

3. Cloud service provider problem formulation

In this section, we present a *bi*-objective CSP problem formulation based on the work of Coutinho et al. [8]. The authors considered a single parametric objective function to evaluate the payment cost and execution time required for performing a workflow on VMs. The primary criterion for CSP customers is to execute workflows with varying resource requirements on cloud-based VMs within reasonable time frames and limited budgetary constraints. CSPs are responsible for providing the necessary underlying resources to customers by offering various options for executing VMs. For instance, Table 6. in the appendix provides some input setting examples for performing tasks on several selected CSPs, including Microsoft Azure, Google, and Amazon CSPs. The provided data were obtained from the URL addresses https://cloud.google.com/, https://azure.microsoft.com/en-us.

In this paper, an Infrastructure as a Service (Iaas) model is considered in which users submit minimum values of some requests including disk storage, memory, and graph processor. The two proposed objective criteria are the paid cost and execution time. To formulate the mathematical model, a tuple binary variable is defined where

$$x_{p,i,t} = \begin{cases} 1 & \text{If package } i \text{ of kind } p \text{ is selected at time } t \\ 0 & \text{Otherwise.} \end{cases}$$

The other used notations are inserted in Table (3). The processing capacity of each VM of type p is represented by the number of Giga floating operations per second ($GFlops$), or g_p, which is delivered along with the number of hours of execution time. The VM also has disk space d_p and memory m_p allocated. Here, the objective is to reduce the total costs and execution time related to purchasing VMs from a specific CSP.

Table 3. Notations used in the BO-CSP problem formulation

Notation	Definition	Notation	Definition
P	Different groups of VMs	g_p	Graph processor provided by VM type p
C_{total}	Total paid cost	m_p	Available memory in VM type p
I	Set of virtual machines in a group	G_{min}	Minimum required graph processor
T	Set of execution times	D_{min}	Minimum required disk storage
$X_{P,I,T}$	Vector solution of all $x_{p,i,t}$	M_{min}	Minimum required memory
$X^1_{P,I,T}$	Elements of $X_{P,I,T}$ with value 1	t_{min}	Minimum execution time
c_p	Cost of using VM type p	Z^i_{lex}	Lexicographic Pareto solution for $i = 1, 2$
d_p	Disk storage of VM type p	S_E	Set of supported Pareto solutions
		NS_E	Set of non-supported Pareto solutions

Next, we present a *bi*-objective mixed integer model and discuss how it is used to solve the CSP. A different version of this model with a single parametric objective function, including paid cost and execution time objective criteria, is given by Coutinho et al.

[8]. The proposed *bi*-objective mixed integer mathematical model is given as follows:

$$z_{time} = \min \ \ t_m \tag{3.1a}$$

$$z_{cost} = \min \sum_{p=1}^{P} \sum_{i=1}^{N_m} \sum_{t=1}^{T} c_p x_{p,i,t} \tag{3.1b}$$

$$s.t. \sum_{p=1}^{P} \sum_{i=1}^{N_m} \sum_{t=1}^{T} c_p x_{p,i,t} \leq C_{total} \tag{3.1c}$$

$$\sum_{p=1}^{P} \sum_{i=1}^{N_m} \sum_{t=1}^{T} g_p x_{p,i,t} \geq G_{min} \tag{3.1d}$$

$$\sum_{p=1}^{p} \sum_{i=1}^{N_m} d_p x_{p,i,t} \geq D_{min} x_{p',i',t} \quad , \forall t \in T, \forall p' \in P, \forall i' \in I \tag{3.1e}$$

$$\sum_{p=1}^{p} \sum_{i=1}^{N_m} m_p x_{p,i,t} \geq M_{min} x_{p',i',t} \quad , \forall t \in T, \forall p' \in P, \forall i' \in I \tag{3.1f}$$

$$\sum_{p=1}^{p} \sum_{t=1}^{T} x_{p,i,t} \leq N_m \quad \forall i \in I \tag{3.1g}$$

$$t_{min} \geq t x_{p,i,t} \forall p \in P, \ \forall i \in I, \ \forall t \in T \tag{3.1h}$$

$$x_{p,i,t+1} \leq x_{p,i,t}, \ \forall p \in P, \ \forall i \in I, \ \forall t \in T \tag{3.1i}$$

$$x_{p,i+1,t} \leq x_{p,i,t}, \ \forall p \in P, \ \forall i \in I, \ \forall t \in T \tag{3.1j}$$

$$x_{p,i,t} \in \{0,1\}, \ \forall p \in P, \ \forall i \in I, \ \forall t \in T. \tag{3.1k}$$

To address the BO-CSP model, given by equations (3.1a)-(3.1k), we suggest employing a dichotomic approach to identify the PO solutions. In contrast to parametric models where objective function weights are determined upfront, our proposed method involves obtaining the first two minimum lexicographically PO solutions. Subsequently, the weights associated with objective functions (3.1a) and (3.1b) are taken into account. The two objective functions (3.1a) and (3.1b), represent the execution time and paid cost, respectively. Constraint (3.1c) ensures that the total paid cost does not exceed a threshold value of c_m. Constraint (3.1d) requires that the provided graph processor be greater than or equal to a pre-specified value of G. Inequalities (3.1e) and (3.1f) ensure that the provided disk storage and memory capacity in each period are sufficiently large with respect to their lower bound values, D_s and M_c, respectively. Constraint (3.1g) states that the total number of used VMs should not exceed an upper bound, N_m. Constraint (3.1h) specifies the required time for executing a task. Constraint (3.1i) ensures that there are no idle periods between the use of VMs of a particular kind p. Constraint (3.1j) guarantees symmetric solutions. Finally, Constraint (3.1k) defines the decision variable $x_{p,i,t}$ as binary $\{0,1\}$ values, as described in

Coutinho et al. [8]. To evaluate the proposed BO-CSP problem, some necessary definitions and preliminary information about PO solutions are provided, as outlined by Ehrgott [10]. A vector $Z = \begin{bmatrix} z_{time} \\ z_{cost} \end{bmatrix}$ represents the objective function values associated with a feasible solution to the mathematical model (1a)-(1k).

Definition 1. The set of all feasible solutions, $X_{P,I,T}$, to the BO-CSP mathematical model, (3.1a)-(3.1k), is feasible set, and the set of $Z(X_{P,I,T}) = \begin{bmatrix} z_{time}(X_{P,I,T}) \\ z_{cost}(X_{P,I,T}) \end{bmatrix}$ of all feasible solutions is called the design space of objective functions.

For an in-depth examination of the design space of multi-objective programming, we refer interested readers to the Mattson et al. [14].

Definition 2. A vector $Z = \begin{bmatrix} z_{time} \\ z_{cost} \end{bmatrix}$ from the design space is dominated by the vector $\bar{Z} = \begin{bmatrix} \bar{z}_{time} \\ \bar{z}_{cost} \end{bmatrix}$ if both $\bar{z}_{time} \leq z_{time}$, $\bar{z}_{cost} \leq z_{cost}$ and $\bar{Z} \neq Z$.

