An Energy-Aware Combinatorial Auction-Based Virtual Machine Scheduling Model and Heuristics for Green Cloud Computing

Erbil Öner and Ali Haydar Özer*

Department of Computer Engineering,

Faculty of Engineering,

Marmara University, Istanbul, Turkey.

E-mail addresses: *erbiloner@marun.edu.tr*, *haydar.ozer@marmara.edu.tr*

Full Citation:

Erbil Öner, Ali Haydar Özer,

An energy-aware combinatorial auction-based virtual machine scheduling model and heuristics for green cloud computing,

Sustainable Computing: Informatics and Systems,

Volume 39, 100889

2023,

ISSN 2210-5379,

https://doi.org/10.1016/j.suscom.2023.100889.

*Corresponding author. Address: Department of Computer Engineering, Faculty of Engineering, Marmara University, 34854, Istanbul, Turkey. Tel: +90-216-777-3533 E-mail address: haydar.ozer@marmara.edu.tr

Abstract

Considering the increasing demand for cloud computing, and the financial and environmental impact of the increasing energy consumption trend of data centers, improving energy efficiency is vital for cloud service providers. In this study, an energy-aware virtual machine scheduling model is proposed which is based on the multi-unit nondiscriminatory combinatorial auction. The model includes a powerful bidding language that allows users to declare their complicated virtual machine requests using logical AND and OR relations along with the time constraints. The study also presents the formal definition of the model and the associated optimization problem for determining the optimum schedule and energy-efficient placement of VMs on physical servers. The optimization problem is formulated using integer linear programming and several heuristic solution methods including the Genetic Algorithm are proposed for this problem. The performances of the model and the proposed heuristics are assessed on a comprehensive test suite. The proposed model is estimated to provide approximately a 37% improvement in revenues, and the solution methods are estimated to provide high-quality solutions within only 5% of the optimum which enable the model to be deployed in large-scale clouds.

Keywords: Cloud Computing; Resource Scheduling; Combinatorial Auctions; Energy-Aware; Genetic Algorithm; Virtual Machine Placement.

1 Introduction

Cloud computing has an important place in the Information Technologies (IT) industry as it offers users an environment where they can develop and manage their applications and services without the need to set up their own IT infrastructure. Cloud service providers offer flexible and scalable environments with a pay-as-you-go pricing model that prevents users from over-provisioning or under-utilizing resources [1]. Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) models have been proposed to examine the cloud environment in detail which vary according to management and control capabilities [2].

Cloud computing systems consist of computing and storage resource pools in data centers that require large amounts of energy. The amount of energy consumed by data centers in the United States was reported as 70 billion kWh in 2014 [3]. Further studies estimate that the amount of energy required will increase with the growing demand for cloud computing systems, with the rate of increase depending on the success of the efficiency measures taken [4]. Currently, annual data center energy demand accounts for approximately 1% of global electricity consumption and is expected to reach approximately 4.5% by 2025 [5, 6]. The cost of this intensive energy use constitutes the biggest share in the budget of cloud service providers [7].

Beyond the financial impact, the environmental aspects of energy use should also be considered. Recent studies estimate that the share of data centers in the world's total carbon emissions is approximately 0.3% [8], and data centers are responsible for approximately 45% of the total carbon emissions of the ICT industry [9]. Considering this negative impact on global warming, reducing energy use and encouraging cloud service providers to adopt renewable energy sources are essential for sustainability. Major cloud service providers such as Google and Amazon have invested heavily in renewable energy in recent years, and both companies have declared that they aim to operate on a carbon-free energy ecosystem by 2030 [10, 11].

The most significant component of the energy use of a data center is the energy use of the ICT equipment which also indirectly affects the infrastructure energy cost. [12]. An effective way of reducing the energy consumption of physical servers is to increase their utilization levels [13–15] which can be achieved with the use of virtualization and consolidation technologies. With these methods, energy efficiency can be significantly increased by placing virtual machines (VMs) on the smallest number of physical servers possible, i.e. by consolidating VMs and putting the remaining idle physical servers into low-power mode, also known as sleep mode.

In this study, an Energy-Aware Combinatorial Auction-Based Virtual Machine Scheduling Model (ECO-CABS) is proposed for providing an energy-efficient scheduling solution for the cloud environment based on the combinatorial auction institution. In this scheduling model, users submit their requests as bids using the provided bidding language which allows users to request a set of VMs along with the duration requested. They can also declare a time window inside their bids so that the VMs specified in their bids are allocated to the users for the required duration in the specified time window. The proposed model is based on the powerful multi-unit nondiscriminatory combinatorial auction mechanism [16] which allows users to express their complex VM requests using logical AND and OR relations in a single bid. The objective of the model is to satisfy the maximal set of bids providing the highest profit for the cloud provider while considering the non-linear energy costs of the physical servers.

This study presents the formal definition of the ECO-CABS model along with an integer linear programming formulation of the associated optimization problem for outcome determination. Since the optimization problem is proven to be intractable, several heuristic solution methods including the Genetic Algorithm are proposed. The performance of the model and the quality of the solutions found by the proposed methods are demonstrated on a comprehensive test suite prepared using a novel test case generator designed for this model.

The paper is organized as follows: A brief literature review is provided in the next section. In Section 3, the ECO-CABS model is explained in detail using an example scenario. The mathematical formulation of the ECO-CABS model and the corresponding optimization problem are introduced in Section 4. The proposed heuristic methods are explained in Section 5 and the experimental results are discussed in Section 6. Finally, the paper is concluded in Section 7.

2 Literature Review

In this section, a brief survey of energy-aware VM allocation and scheduling methods proposed in the literature is presented. Beloglazov et al. [17] propose an energy-aware resource allocation and dynamic consolidation technique. Their first method, which consists of two main components, is responsible for the allocation of virtual machines, while their second method is responsible for the selection of virtual machines to be migrated. Rawas et al. [18], on the other hand, propose both online and offline virtual machine placement model called LECC which aims to reduce energy consumption and carbon footprint of geographically distributed data centers.