Remark 1. A vector $\bar{Z} = \begin{bmatrix} \bar{z}_{time} \\ \bar{z}_{cost} \end{bmatrix}$ from the design space is non-dominated point if there is no other solution vector in the design space which is dominated to the vector $\bar{Z}$. Furthermore, $\bar{Z} < Z$ if both $\bar{z}_{time} < z_{time}$ and $\bar{z}_{cost} < z_{cost}$.

Definition 3. A feasible vector solution $\bar{X}_{P,I,T}$ is a weakly optimal solutions, if there is no $X_{P,I,T}$ so that $Z(X_{P,I,T}) < Z(\bar{X}_{P,I,T})$. The weakly optimal solution also is called Pareto optimal solution.

Definition 4. A feasible vector solution $\bar{X}_{P,I,T}$ to the problem (3.1a)-(3.1k) is a strongly efficient solution if no other feasible solution $X_{P,I,T}$ exists satisfying $Z(X_{P,I,T}) \leq Z(\bar{X}_{P,I,T})$ and $Z(X_{P,I,T}) \neq Z(\bar{X}_{P,I,T})$.

Remark 2. In practical report of *PO*s only the strongly efficient solutions should be mentioned, and the weakly *PO*s should be filtered out.

Definition 5. The vector $Z_1 = (z^1_{time}, z^1_{cost})$ is lexicographically smaller than $Z_2 = (z^2_{time}, z^2_{cost})$ with respect to the first entry and denoted by $Z_1 \prec_1 Z_2$ provided that $z^1_{time} < z^2_{time}$. Similarly, Z_1 is lexicographically smaller than Z_2 concerning to the second entry and denoted by $Z_1 \prec_2 Z_2$ provided that $z^1_{cost} < z^2_{cost}$.

Definition 6. The minimum lexicographically solution from the design concerning to the ith entry is denoted by Z^i_{lex}, $i = 1, 2$.

The set of PO solutions can be divided into two primary sets: boundary and interior solutions. This categorization has been discussed in Przybylski et al. [21]. The

boundary PO solutions are connected by a line and act as a delimiter for the solution points within the design space.

- The set of all PO solutions related to a weighted single objective function $\min\{\alpha_1 Z_1 + \alpha_2 Z_2 | \alpha_1 + \alpha_2 = 1, \quad \alpha_1 \geq 0, \quad \alpha_2 \geq 0\}$ subject to the given constraints is called supported PO solutions and are denoted By $S_E = \{Z_{s_1}, ..., Z_{s_f}\}$.
- Two PO solutions $Z_{s_i} = (z^i_{time}, z^i_{cost})$ and $Z_{s_j} = (z^j_{time}, z^j_{cost})$ from the set S_E are adjacent provided that
  1. $Z_{s_i} \prec_1 Z_{s_j}$ and $Z_{s_j} \prec_2 Z_{s_i}$
  2. there is no $Z_{s_h} \in S_E$ so that $Z_{s_i} \prec_1 Z_{s_h} \prec_1 Z_{s_j}$ and $Z_{s_j} \prec_2 Z_{s_h} \prec_2 Z_{s_i}$.
- The set of all PO solutions not included in S_E is called the set of nonsupported efficient solutions and denoted by $NS_E = \{Z_{ns_1}, ..., Z_{ns_{f'}}\}$.
- In Fig (4.2).b, the two circle points marked by red color together with two circle points marked by blue color are supported PO solutions, while the uncolored circles inside the triangles are nonsupported PO solutions.

4. A dichotomic two-phase solution approach

This section, we explain the proposed methodology to choose the best candidate among the available CSPs where the two-phase dichotomic algorithm for the given *bi*-objective problem formulation is detailed. In phase 1, we propose a dichotomic scheme where the weights of the execution time (3.1a) and paid cost objective functions (3.1b) are calculated recursively. Next, in phase 2, the set of all interior PO solutions is introduced.

4.1. Phase 1

In the first phase of the dichotomic two-phase algorithm, the supported PO solutions are determined by initially considering the two supported optimal lexicographically POs, $Z^1_{lex} = (z^1_{time}, z^1_{cost})$, $Z^2_{lex} = (z^2_{time}, z^2_{cost})$. To obtain Z^1_{lex} we solve the objective function (3.1a) together with constraints (3.1c)-(3.1k), the solution of this problem yields z^1_{time} and z^1_{cost} and thereby Z^1_{lex}. Similarly to obtain the Z^2_{lex} we solve the objective function (3.1b) with constraints (3.1c)-(3.1k), the obtained solution yields z^2_{time} and z^2_{cost} and thereby Z^2_{lex}. Next, the normal vector, $\alpha(Z^1_{lex}, Z^2_{lex}) = (\alpha_1, \alpha_2)$, to the line connected two Lexicographically POs, Z^1_{lex} and Z^2_{lex}, is determined where $\alpha_1 = z^1_{cost} - z^2_{cost}$ and $\alpha_2 = z^2_{time} - z^1_{time}$. Then the problem BO-P2, (4.1),

$$\begin{aligned} \text{BO-P2:} \quad & \min Z = \alpha_1 z_{time} + \alpha_2 z_{cost} \\ & s.t. \quad (3.1c) - (3.1k) \end{aligned} \tag{4.1}$$

is solved. Indeed, we replace the original problem BO-CSP with a newly weighted single objective function. Whenever a new supported PO solution Z_{int} is obtained between two existing PO solutions Z^1_{lex} and Z^2_{lex}, we update the set $S_E = \{Z^1_{lex}, Z_{int}, Z^2_{lex}\}$, as shown in part a of Fig (4.2). The adjacent members of the set S_E are then determined based on a lexicographically increasing order concerning the z_{time} component, where all members of the set S_E are strong PO solutions.
Next, we proceed to new adjacent solutions to find the newly possible existing support PO solutions. Therefore, the supported PO solutions between pairs Z^1_{lex}, Z_{int}, and Z_{int}, Z^2_{lex} is sought. When a newly supported PO solution is obtained, then it is inserted between its two adjacent. This iterative method is followed up while a new PO solution is not generated. The final supported PO set is denoted by $S_E = \{Z^1_{lex}, Z^1_{int}, ..., Z^m_{int}, Z^2_{lex}\}$. Next, we proceed to Phase 2 of the two-phase algorithm to identify the nonsupported PO solutions.

4.2. Phase 2

Phase 1 generates all supported PO solutions. Then, in Phase 2, the set of all nonsupported PO solutions NS_E is generated by using the solutions obtained in phase 1. The nonsupported PO solutions are sought within the triangle region comprised by two adjacent members of the set S_E. Let $Z_{s_1} = (z^{s_1}_{time}, z^{s_1}_{cost})$ and $Z_{s_2} = (z^{s_2}_{time}, z^{s_2}_{cost})$ be two adjacent belong to the set S_E where $Z_{s_1} \prec_1 Z_{s_2}$ and $Z_{s_2} \prec_2 Z_{s_1}$, see part b of Fig (4.2). To seek the triangle regions, three new constraints, including (4.2a)-(4.2c), are added to the set of constraints (3.1c)-(3.1k), see part a of Fig (4.2).

$$z^{s_1}_{time} \leq z_{time} \leq z^{s_2}_{time} \tag{4.2a}$$
$$z^{s_2}_{cost} \leq z_{cost} \leq z^{s_1}_{cost} \tag{4.2b}$$
$$z_{time}(z^{s_1}_{cost} - z^{s_2}_{cost}) + z_{cost}(z^{s_2}_{time} - z^{s_1}_{time}) - z^{s_1}_{time} z^{s_2}_{cost} + z^{s_1}_{cost} z^{s_2}_{time} \geq 0. \tag{4.2c}$$

Definition 7. Let Z_1 and Z_2 be two adjacent PO solutions of the set S_E, then the set of all solutions satisfying at constraints (4.2a)-(4.2c) is defined as the triangle of Z_1 and Z_2 and denoted by $Tr(Z_1, Z_2)$.

Then all obtained PO solutions within a triangle are cut off by adding some more constraints. Let $\bar{X}_{P,I,T}$ be a feasible vector solution within the selected triangle in Phase 2. We set constraints (4.3) by choosing $\{p, i, t\}$s from the vector $\bar{X}_{P,I,T}$ that are equal to one and denote them as $\bar{X}^1_{P,I,T}$. By adding constraints (4.3) the current obtained nonsupported PO solution within the selected triangle drops out from the feasible region.

$$\sum_{p=1}^{P} \sum_{i=1}^{N_m} \sum_{t=1}^{T} X_{p,i,t} \leq |\bar{X}^1_{P,I,T}| - 1, \tag{4.3}$$

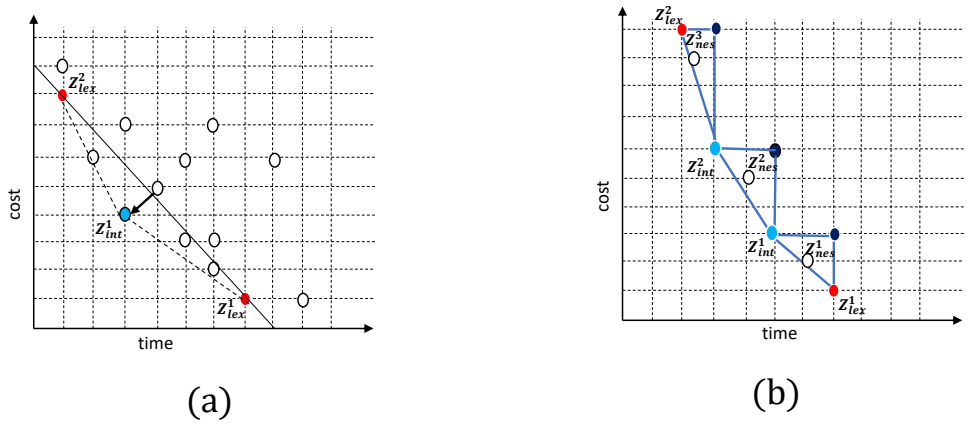

Figure 1. **(a) illustrate obtaining support PO solutions and (b) illustrate obtaining the non-support PO solutions.**

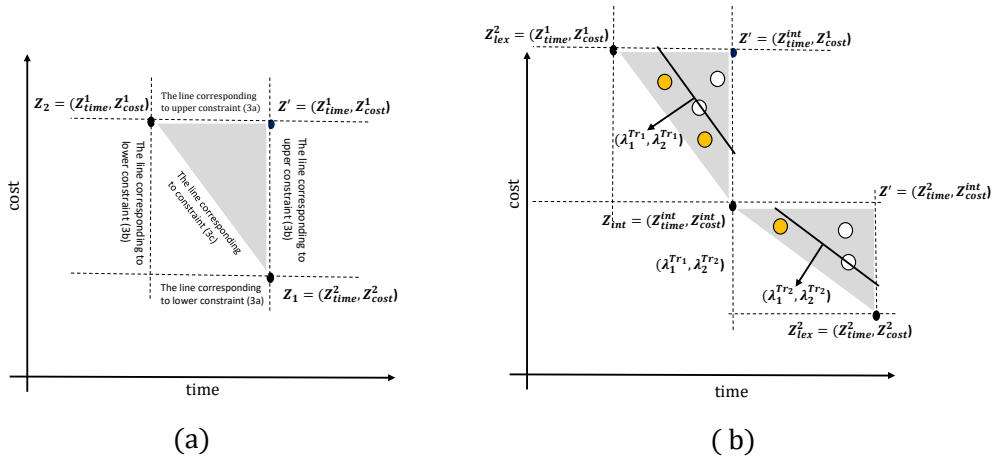

Figure 2. **(a) shows the triangle region related to constraints (4.2a)-(4.2c) and (b) shows two adjacent triangle regions sweeping by constraint cut (4.3).**

where $|\bar{X}^1_{P,I,T}|$ denotes the number of components of $\bar{X}^1_{P,I,T}$. Constraints (4.3) are recursively added to the feasible region of constraints (3.1c)-(3.1k) until there is no generated new nonsupported PO solution, see part b of Fig (4.2). Then, the set of nonsupported PO solutions is denoted by $NS_E = \{Z^1_{ns_e}, Z^2_{ns_e}, \ldots, Z^m_{ns_e}\}$, see part b of Fig (4.2).

In Fig (4.2), phases 1 and 2 of the proposed dichotomic method are illustrated. In part a of Figure 1, two lexicography PO solutions are marked with red circles. Subsequently, a new supported PO solution may be generated by searching in the original feasible region solution of the constraints (1c)-(1k). This new solution is represented by a light blue marked circle in Fig (4.2) part a. After determining all supported

PO solutions in phase 1, some triangle regions that may contain nonsupported PO solutions are identified by utilizing constraints (4.3), as shown in part b of Fig (4.2). Algorithm (1) provides the steps of phase 1 in detail in which all supported PO solutions are found.