Ghribi et al. [19] introduce two algorithms using integer linear programming: an optimal energy-aware allocation algorithm which is solved as a bin packing problem for consolidation, and a migration algorithm to continuously optimize the number of used servers. The second algorithm uses the first one to reduce energy consumption. Zhu et al. [20] propose a three-dimensional virtual resource scheduling method in which the virtual resource allocation problem is considered as the multi-dimensional vector bin packing problem. They propose several heuristics for each of the allocation, scheduling, and optimization phases of their method.

Ding et al. [21] propose an energy-aware virtual machine scheduling method that satisfies deadline constraints. The method uses a performance-to-power ratio to rank physical machines while operating them at their optimal frequency level. In another study by Ruan et al. [22], the same performance-to-power ratio strategy is used for scheduling virtual machines. In this energy-efficient solution, the performance-to-power ratio of physical machines is based on a metric that is used in the SPECpower_ssj2008 [23] benchmark suite. The study also presents experimental results conducted using the CloudSim [24] platform. An energy-aware scheduling algorithm for scientific workflows has been introduced by Li et al. [25]. In this approach, the optimal virtual machine types are selected for scientific tasks. Following the virtual machine selection procedure, policies such as task merging and reuse of idle virtual machines are applied by the algorithm.

In the work of Chandio et al. [26], six energy-aware virtual machine strategies are proposed using dynamic voltage-frequency scaling technology with deadline constraints for minimizing the makespan. They present the effectiveness of these strategies on real-world high-performance workloads. Ghose et al. [27] introduce scheduling approaches for energy-aware scheduling of tasks with hard deadline constraints. Through several experiments, they estimate an energy-saving rate of 43% while satisfying the deadlines of every task available in the experiment.

Dai et al. [28] propose two approximation heuristics for the energy-aware virtual machine scheduling problem. The problem is formulated as an ILP and the approximation algorithms use the relaxed version of the ILP. Mishra et al. [29] have introduced an energy-efficient virtual machine allocation method that has a two-step allocation process where tasks are assigned to virtual machines and virtual machines to physical servers. The proposed method aims to increase energy efficiency using virtual machine consolidation.

In addition to the approximation and heuristic approaches, metaheuristic algorithms have also been used for the energy-aware virtual machine scheduling problem. Duan et al. [30] propose an energy-efficient virtual machine scheduler based on Ant Colony Optimization. The scheduler employs a prediction model along with an algorithm to provide effective host utilization in heterogeneous data centers. Tao et al. [31] formulate resource scheduling as a Multi-Objective Optimization Problem. Their approach adopts a hybrid Genetic Algorithm (GA) to find Pareto optimal solutions using the objectives of minimizing both makespan and energy usage. Another GA-based energy-aware scheduling approach has been introduced by Fernández-Cerero et al. [32]. The proposed approach not only aims to provide efficient utilization but also considers the security demands of the users. The policies defined in the study regulate the balance between the makespan and energy usage of the tasks. Lei et al. [33] address the problem of task scheduling in a data center that uses renewable energy sources along with conventional ones. The study proposes a multi-objective, co-evolutionary approach enhanced with an opposition-based learning strategy. In another learning-based study, Sharma and Garg [34] propose an energy-efficient task scheduling model using supervised neural networks. The primary objective of this model is to reduce makespan while also reducing energy usage. At the same time, it also aims to reduce the execution overhead and the number of active racks. The authors used an artificially created dataset to train the network and obtained 99.9% accuracy. The results of the experiments indicate approximately 60% improvement in makespan while reducing the energy consumption by approximately 45% for heavily loaded cloud environments.

Unlike the other studies using a centralized management system, Wang et al. [35] propose a decentralized multi-agent-based approach for the resource allocation problem. In this method, each agent is responsible for managing a physical server cluster. Instead of making virtual machine consolidation decisions through a central agent, these decisions are made based on negotiations between the agents. These agents negotiate and exchange virtual machines to minimize the total energy consumption of the system.

Some of the studies in the literature employ the Particle Swarm Optimization (PSO) metaheuristic for the resource scheduling problem. For instance, Beegom and Rajasree [36] propose a PSO-based technique that combines two minimization objectives into a single objective function using the weighted sum approach. The priority between minimizing the total execution time and cost can be decided using these weights. Similarly, Kumar and Sharma [37] have introduced a PSO-based scheduling approach in which energy consumption is also considered alongside these factors and all three are combined into a single objective function. Besides makespan and cost metrics, additional metrics like task rejection ratio and throughput are presented in this study.

Kessaci et al. [38] introduce two algorithms using a multi-start local search heuristic. One of them is modeled as a single-objective optimization problem while the other is modeled as a multi-objective optimization problem. Ilager et al. [39], on the other hand, have proposed a thermal and energy-aware algorithm based on the Greedy Randomized Adaptive Search Procedure metaheuristic. Their approach not only prevents the physical servers from operating at maximum load to maintain their temperature but also minimizes the number of underutilized physical servers by efficiently scheduling virtual machines.

In their recent works, Chen and Zhang [40] and Pradhan and Satapathy [41] both focus on optimizing task scheduling to minimize makespan and energy consumption. Chen and Zhang [40] propose a Diversity-Aware Marine Predators Algorithm (DAMPA) for task scheduling in cloud computing. This algorithm maintains population diversity and balances exploration and exploitation abilities. Two presented case experiments show that DAMPA provides a reduction in both makespan and energy consumption. Pradhan and Satapathy [41] introduce an Energy Aware Genetic Algorithm for task scheduling in a heterogeneous multi-cloud environment. They use a fitness function to minimize makespan and the energy consumption is calculated using the optimal objective value.