Algorithm 1 phase1 of the dichotomic algorithm

Input: The two lexicographic optimal solutions .
Output: The set of all supported PO solutions.
Let $Z^1_{lex} = (z^1_{time}, z^1_{cost})$ and $Z^2_{lex} = (z^2_{time}, z^2_{cost})$ be two Lex_{min} PO solutions where $Z^1_{lex} \prec_1 Z^2_{lex}$ and $Z^2_{lex} \prec_2 Z^1_{lex}$.
Set $Z_1 \leftarrow Z^1_{lex}$, $Z_2 \leftarrow Z^2_{lex}$ and $Z_{next} \leftarrow Z_2$.
Set $S_E = \{Z_1, Z_2\}$
if $Z_1 = Z_2$ **then**
  The PO solution is unique; stop.
else
  Let $\alpha = (\alpha_1, \alpha_2)$ be the normal vector to the line connecting Z_1 to Z_2.
  Solve problem (4.1) by using α.
  Let $\bar{Z}(\alpha) = \alpha_1 \bar{Z}_{time} + \alpha_2 \bar{Z}_{cost}$ be the obtained optimal solution.
  if $\bar{Z}(\alpha) < Z_1\alpha_1 + Z_2\alpha_2$ **then**
    $Z_1 \leftarrow Z_1$ and $Z_2 \leftarrow \bar{Z}(\alpha)$, $Z_{next} \leftarrow Z_2$
    $Z^i_{int} \leftarrow \bar{Z}(\alpha)$ and $i \leftarrow i + 1$
    Set $S_E = S_E \cup \{Z^i_{int}\}$
  else
    if $Z_{next} \neq Z_2$ **then**
      $Z_1 \leftarrow Z_2$ and $Z_2 \leftarrow Z_{next}$
    else
      Stop, return $S_E = \{Z_1, Z^1_{int}, ..., Z^m_{int}, Z_2\}$ as the supported PO solution set.
    end if
  end if
end if

Next, we apply the set S_E to locate the non-supported PO solutions. We examine adjacent solutions from the set S_E that form a triangle, as indicated by constraints (4.2a)-(4.2c) and represented in Fig (4.2) part b. Algorithm (2) depicts phase 2 of the two-phase algorithm used to identify nonsupported PO solutions within the triangle formed by members of S_E. The flowchart (3) provides an overview of the entire dichotomic algorithm, including phases 1 and 2 of solution approach.

4.3. Computational complexity and alternative methods

The computational complexity of the proposed dichotomic algorithm and some alternative methods for binary *bi*-objective problems is an important consideration, especially for solving large scale problems. Since the single combinatorial objective function are NP-hard [18, 23], inherently the binary problems defined by objectives (3.1a) and (3.1b) under the common constraints (3.1c)-(3.1k) are NP-hard, then the corresponding *bi*-objective problem is also NP-hard. Note that, here the coefficient matrix (3.1c)-(3.1k) in the constraint of preposed model is not totally unimodular; therefore, the binary model would not be reduced to a continuous one. However, Some classical models reduce the computational effort and run time to solve large-scale binary combinatorial optimization problems effectively. For example, branch

Algorithm 2 phase2 of the dichotomic algorithm

Input: The set of all supported PO solutions.
Output: The set of all nonsupported PO solutions.
Let $S_E = \{Z_1, Z_{int}^1 ... Z_{int}^m, Z_2\}$ be the set of all supported PO solutions.
Let $Tr = \{Tr_1, ..., Tr_{m+1}\}$ be the set of all triangles of adjacent members of S_E.
Let $NS_E = \{\}$ be the set of all nonsupported PO solutions.
for $n = 1, ..., m+1$ **do**
  Let Z_n and Z_{n+1} be two adjacent members of the set S_E.
  Solve problem (4.1) by using $\alpha(Z_n, Z_{n+1})$
  Let $X_{P,I,T}^n$ and $X_{P,I,T}^{n+1}$ be the related vector solutions corresponding to Z_n and Z_{n+1}.
  Let $X_{P,I,T}^{n,1}$ and $X_{P,I,T}^{n+1,1}$ be the set of indices (p,i,t) where $x_{p,i,t}^n$ and $x_{p,i,t}^{n+1}$ are equal one.
  Add two following cut constraints to the problem (4.1) and call the new problem BO-P2,

$$\sum_{p=1}^{P}\sum_{i=1}^{N_m}\sum_{t=1}^{T} x_{p,i,t} \leq |X_{P,I,T}^{n,1}| - 1$$

$$\sum_{p=1}^{P}\sum_{i=1}^{N_m}\sum_{t=1}^{T} x_{p,i,t} \leq |X_{P,I,T}^{n+1,1}| - 1$$

  Solve the problem BO-P2 ▷ Iteration step
  if The problem BO-P2 is infeasible **then**
    stop and return the set of nonsupported PO solutions within triangle n.
  else
    Let $\bar{X}_{P,I,T}$ be the optimal Solution to the problem BO-P2.
    Set $NS_E = NS_E \cup \bar{X}_{P,I,T}$
    Add the cut constraint (4.3) related to $\bar{X}_{P,I,T}$ to the problem BO-P2.
  end if
  Go to iteration step.
end for

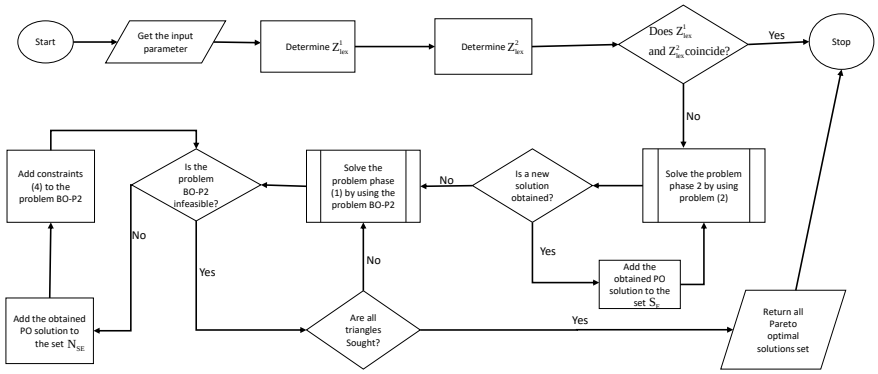

Figure 3. **The flowchart illustraing the dichotomic algorithm.**

and bound [15], heuristics and meta-heuristics [11], evolutionary techniques [17] and neural networks [25] are some well-known methods to tackle these problems. Fur-

thermore, binary *bi*-objective optimization is more challenging topic, particularly for large scale problems. Some classical methods to deal with *bi*-objective optimization problems include,

1. The ϵ-constraint method that converts one objective into constraints and optimizes the other objective. The ϵ parameter defines the allowable level for the objective constraint, see Bérubé et al. [4]. This method needs $o(kN_o])$ time complextiy, where k is the number of ϵ values and N_o is time run to solve single objectives, see [4].

2. Branch and bound algorithms that divide the problem into some smaller problems (branching) and eliminate some unnecessary sub-problems (bounding), see [2, 19]. In worst case branch and bound needs $o(2^n)$ run time, where n is number of variables, [27]

3. Rectangle splitting that iteratively divides objective space into hyper rectangles by eliminating dominated regions, see [19]. This method is doubly exponential in running time since $O(kT_{sub})$ computational effort is needed, where k is the number of POs and T_{sub} is the time to solve the single objective optimization problem, see [19].

4. Two phase dichotomic binary optimization method that in phase 1 seeks all supported solutions, and in phase 2 seeks all non-supported solutions in triangles between supported solutions [21]. The computational run time here is $o(T_{s_e} + T_{ns_e})$, where T_{s_e} is the run time to map out the triangle corresponding to supported points and T_{ns_e} is the run time to search within triangles to find the non-supported POs. This algorithm is fast in phase 1 but heavy in phase 2.

In this research, we employed the dichotomic solution approach due to its completeness and constructing corresponding weights to help the decision maker to choose some preferred PO solutions.

5. Experimental analysis

This section verifies the efficacy of the proposed methodology, using some real-world datasets. Some standards VMs were selected from three CSPs including Google, Microsoft Azure and Amazon. The necessary data has been inserted in Table (6). The two-phase *bi*-objective algorithm has been implemented in Python version 3.9.1 using the *Pyomo* optimization modeling language and *GLPK* software package.

Example 1.

In this example, we examine the data of Amazon cloud service taken from the URL address https://aws.amazon.com/, and given in Table (6). The minimum required power is $G = 440$ core, the minimum required disk storage is $D_{min} = 250$, the

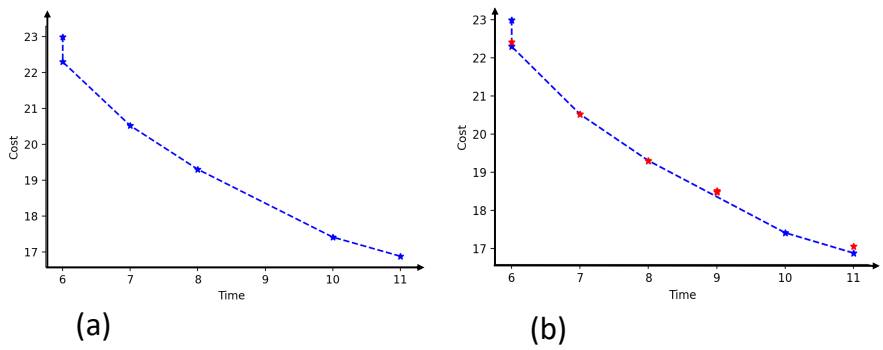

Figure 4. (a) The supported PO solutions and (b) The supported and nonsupported PO solutions for Example (1)

minimum required memory is $M_{min} = 100$, the total available paid cost is $C_{total} = 24$, the maximum available execution time is $T = 12$, the number of each kind of VMs is $I = 5$, and the maximum number of available VMs is $N_m = 18$.

1. The initial minimum lexicographically PO solutions are obtained at $Z^1_{lex} = (22.29, 6)$ and $Z^2_{lex} = (16.88, 11)$. Indeed, the minimum execution time is $z^1_{time} = 6$, while its related paid cost is $z^1_{cost} = 22.29$. Besides, the minimum paid cost is $z^2_{cost} = 16.88$, while its related execution time is $z^2_{time} = 11$.

2. Next, in phase 1, all supported PO solutions, including $(17.41, 10)$, $(19.30, 8)$, $(20.51, 7)$ are obtained. Indeed the set of supported PO solution is

$$S_E = \{(16.88, 11), (17.41, 10), (19.30, 8), (20.51, 7), (22.29, 6)\},$$

see part a of Fig (4).

3. At last, by using phase 2 of the two-phase algorithm all nonsupported PO solutions have been obtained. The set $NS_E = \{(17.05, 11), (18.47, 9)\}$, see part b of Fig (4).

4. The set

$$S_E \cup NS_E = \{(16.88, 11), (17.05, 11), (17.41, 10), (18.47, 9), (19.30, 8), (20.51, 7), (22.29, 6)\},$$

includes all PO solutions. Here, $(17.05, 11)$ is a weak PO solution, obtained in phase 2, within triangle Tr_1. From a theoretical standpoint, we generate all weakly Pareto optimal solutions; however, from a practical perspective, the weakly PO solutions should be filtered out.. Also, the nonsupported PO solution $(18.47, 9)$ is within triangle Tr_2. The other triangles have been sought, but there are no PO solutions within them.

Different values for the required graph processor and the obtained PO solutions														
380		385		390		395		400		405		410		
GP	0.7GP	GP	0.7GP	GP	0.7GP	GP	0.7GP	GP	0.7GP	GP	0.7GP	GP	0.7GP	
(17,7)	(26,7)	(17,7)	(26,7)	(18,6)	(27,7)	(18,7)	(27,7)	(18,7)	(28,6)	(20,4)	(28,7)	(21,3)	(29,6)	
(18,5)	(27,5)	(19,4)	(28,4)	(19,4)	(28,5)	(20,3)	(28,5)	(20,4)	(28,7)	(22,2)	(29,5)	(22,2)	(30,4)	
(20,2))	(28,4)	(21,2)	(29,3)	(20,3)	(29,3)	(21,2)	(30,3)	(21,3)	(31,3)		(31,3)		(31,3)	
(21,2)	(29,3)			(21,2)				(22,2)						

Table 4. The *PO*s of dataset in Table (6) vs scenario with GP values reduced by factor 0.7.

5. Note that the execution time value is integer, and in phase 1, the supported PO solutions have been found at execution times, $\{6, 7, 8, 10, 11\}$. The only remaining execution time between 6 and 10 is 9, which has been found at PO solution $(18.47, 9)$.

Next, to provide a sensitivity analysis, we consider data of the Microsoft Azure cloud taken from the URL address https://azure.microsoft.com/en-us given in Table (6). However, the value of graph processor G_{min} has been increased from 380 to 410 by step-size 5. The set of all PO solutions with their related objective values have been reported in the first part of Table (4). To compare the results in a table, we restricted the cost values to integer numbers. The obtained solutions are inserted lexicographically concerning to the paid cost and execution time objective values in Table (4). As the value of G_{min} increases, their related PO solutions in both objectives criteria, including paid cost and execution time, increase. For example, the solution $(18, 5)$ is PO solution related to the value $G_{min} = 380$. While the PO solution $(18, 6)$ is related to the $G_{min} = 390$, and $(18, 7)$ is related to the $G_{min} = \{390, 395, 400\}$. Indeed, when G_{min} increases, both the values of *cost* and time increase or remain unchanged.
Then to evaluate the effect of decreasing the graph provided processor (GP) by the selected cloud, Microsoft Azure, we multiplied the column with the headline graph processor of Table (6) by coefficient 0.7. The obtained results are inserted in the second part of Table (4). Comparing the results of the first and second parts of Table (4) reveals the cost objective values considerably are increased while the changes in execution time are insignificant. However, the obtained PO solutions in the first part of Table (4) are dominated by the PO solutions of the second part and are more efficient.

Example 2.

In this example, we compare three CSPs, including Google, Microsoft Azure, and Amazon, based on several standard VMs listed in Table (6). We gathered the necessary data from URL addresses https://cloud.google.com/, https://azure.microsoft.com/en-us and https://aws.amazon.com/. To ensure a fair comparison of the proposed VMs by these CSPs, we evaluated their performance under similar conditions, using the data presented in Table (6). The obtained results

are authentic only for these selected VMs. Four scenarios are considered in Table (5) where the results for different values of total permitted paid cost (C_{total}), number of used VMs (N_m), minimum required memory (M_{min}) and minimum necessary of disk storage (D_{min}) have been compared. In the case, where only there is a single PO solution, we report their related weight as *unique* in Table (5).