He et al. [42], and Tarafdar et al. [43] propose task scheduling methods to address energy consumption and deadline constraints. He et al. [42] propose a two-stage scheduling method for deadline-constrained tasks, combining Enhanced Ant Colony Optimization (EACO) with the Modified Backfilling (MBF) algorithm. Their method effectively reduces makespan and energy consumption while increasing the task completion rate. Tarafdar et al. [43] present two scheduling approaches for deadline-sensitive tasks in a heterogeneous cloud environment: a greedy heuristic based on the Linear Weighted Sum technique and an Ant Colony Optimization-based method. They also propose a strategy for scaling cloud resources to improve energy efficiency and the schedulability of tasks.

Liu et al. [44] suggest a greedy scheduling strategy based on granular computing. The authors present numerical experiments on the CloudSim platform demonstrating an increase in resource utilization and a reduction in energy consumption. Hussain et al. [45] propose the Energy and Performance-Efficient Task Scheduling Algorithm (EPETS) in a heterogeneous virtualized cloud designed as a two-stage algorithm. In the first stage, the algorithm aims to reduce makespan while satisfying the deadline constraints, and in the second stage, it aims to reduce energy consumption without violating deadline constraints. Walia et al. [46] introduce a Hybrid Scheduling Algorithm (HS) based on the Genetic Algorithm (GA) and Flower Pollination Algorithm (FPA) for cloud environments. The results presented indicate that the proposed algorithm outperforms each of the GA and FPA in terms of resource utilization, completion time, and energy consumption for both homogeneous and heterogeneous environments.

Ye et al. [47] propose a reliability-aware and energy-efficient workflow scheduling algorithm called REWS, which aims to reduce energy consumption while satisfying workflow reliability constraints. The algorithm divides the workflow reliability constraint into task sub-reliability constraints and schedules tasks with an energy-efficient scheduling strategy. Similarly, Konjaang et al. [48] present an energy-efficient virtual machine mapping algorithm (EViMA) for workflows with deadlines in a cloud environment. EViMA aims to find an effective schedule that reduces cloud data center energy consumption, execution makespan, and execution cost and hence improves resource management in clouds.

Bugingo et al. [49] also tackle the deadline-constrained workflow scheduling problem in the cloud. However, they propose a multiobjective mechanism and investigate the relationship between the objectives of cost minimization and energy consumption minimization. They propose an algorithm with two variants to satisfy both users and providers during the configuration selection process. Based on the evaluation results, significant reductions in energy consumption and cost can be obtained by using the proposed heuristic. Similarly, Tarafdar et al. [50] present a novel workflow scheduling approach for the Workflow as a Service (WaaS) platform that aims to reduce the mean workflow execution time, increase energy efficiency, and reduce renting costs of the resources in cloud environments. In a related study, Alsadie [51] presents a metaheuristic framework called MDVMA for dynamic virtual machine allocation in clouds. The MDVMA aims to reduce energy usage, makespan, and cost simultaneously by using a non-dominated sorting genetic algorithm (NSGA-II) algorithm-based metaheuristic approach for multi-objective task scheduling. Sahoo et al. [52] present another approach with dual objectives (the minimization of makespan and energy usage). They propose a learning automata-based scheduling framework and the LA-based Scheduling (LAS) algorithm for deadline-sensitive task scheduling. They demonstrate the effectiveness of their approach using simulations.

Auction-based methods have also been proposed for the allocation and scheduling of VMs in clouds. Prodan et al. [53] have proposed a Continuous Double Auction (CDA) model for scheduling scientific tasks. In their work, tasks are modeled as a directed acyclic graph. A negotiation mechanism based on the CDA model is established between the sellers and bidders to determine the prices of the resources to be used, and scheduling decisions are optimized in terms of execution cost and time. Another auction-based virtual machine scheduling algorithm has been proposed by Kong et al. [54], which takes auction deadlines and network bandwidth into account while ensuring effective resource utilization.

Finally, in their studies, Gamsız and Özer [55, 56] have proposed a combinatorial auction-based VM allocation and placement model called the ECO-VMAP model that includes nonlinear energy usage costs of physical servers. Similar to the ECO-CABS model proposed in this study, the ECO-VMAP model also utilizes multi-unit nondiscriminatory combinatorial auction institution [16] for allocating or renting a set of virtual VMs for a predefined and fixed interval. It introduces the concept of slots for physical servers where each physical server is assumed to be preconfigured for a predefined VM type and can have one or more slots in which a single VM of the predefined VM type can be placed.

The ECO-CABS model proposed in this study complements the ECO-VMAP model by addressing the energy-aware resource scheduling problem in clouds. The bidding language is improved to enable the users to declare their preferences for virtual resources along with the duration they request these resources and to declare a time window in which they request these virtual resources. The energy model introduced in the ECO-VMAP model requires that the total energy consumption of *each slot* of a physical server should be known for the rental interval. The energy model of the ECO-CABS model is simplified both in terms of easy applicability and modeling complexity without sacrificing its effectiveness. The ECO-CABS model requires only idle and maximum power values required for a server which are already available in the technical specifications of most of the physical servers (also available in the SPECpower_ssj® benchmark [23]). Being a scheduling model, and having an improved bidding language and an energy model, the optimization problem of the ECO-CABS model is significantly different from the optimization model of the ECO-VMAP model. Hence, the solution methods and the experiments are also significantly different.

For further discussion on energy-aware task scheduling approaches, the reader is referred to the recent survey of Ghafari et al. [57].

3 The ECO-CABS Model

In this section, the ECO-CABS model will be introduced on an example scenario. There are three steps in the model. The first step is to define physical servers and VM types in a cloud environment. In this scenario, seven VM configurations and one physical server instance for each VM configuration have been defined. The list of physical servers and the VM types can be seen in Table 1 and in Table 2, respectively. The VM configurations differ for various computing requirements. For example, configuration GP represents general-purpose VM instances that are optimized to run basic applications, while CO represents compute-optimized VM instances that require high-performance CPUs. MO and SO represent memory-optimized and storage-optimized VM instances, respectively.

Table 1: List of physical server instances used in the example scenario, including their configurations and power requirements. Power requirements of the reference servers are obtained from SPECpower_ssj® benchmark [23]. Note that these values may not reflect the true power requirements of these servers.