Scenario 1. First, we compared the solutions by increasing the total paid costs, where the upper value for the paid cost is taken from the set $C_{total} = \{16, 17, 18, 20\}$. As C_{total} increases in constraint (3.1c) the minimum execution time in constraint (3.1h) either decreases or remains unchanged, since the variation of efficient VMs increases. Therefore, higher performances are delivered for the requested tasks.

Scenario 2. In the second case, we examine the variations of PO solutions for the number of admitted VMs in constraint (3.1g). We considered three cases where N_m was set to $\{9, 13, 17\}$. By increasing the values of N_m the payment costs and execution times appear more efficient values or at least remain unchanged.

Scenario 3. In the third case, the results with different values of requested memory are compared. We increase the values of requested memory by $M_{min} = \{150, 200, 250, 300, 350\}$, see the third part of Table (5). When the values of requested memory related to constraint (3.1f) increase efficiency of the PO solutions decreases. Indeed, the values of paid cost and execution time increase or at least remain unchanged.

Scenario 4. In the last case of Table (5), we compared the results with respect to the requested disk storage. Here, the values of requested disk storage are $D_{min} = \{250, 300, 350, 400\}$. Again the sensitivity analysis indicates as the values of minimum requested disk storage increases, then the efficiency of the PO solutions decreases.

6. All PO solutions and their related weights

Using the proposed two-phase dichotomic method, all supported and nonsupported PO solutions, as well as their related weight vector α is obtained. Furthermore, all intervals of weights related to z_{time} and z_{cost} are determined, and their related PO solutions are obtained. Indeed, in the dichotomic proposed method, first, two minimum lithographically PO solutions are obtained, where their related weights are $(1, 0)$ and $(0, 1)$. Then the others supported and nonsupported PO solutions with their related weight vector are determined. If we set the supported PO solutions where adjacent solutions are next together, then the intervals of weights are obtained. To find the intervals where PO solutions are optimal, we consider two adjacent PO from the set S_E and determine the normal vector connecting these adjacent supported PO solutions. For example, consider $Z_{s_i} = (z^{s_i}_{time}, z^{s_i}_{cost})$ with $\alpha_{s_i} = (\alpha_1^{s_i}, \alpha_2^{s_i})$ and $Z_{s_{i+1}} = (z^{s_{i+1}}_{time}, z^{s_{i+1}}_{cost})$ with $\alpha_{s_i} = (\alpha_1^{s_{i+1}}, \alpha_2^{s_{i+1}})$ and let $\alpha_{s_i,s_{i+1}} = (\alpha_1^{s_i,s_{i+1}}, \alpha_2^{s_i,s_{i+1}})$ be the normal vector to the line connecting Z_{s_i} and $Z_{s_{i+1}}$. Then $Z(\alpha)$ would be

Table 5. **The POs and their related weights for different scenarios in Example (2)**

G_{min} = 400, M_{min}=100, D_{min}=150, N_m = 17, I = 5, T = 10						
CSP	Microsoft Azure		Google		Amazon	
	solution	α	solution	α	solution	α
$C_{total} = 16$	(15.87, 10)	unique	(13.40, 2)	unique	Inf	–
$C_{total} = 17$	(15.87, 10) (16.42, 9) (16.98, 8)	(1,0) (0.65,0.35) (0, 1)	(13.40, 2)	unique	(16.13, 10) (16.61, 9)	(1,0) (0, 1)
$C_{total} = 18$	(15.87, 10) (16.42, 9) (17.17, 8) (17.51, 7)	(1, 0) (0.65, 0.35) (0.65, 0.35) (0,1)	(13.40, 2)	unique	(16.13, 10) (16.61, 9) (17.13, 8)	(1,0) (0.34, 0.66) (0, 1)
$C_{total} = 20$	(15.87, 10) (16.98, 8) (17.54, 7) (18.29, 6) (18.90, 5) (19.50, 4)	(1, 0) (0.65, 0.35) (0.63, 0.37) (0.60, 0.40) (0.60, 0.40) (0,1)	(13.40, 2)	unique	(16.13, 10) (16.61, 9) (17.13, 8) (18.25, 7) (19.71, 6)	(1,0) (0.66, 0.34) (0.60, 0.40) (0.60, 0.40) (0, 1)
$N_M = 9$	(18.22, 10) (18.55, 9) (19.05, 8) (19.49, 7) (19.79, 6)	(1,0) (0.75, 0.25) (0.71, 0.29) (0.71, 0.29) (0,1)	Inf	–	(19.41, 10)	unique
$N_M = 13$	(16.85, 10) (17.74, 8) (18.43, 7) (18.88, 6) (19.50, 5)	(1,0) (0.66, 0.34) (0.66, 0.34) (0.66, 0.34) (0,1)	Inf	–	(17.62, 10) (18.12, 9) (18.75, 8) (19.94, 7)	(1, 0) (0.64, 0.36) (0.57, 0.43) (0, 1)
$N_m = 17$	(15.87, 10) (16.98, 8) (17.54, 7) (18.29, 6) (18.90, 5) (19.50, 4)	(1, 0) (0.65, 0.35) (0.63, 0.37) (0.60, 0.40) (0.60, 0.40) (0,1)	(13.40, 2)	unique	(16.13, 10) (16.61, 9) (17.13, 8) (18.25, 7) (19.71, 6)	(1,0) (0.66, 0.34) (0.60, 0.40) (0.60, 0.40) (0, 1)
$M_{min} = 150$	(17.36, 8) (17.73, 7) (18.29, 6) (18.90, 5) (19.50, 4)	(1,0) (0.73, 0.27) (0.63, 0.37) (0.63, 0.37) (0,1)	(13.40, 2)	unique	(17.13, 8) (18.25, 7) (19.71, 6)	(1, 0) (0.58, 0.42) (1, 0)
$M_{min} = 200$	(18.29, 6) (19.50, 4)	(1,0) (0,1)	(13.40, 2)	unique	(19.24, 7) (19.71, 6)	(1, 0) (0, 1
$M_{min} = 250$	(18.90, 5) (19.50, 4)	(1,0) (1,0)	(13.40, 2)	unique	Inf	–
$M_{min} = 300$	(19.50, 4)	unique	(13.4, 2)	unique	Inf	–
$M_{min} = 350$	(19.92, 4)	unique	(13.4, 2)	unique	Inf	–
$D_{min} = 250$	(16.65, 10) (16.70, 9) (17.17, 8) (17.73, 7) (18.29, 6) (18.90, 5) (19.50, 4)	(1,0) (0.75,0.25) (0.65,0.35) (0.65,0.35) (0.65,0.35) (0.62,0.38) (0,1)	Inf	–	(16.13, 10) (16.61, 9) (17.13, 8) (18.25, 7) (19.71, 6)	(1,0) (0.66, 0.34) (0.51, 0.49) (0.51, 0.49) (0, 1)
$D_{min} = 300$	(17.18, 8) (18.29, 6) (18.90, 5) (19.50, 4)	(1,0) (0.65,0.35) (0.65,0.35) (0,1)	Inf	–	(16.13, 10) (16.61, 9) (17.13, 8) (18.25, 7) (19.71, 6)	(1,0) (0.68, 0.32) (0.50, 0.50) (0.42, 0.58) (0, 1)
$D_{min} = 350$	(17.73, 7) (18.29, 6) (18.90, 5) (19.50, 4)	(1,0) (0.63, 0.37) (0.62,0.38) (0,1)	Inf	–	(16.13, 10) (16.64, 9) (18.13, 8) (19.71, 6)	(1,0) (0.68, 0.32) (0.50, 0.50) (0, 1)