Server Instance	Reference Server	Supported VM Configuration	Max VM Size That Can Be Hosted	Idle Power (W)	Full Power (W)
m_1	Dell PowerEdge R6515	GP-1	48	55	231
m_2	ASUSTeK Computer RS700A-E9-RS4V2	GP-2	96	106	430
m_3	Lenovo ThinkSystem SR860V2	CO-1	96	138	721
m_4	HP Enterprise Superdome Flex 280	CO-2	96	258	1117
m_5	Inspur Corporation Inspur NF5180M6	MO-1	48	156	680
m_6	Supermicro SuperServer SYS-740GP-TNRT	MO-2	64	118	633
m_7	Fujitsu PRIMERGY RX2540 M6	SO-1	64	137	550

Table 2: Base virtual machine instance types (of size 1x) used in the example scenario.

VM Type	VM Configuration	Base Virtual CPUs	Base Memory	Reservation Price per Time Slot
v_1	GP-1	1	1 GB	\$0.041
v_2	GP-2	1	2 GB	\$0.050
v_3	CO-1	1	4 GB	\$0.066
v_4	CO-2	1	8 GB	\$0.076
v_5	MO-1	1	8 GB	\$0.100
v_6	MO-2	1	10 GB	\$0.120
v_7	SO-1	1	4 GB	\$0.130

Physical server instances are preconfigured to run a specific virtual machine type. Depending on the computing capacity of the physical server instances, the maximum size of the virtual machine instances that can be executed on a physical server instance varies. In this scenario, seven types of virtual machine configurations are defined as seen in Table 2. Each VM type is associated with a number of virtual CPUs and an amount of memory which indicates the *base* size of the corresponding VM type. Users can, however, request various sized VMs of any VM type where the size of a VM denotes the multiplier of the base computing power defined in Table 2. For instance, if a user requests VM type v_4 of size 3.25x then the VM the user requests will have (3.25 x 1 =) 3.25 virtual CPUs and (3.25 x 4 =) 13 GB of memory. Note that the size of a VM can also be less than 1.

In this scenario, the scheduling period is defined as 10 time slots (e.g. 10 days or 10 weeks), that is, VMs will be allocated to the users within this time frame. In other words, each server has 10 discrete time slots in which one or more virtual machines can be scheduled.

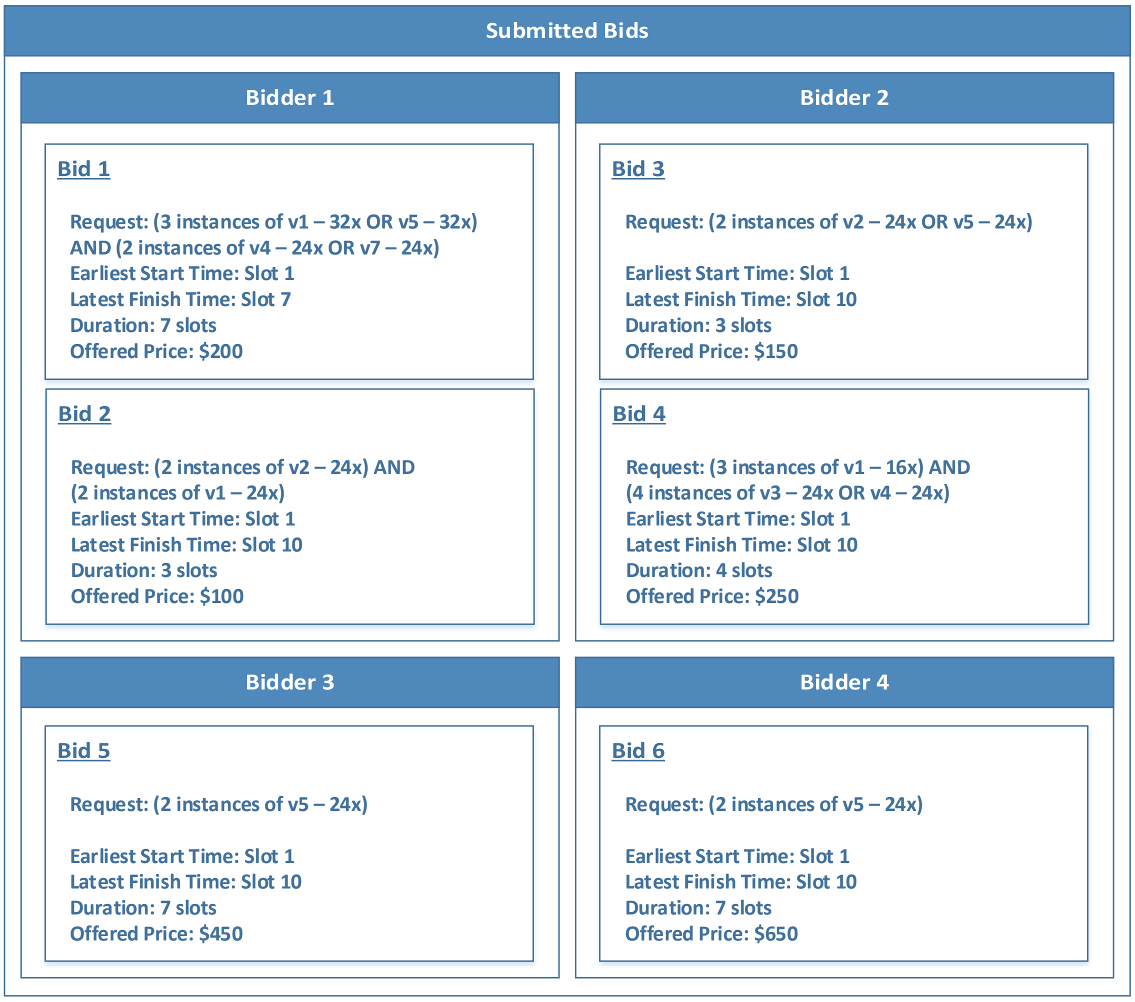
Note that in the ECO-CABS model, the service provider may declare a *reservation* *price* for each VM type which indicates *the minimum profit* that the provider wants to make when 1x size of the corresponding VM type is scheduled for one time slot. Referring to the example values in Table 2, if a VM type v_1 of size 24x is scheduled for 3 time slots, then the reservation price for this VM will be ($0.041 x 24 x 3 =) $2.95.