represented as follows

$$Z(\alpha) = \begin{cases} Z_{s_i} & \text{if} \quad \alpha_1^{s_i,s_{i+1}} \leq \alpha_1 \leq \alpha_1^{s_i} \quad \text{and} \quad \alpha_2^{s_i} \leq \alpha_2 \leq \alpha_2^{s_i,s_{i+1}} \\ Z_{s_{i+1}} & \text{if} \quad \alpha_1^{s_{i+1}} \leq \alpha_1 \leq \alpha_1^{s_i,s_{i+1}} \quad \text{and} \quad \alpha_2^{s_i,s_{i+1}} \leq \alpha_2 \leq \alpha_2^{s_{i+1}}. \end{cases}$$

For example, consider the row with row-head $D_{min} = 300$ of Table (5). Here, five supported PO solutions including $S_E = \{(16.13, 10), (16, 61, 9), (17.13, 8), (18.25, 7), (19.71, 6)\}$ have been obtained, and their related weights are $\{(1, 0), (0.68, 0.32), (0.5, 0.5), (0.42, 0.58), (0, 1)\}$.

$$Z(\alpha) = \begin{cases} (16.13, 10) & \text{if} \quad 0.68 \leq \alpha_1 \leq 1 \quad \text{and} \quad 0 \leq \alpha_2 \leq 0.32 \\ (16.61, 9) & \text{if} \quad 0.65 \leq \alpha_1 \leq 0.68 \quad \text{and} \quad 0.32 \leq \alpha_2 \leq 0.35 \\ (17.13, 8) & \text{if} \quad 0.47 \leq \alpha_1 \leq 0.65 \quad \text{and} \quad 0.35 \leq \alpha_2 \leq 0.53 \\ (18.25, 7) & \text{if} \quad 0.4 < \alpha_1 \leq 0.47 \quad \text{and} \quad 0.53 \leq \alpha_2 \leq 0.6 \\ (19.71, 6) & \text{if} \quad 0 \leq \alpha_1 \leq 0.4 \quad \text{and} \quad 0.6 \leq \alpha_2 \leq 1. \end{cases}$$

6.1. A brief discussion in obtained solutions:

The results indicate that the main difference between this work and previous research, as discussed in section (2), is our approach of determining all PO solutions and obtaining their related weights. In contrast, other researches have used parametric *bi*-objective methods where fixed weights are assigned to the objective criteria, transforming the bi-objective problem into a single-objective problem. For instance, in the works of Coutinho et al. [8], Coutinho et al. [9], only a limited set of weights, such as $\{(1, 0), (0.5, 0.5), (0, 1)\}$, were assigned to the objectives criteria, while other potential weights were neglected. Our approach, on the other hand, introduces all PO solutions and their related weights, resulting in a more comprehensive analysis. Therefore, the key distinction between our work and previous researches is our method of exploring the entire range of weights rather than relying on a limited set of predetermined weights. This enables us to gain a better understanding of the trade-offs between different objectives and identify the optimal PO solutions.

7. Conclusion and future works

In this paper, we considered a CSP selection where some VMs are employed to conduct the user requests workloads. A *bi*-objective integer linear programming has been formulated to determine all PO solutions as well as their related weights. Indeed, to find the PO solutions there are two main approaches including parametric and dichotomic methods. In implementation parametric methods some weights for objectives criteria are given then the related PO solutions are obtained. But in the

proposed dichotomic methods by determining two minimum lexicographically PO solutions, recursively related intervals for weights, and all PO solutions are obtained. In this paper, we applied an exact dichotomic method for selecting some VMs from a set of some candidate CSPs to explore the design space of two objective criteria including execution time and paid costs. Besides some customers's requirements are met by related tight constraints. The validity of the proposed *bi*-objective model as well as its solution approach is verified by some provided experimental results. For future work, one could focus on bi-objective FCs that allow for obtaining services from multiple CPS. Additionally, exploring PO solutions in cases where three or more objective criteria are evaluated would be an intriguing research topic.

8. Technical Comment Report

In this paper, we investigate a *bi*-objective cloud computing provider selection problem that considers two contrasting combinatorial objective functions: minimizing execution times and paid costs. To address this problem, we propose an efficient dichotomic solution approach to provide solutions for all PO alternatives. The main contribution of this paper lies in formulating a bi-objective problem for cloud service providers CSPs that helps users select the most suitable CSP based on their available budget and limited execution time. Additionally, our proposed two-phase dichotomic method ensures the identification of not only a few, but all possible PO solutions. Overall, this research presents a novel approach to tackle the bi-objective cloud computing selection problem and offers valuable insights for users in making informed decisions while considering their budget constraints and time limitations.

Conflict of Interest: The authors declare that they have no conflict of interest.

Data Availability: Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

References

[1] S. Abdi, L. PourKarimi, M. Ahmadi, and F. Zargari, *Cost minimization for bag-of-tasks workflows in a federation of clouds*, The Journal of Supercomputing **74** (2018), no. 6, 2801–2822.
https://doi.org/10.1007/s11227-018-2322-9.

[2] N. Adelgren and A. Gupte, *Branch-and-bound for biobjective mixed-integer linear programming*, INFORMS J. Comput. **34** (2022), no. 2, 909–933.

[3] J. Bajo, F. De la Prieta, J.M. Corchado, and S. Rodríguez, *A low-level resource allocation in an agent-based cloud computing platform*, Applied Soft Computing **48** (2016), 716–728.
https://doi.org/10.1016/j.asoc.2016.05.056.

[4] J.F. Bérubé, M. Gendreau, and J.Y. Potvin, *An exact ϵ-constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits*, European J. Oper. Res. **194** (2009), no. 1, 39–50. https://doi.org/10.1016/j.ejor.2007.12.014.

[5] S. Chaisiri, B.S. Lee, and D. Niyato, *Optimal virtual machine placement across multiple cloud providers*, 2009 IEEE Asia-Pacific Services Computing Conference (APSCC), IEEE, 2009, pp. 103–110.

[6] ———, *Optimization of resource provisioning cost in cloud computing*, IEEE transactions on services Computing **5** (2011), no. 2, 164–177. https://doi.org/10.1109/tsc.2011.7.