The second step in the model is to collect bids from users. The ECO-CABS model is based on the multi-unit non-discriminatory combinatorial auction mechanism defined by Özer and Özturan [16]. By virtue of this mechanism, users can express their complex requests in their bids using logical OR and logical AND relations. The logical OR relation allows users to request alternative virtual machine types in their bids whereas virtual machines that need to run simultaneously can be easily specified using the logical AND relation. Along with the VM requests, users can also state the duration of the allocation, i.e. the requested number of time slots, and restrict the time frame in which the VMs will be scheduled, i.e. they can declare the earliest start and latest finish time slots for the schedule.

The bids submitted by the bidders (users) for the example scenario are illustrated in Figure 1. There are four bidders submitting six bids. Bid 1 submitted by Bidder 1 consists of two subbids. In the first subbid, Bidder 1 requests three instances of VM type v_1 of size 32x *or* v_5 of size 32x. In the ECO-CABS model, the virtual machines requested in a subbid are defined as alternatives to each other, that is there is a logical OR relationship between the VM types requested inside a subbid. Therefore, the first subbid can be *satisfied* if Bidder 1 gets one of the following configurations:

- Three instances of $v_1 - 32x$
- Two instances of $v_1 - 32x$ and one instance of $v_5 - 32x$
- One instance of $v_1 - 32x$ of size 32x and two instances of $v_5 - 32x$
- Three instances of $v_5 - 32x$

Since the size of a VM type denotes the multiplier of the base computing power of the corresponding VM type, v_1 instances scheduled for this bidder will have (32 x 1 =) 32 virtual CPUs and (32 x 1 =) 32 GB of memory, whereas v_5 instances scheduled for this bidder will have (32 x 1 =) 32 virtual CPUs and (32 x 8 =) 256 GB of memory.



Submitted Bids

Bidder 1

Bid 1

Request: (3 instances of v1 – 32x OR v5 – 32x)
AND (2 instances of v4 – 24x OR v7 – 24x)
Earliest Start Time: Slot 1
Latest Finish Time: Slot 7
Duration: 7 slots
Offered Price: $200

Bid 2

Request: (2 instances of v2 – 24x) AND
(2 instances of v1 – 24x)
Earliest Start Time: Slot 1
Latest Finish Time: Slot 10
Duration: 3 slots
Offered Price: $100

Bidder 2

Bid 3

Request: (2 instances of v2 – 24x OR v5 – 24x)

Earliest Start Time: Slot 1
Latest Finish Time: Slot 10
Duration: 3 slots
Offered Price: $150

Bid 4

Request: (3 instances of v1 – 16x) AND
(4 instances of v3 – 24x OR v4 – 24x)
Earliest Start Time: Slot 1
Latest Finish Time: Slot 10
Duration: 4 slots
Offered Price: $250

Bidder 3

Bid 5

Request: (2 instances of v5 – 24x)

Earliest Start Time: Slot 1
Latest Finish Time: Slot 10
Duration: 7 slots
Offered Price: $450

Bidder 4

Bid 6

Request: (2 instances of v5 – 24x)

Earliest Start Time: Slot 1
Latest Finish Time: Slot 10
Duration: 7 slots
Offered Price: $650

Figure 1: Submitted bids in the example scenario.

In the second subbid of Bid 1, Bidder 1 requests two instances of v_4 of size 24x *or* v_7 of size 24x. Similar to the first subbid, the second subbid can be *satisfied* if Bidder 1 gets one of the following configurations:

- Two instances of $v_4 - 24x$
- One instance of $v_4 - 24x$ and one instance of $v_7 - 24x$
- Two instances of $v_7 - 24x$

In the ECO-CABS model, there is a logical AND relationship between subbids, and therefore for a bid to be *satisfied*, all of the subbids inside the bid should be satisfied simultaneously.

In Bid 1, Bidder 1 also requests VMs for a duration of 7 slots, and the earliest start slot is Slot 1 and the latest finish time is Slot 7. For this bid, Bidder 1 offers \$200, that is the bidder is willing to pay this amount if requested VMs in both subbids are allocated to the bidder for 7 time slots, beginning at Slot 1 and ending at Slot 7. Note that declaring the earliest start and the latest finish times is optional in the ECO-CABS model. If a bidder does not declare these values in a bid, simply the whole scheduling period is used for that bid.

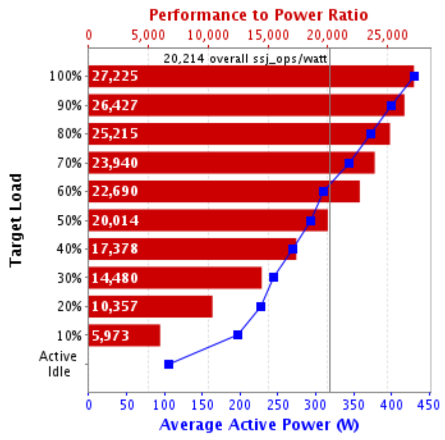
Further note that Bidder 1 also submits a second bid, Bid 2, which is independent of Bid 1 in this scenario. Thus, in the outcome, none, one, or both of these bids can be satisfied. The ECO-CABS model also allows bidders to declare XOR relations between their bids. That is, the model allows Bidder 1 to request that at most one of these bids be satisfied.

The bids in this scenario are formally defined using the bidding language of the model as follows:

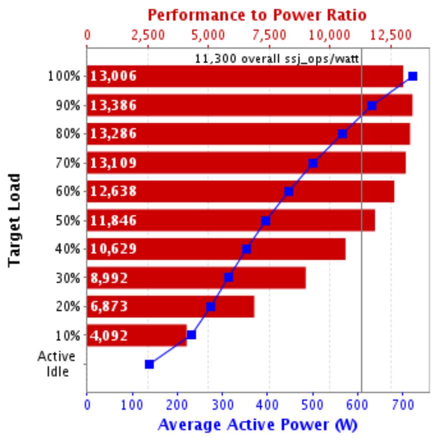
- $b_1 = \langle\{(\{v_1 - 32x, v_5 - 32x\}, 3), (\{v_4 - 24x, v_7 - 24x\}, 2)\}, 1, 7, 7, \$200\rangle$
- $b_2 = \langle\{(\{v_2 - 24x\}, 2), \{v_1 - 24x\}, 2)\}, 1, 10, 3, \$100\rangle$
- $b_3 = \langle\{(\{v_2 - 24x, v_5 - 24x\}, 2)\}, 1, 10, 3, \$150\rangle$
- $b_4 = \langle\{(\{v_1 - 16x\}, 3), (\{v_3 - 24x, v_4 - 24x\}, 4)\}, 1, 10, 4, \$250\rangle$
- $b_5 = \langle\{(\{v_5 - 24x\}, 2)\}, 1, 10, 7, \$450\rangle$
- $b_6 = \langle\{(\{v_5 - 24x\}, 2)\}, 1, 10, 7, \$650\rangle$

The bidding language of the ECO-CABS model defines that each bid consists of a non-empty set of subbids, the earliest start time, the latest finish time, the duration of allocation, and the offered price. Each subbid, on the other hand, consists of a non-empty set of VM types including their size and the number of requested VMs. The formal definition of the ECO-CABS model is given in Section 4.

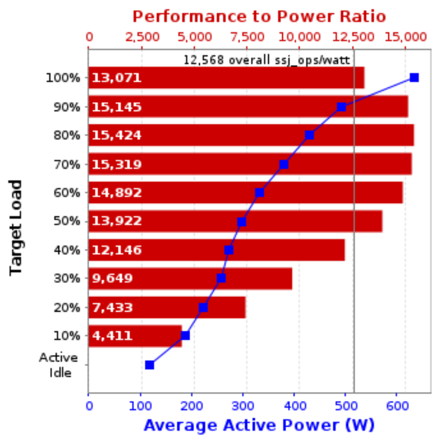
In addition to the powerful bidding language which allows bidders to express their complex preferences for VMs, another feature of the model is to provide an energy-aware mapping of VMs to the physical servers to increase the energy efficiency of cloud data centers. Different metrics are used in the literature to measure and monitor the energy efficiency in data centers [58]. The most important of these metrics is the Power Usage Effectiveness (PUE) metric [59]. This metric refers to the ratio of the total facility energy in a data center to the energy used by the IT equipment. Apart from the physical servers in data centers, other factors such as cooling and power distribution units also consume energy [60]. Using the PUE metric, one can estimate the energy use of data centers by the energy use of physical servers. Therefore, only the power usage characteristics of physical servers will be taken into account in this study to increase the energy efficiency of data centers.



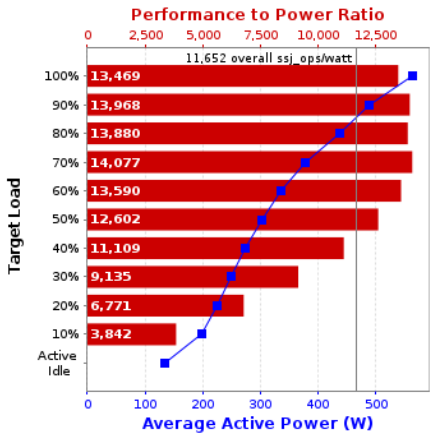
(a) ASUSTeK Computer Inc. RS700A-E9-RS4V2



(b) Lenovo Global Technology ThinkSystem SR860V2



(c) Supermicro Inc. SuperServer SYS-740GP-TNRT

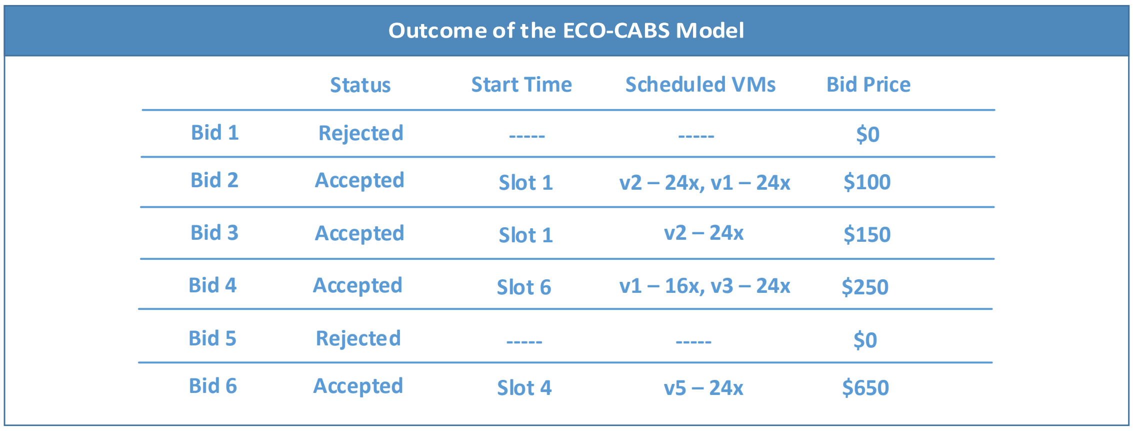


(d) Fujitsu PRIMERGY RX2540 M6

Figure 2: Power usage characteristics of the reference physical servers used in the example scenario. The graphs are obtained from SPECpower_ssj® benchmark [23].