[7] D. Chaudhary and B. Kumar, *Cost optimized hybrid genetic-gravitational search algorithm for load scheduling in cloud computing*, Applied Soft Computing **83** (2019), 105627. https://doi.org/10.1016/j.asoc.2019.105627.

[8] R.D.C. Coutinho, L.M. Drummond, and Y. Frota, *Optimization of a cloud resource management problem from a consumer perspective*, European Conference on Parallel Processing, Springer, 2013, pp. 218–227.

[9] R.D.C. Coutinho, L.M. Drummond, Y. Frota, and D. de Oliveira, *Optimizing virtual machine allocation for parallel scientific workflows in federated clouds*, Future Generation Computer Systems **46** (2015), 51–68. https://doi.org/10.1016/j.future.2014.10.009.

[10] M. Ehrgott, *Multicriteria Optimization*, vol. 491, Springer Science and Business Media, 2005.

[11] P. Fouilhoux, L. Létocart, and Y. Zhang, *A generic branch-and-cut algorithm for bi-objective binary linear programs*, 2024. https://doi.org/10.48550/arXiv.2410.08722.

[12] T.A.L. Genez, L.F. Bittencourt, and E.R.M. Madeira, *Time-discretization for speeding-up scheduling of deadline-constrained workflows in clouds*, Future Generation Computer Systems **107** (2020), 1116–1129. https://doi.org/10.1016/j.future.2017.07.061.

[13] M. Masdari, S.S. Nabavi, and V. Ahmadi, *An overview of virtual machine placement schemes in cloud computing*, Journal of Network and Computer Applications **66** (2016), 106–127. https://doi.org/10.1016/j.jnca.2016.01.011.

[14] C.A. Mattson, A.A. Mullur, and A. Messac, *Smart pareto filter: Obtaining a minimal representation of multiobjective design space*, Eng. Optim. **36** (2004), no. 6, 721–740. https://doi.org/10.1080/0305215042000274942.

[15] D.R. Morrison, S.H. Jacobson, J.J. Sauppe, and E.C. Sewell, *Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning*, Discrete Optim. **19** (2016), 79–102. https://doi.org/10.1016/j.disopt.2016.01.005.

[16] M. Noshy, A. Ibrahim, and H.A. Ali, *Optimization of live virtual machine migration in cloud computing: A survey and future directions*, Journal of Network and

Computer Applications **110** (2018), 1–10.
https://doi.org/10.1016/j.jnca.2018.03.002.

[17] O.N. Oyelade, J.O. Agushaka, and A.E. Ezugwu, *Evolutionary binary feature selection using adaptive ebola optimization search algorithm for high-dimensional datasets*, Plos One **18** (2023), no. 3, e0282812.

[18] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Courier Corporation, 1998.

[19] S.N. Parragh and F. Tricoire, *Branch-and-bound for bi-objective integer programming*, INFORMS J. Comput. **31** (2019), no. 4, 805–822.

[20] T.P. Pham, J.J. Durillo, and T. Fahringer, *Predicting workflow task execution time in the cloud using a two-stage machine learning approach*, IEEE Transactions on Cloud Computing **8** (2020), no. 1, 256–268.

[21] A. Przybylski, X. Gandibleux, and M. Ehrgott, *Two phase algorithms for the bi-objective assignment problem*, European J. Oper. Res. **185** (2008), no. 2, 509–533.
https://doi.org/10.1016/j.ejor.2006.12.054.

[22] M.J. Rosa, C.G. Ralha, M. Holanda, and A.P. Araujo, *Computational resource and cost prediction service for scientific workflows in federated clouds*, Future Generation Computer Systems **125** (2021), 844–858.
https://doi.org/10.1016/j.future.2021.07.030.

[23] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*, vol. 24, Springer, 2003.

[24] L. Sun, H. Dong, F.K. Hussain, O.K. Hussain, and E. Chang, *Cloud service selection: State-of-the-art and future research directions*, Journal of Network and Computer Applications **45** (2014), 134–150.
https://doi.org/10.1016/j.jnca.2014.07.019.

[25] Y. Tian, L. Wang, S. Yang, J. Ding, Y. Jin, and X. Zhang, *Neural network-based dimensionality reduction for large-scale binary optimization with millions of variables*, IEEE Transactions on Evolutionary Computation **29** (2024), no. 6, 2328–2342.
https://doi.org/10.1109/TEVC.2024.3400398.

[26] E.L. Ulungu and J. Teghem, *The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems*, Found. Comput. Decis. Sci. **20** (1995), no. 2, 149–165.

[27] T. Vincent, F. Seipp, S. Ruzika, A. Przybylski, and X. Gandibleux, *Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for the biobjective case*, Computers and Operations Research **40** (2013), no. 1, 498–509.
https://doi.org/10.1016/j.cor.2012.08.003.

[28] M. Visée, J. Teghem, M. Pirlot, and E.L. Ulungu, *Two-phases method and branch and bound procedures to solve the bi–objective knapsack problem*, J. Global Optim. **12** (1998), no. 2, 139–155.
https://doi.org/10.1023/A:1008258310679.

Appendix A: The initial data

This appendix provides the initial data for the three CSPs: Microsoft Azure, Google Cloud, and Amazon, see Table 6. For Microsoft Azure, 9 standard VM types were selected; for Google Cloud, 5 types; and for AWS, 7 types.

Table 6. **Data for Amazon, Microsoft Azure and Google clouds**

Data for Amazon cloud				
VM's name	Graph processor (GP) g_p	Memory m_p	Disk storage d_p	Cost c_p
T2.NANO	1Core	$0.5GB$	20GB	0.0081
T2.MICRO	1Core	1GB	40GB	0.0162
T2.SMALL	1Core	2GB	60GB	0.032
T2.MEDIUM	2Core	4GB	80GB	0.0644
T2.LARGE	2Core	8GB	160GB	0.1208
T2.XLARGE	4Core	16GB	320GB	0.2266
T2.2XLARGE	8Core	32GB	640GB	0.4332
Data for Microsoft Azure cloud				
VM's name	Graph Processor (GP) g_p	Memory m_p	Disk Storage d_p	Cost c_p
$B1s$	1Core	1GB	4GB	0.016
$B2s$	2Core	4GB	8GB	0.058
$B1ms$	1Core	2GB	4GB	0.029
$B2ms$	2Core	8GB	16GB	0.107
$B4ms$	4Core	16GB	32GB	0.214
$B8ms$	8Core	32GB	64GB	0.429
$B12ms$	12Core	48GB	96GB	0.643
$B16ms$	16Core	64GB	128GB	0.858
$B20ms$	20Core	80GB	160GB	1.072
Data for Google cloud				
VM's name	Graph processor (GP) g_p	memory m_p	Disk storage d_p	Cost c_p
e2-standard-2	2Core	8GB	10GB	0.067006
e2-standard-4	4Core	16GB	10GB	0.134012
e2-standard-8	8Core	32GB	10GB	0.268024
e2-standard-16	16Core	64GB	10GB	0.536048
e2-standard-32	32Core	128GB	10GB	1.072096