The power usage characteristic of a physical server varies according to the computing components it contains, such as processors, memories, hard disks, solid-state disks, and network interface cards. As the utilization level of a physical server increases, the power it consumes also increases. Also, physical servers consume a significant amount of power even when they are idle. This can be seen in Figure 2 which shows the detailed power usage characteristics of the four of the physical servers selected for this example scenario. With the help of the virtualization technology, virtual machines can be consolidated, i.e. placed on as few physical machines as possible, and physical servers that are idle can be shut down or put into low power mode (sleep mode).

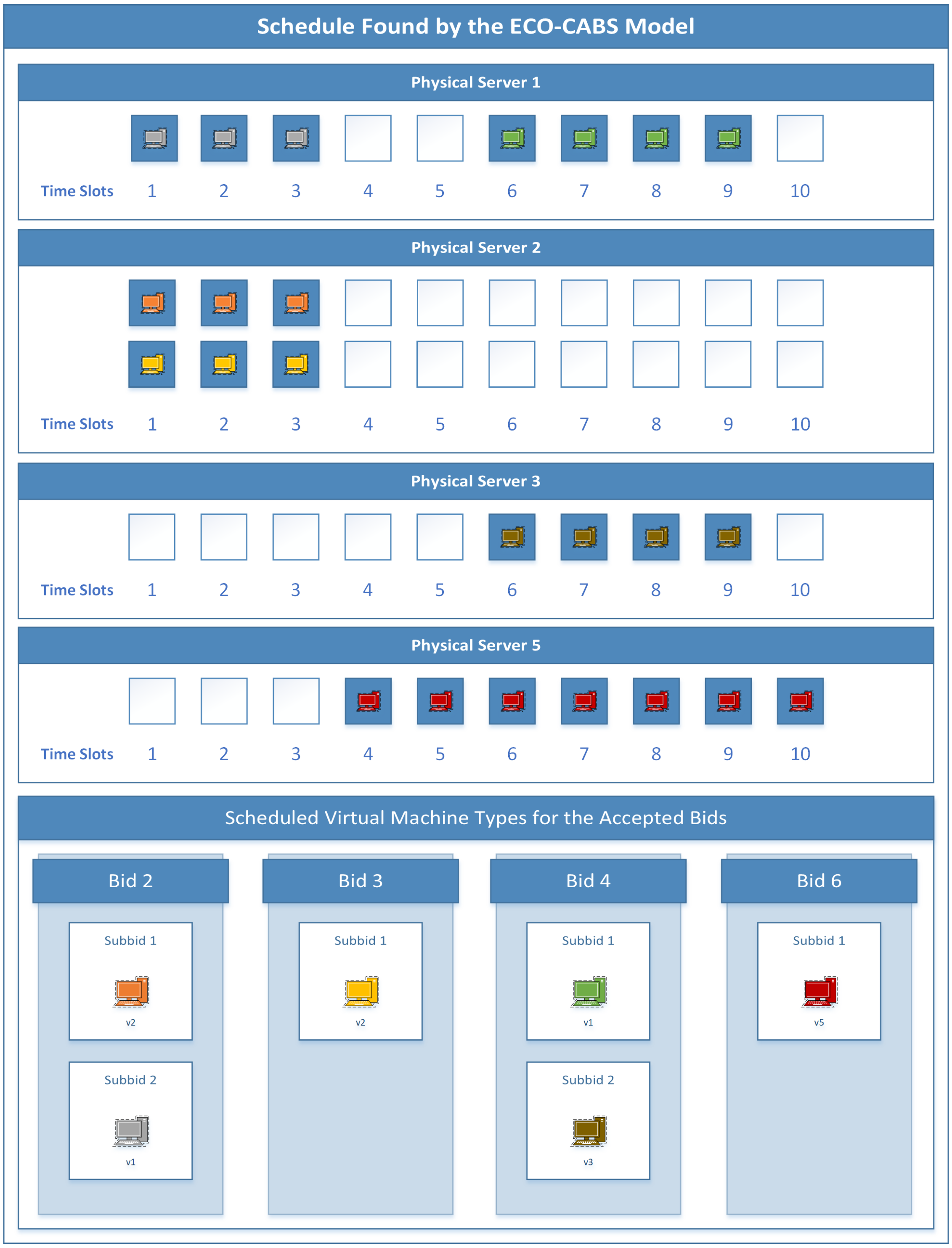
The ECO-CABS model places the requested VMs to the physical servers while considering the energy characteristics of the physical servers. For this purpose, the model uses an energy function that takes into account both idle power and utilization levels of the physical servers. Thus, the model automatically performs the consolidation of VMs while keeping the utilization levels of the physical servers at the optimum level, and hence the model finds the optimal placement of the VMs to the physical servers.


Outcome of the ECO-CABS Model

	Status	Start Time	Scheduled VMs	Bid Price
Bid 1	Rejected	-----	-----	$0
Bid 2	Accepted	Slot 1	v2 – 24x, v1 – 24x	$100
Bid 3	Accepted	Slot 1	v2 – 24x	$150
Bid 4	Accepted	Slot 6	v1 – 16x, v3 – 24x	$250
Bid 5	Rejected	-----	-----	$0
Bid 6	Accepted	Slot 4	v5 – 24x	$650

Figure 3: Outcome of the ECO-CABS model for the example scenario.

The third step in the ECO-CABS model is to determine the outcome. For this purpose, an optimization problem details of which will be introduced in Section 4 is solved. The outcome of the ECO-CABS model for the example scenario can be seen in Figure 3 and the schedule found can be seen in Figure 4. In the outcome, bids 2, 3, 4, and 6 are satisfied. It is impossible to satisfy Bid 1 since three virtual machine instances of size 32x cannot be placed on either physical server m_1 or physical server m_5 both of which have a maximum VM capacity of 48 (See Table 1). Since bids 2 and 3 request the same virtual machine type (v_2) and the requested sizes of 24x allow them to be executed together on a single physical



Schedule Found by the ECO-CABS Model
Physical Server 1
Time Slots 1 2 3 4 5 6 7 8 9 10
Physical Server 2
Time Slots 1 2 3 4 5 6 7 8 9 10
Physical Server 3
Time Slots 1 2 3 4 5 6 7 8 9 10
Physical Server 5
Time Slots 1 2 3 4 5 6 7 8 9 10
Scheduled Virtual Machine Types for the Accepted Bids
Bid 2
Subbid 1
v2
Subbid 2
v1
Bid 3
Subbid 1
v2
Bid 4
Subbid 1
v1
Subbid 2
v3
Bid 6
Subbid 1
v5

Figure 4: Schedule of VMs found by the ECO-CABS model for the example scenario.

server, the model has scheduled both requests for the same time slots on physical server 2, consolidating the VMs and hence minimizing the energy consumption. Among the fifth and sixth bids which request the same VM type of maximum size, the latter which has a higher offered price has been selected by the model.

Although a generic scenario is provided in this section, the bidding language of the ECO-CABS model can capture the preferences of the users in a variety of scenarios in which multiple VMs are needed. Using the instances available in the Amazon Elastic Compute Cloud [61] for instance, some bid examples for a set of well-known application types are as follows. Note that scheduling and price details are not provided in the example bids for clarity.

- Load balancing: In situations where an application experiences high traffic or resource demands, multiple VMs can be used to distribute the load evenly across them. For instance, to request four t3.medium instances in the us-east-1 (North Virginia) region, and use an Application Load Balancer (ALB), the following bid can be submitted:

$$b = \{(\text{us-east-1-t3.medium}, 4), (\text{us-east-1-ALB}, 1)\}$$

- High availability and redundancy: To minimize downtime and ensure continuous operations, multiple VMs can be deployed in different availability zones or regions. For instance, to request two m5.large or m5.xlarge instances in the us-west-1 (Northern California) region, and another two m5.large or m5.xlarge instances in the us-west-2 (Oregon) region to ensure high availability, the following bid can be submitted:

$$\begin{aligned} b = \{&(\text{us-west-1-m5.large, us-west-1-m5.xlarge}, 2), \\ &(\text{us-west-2-m5.large, us-west-2-m5.xlarge}, 2)\} \end{aligned}$$

- Microservices architecture: When an application is designed using microservices, each service can run on a separate VM. This setup allows for independent scaling and management of each service. For instance, for an application with three microservices, to request one t3.small instance for each service in the eu-central-1 (Frankfurt) region or in the ap-southeast-1 (Singapore) region, the following bid can be submitted:

$$b = \{(\text{eu-central-1-t3.small, ap-southeast-1-t3.small}, 3)\}$$

- Disaster recovery: To protect against data loss and minimize downtime in the event of a disaster, organizations may use multiple VMs in geographically diverse locations. These VMs can be used as failover or backup systems. For instance, to request a db.m5.large or db.m5.xlarge instance in the ap-northeast-1 (Tokyo) region for a primary database and another db.m5.large or db.m5.xlarge instance in the ap-northeast-2 (Seoul) region for backup:

$$\begin{aligned} b = \{&(\text{ap-northeast-1-db.m5.large, ap-northeast-1-db.m5.xlarge}, 1),\\ &(\text{ap-northeast-2-db.m5.xlarge, ap-northeast-2-db.m5.xlarge}, 1)\} \end{aligned}$$

- Multi-tier architecture: In multi-tier application architectures, separate VMs can be used for each tier (e.g., web, application, and database layers). For a three-tier application, to request two t3.small or t3.medium instances for the web layer, three t3.medium or t3.large instances for the application layer, and two r5.large or r5.xlarge instances for the database layer, all within the us-east-2 (Ohio) region, the following bid can be submitted:

$$\begin{aligned} b = \{&(\text{us-east-2-t3.small, us-east-2-t3.medium}, 2),\\ &(\text{us-east-2-t3.medium, us-east-2-t3.large}, 3),\\ &(\text{us-east-2-r5.large, us-east-2-r5.xlarge}, 2)\} \end{aligned}$$

4 Formal Definition of the ECO-CABS Model

The ECO-CABS model is defined formally as follows:

- $M = \{m_1, m_2, ..., m_m\}$ is the set of m physical machines available in all data centers of a cloud where each machine may have a different computational capacity, and hence a different power requirement;
- $V = \{v_1, v_2, ..., v_d\}$ is the set of d VM types to be scheduled where a VM type defines

the base configuration of a VM instance and a VM instance is characterized by both its type and a size parameter that is the multiplier of the base configuration features;

- the function $\Theta : M \rightarrow V$ is the mapping function that determines the VM type that each physical machine m_a is configured for $(m_a \in M, \Theta(m_a) \in V)$;
- u_a denotes the maximum capacity of physical machine m_a, that is the how many base VMs of type $\Theta(v_d)$ that physical machine m_a can host while satisfying Quality-of-Service requirements;
- the function $\Omega : V \rightarrow 2^M$ gives the subset of physical machines that are configured to execute VM type v_k where 2^M is the power set of M $(v_k \in V, \Omega(v_k) \subseteq M)$;
- the scheduling period consists of T time slots;
- $B = \{b_1, b_2, ..., b_n\}$ is the set of n submitted bids where a bid b_i is defined as a five tuple $b_i = (S_i, e_i, l_i, d_i, p_i)$, where
  - S_i is the set of subbids of the bid b_i where each subbid $s_{ij} \in S_i$ is a pair $s_{ij} = (H_{ij},\, q_{ij})$ such that $H_{ij} = \{(v_{ij1},\, g_{ij1}), (v_{ij2},\, g_{ij2}), \ldots (v_{ijr},\, g_{ijr})\}$ is the set of r requested alternative VM type-size pairs, q_{ij} is the number of VM instances requested in the subbid s_{ij}, and each pair $(v_{ijk},\, g_{ijk}) \in H_{ij}$ denotes the requested alternative VM type $v_{ijk} \in V$ and its requested size $g_{ijk} \in \mathbb{R}^+$;
  - e_i the earliest start slot and l_i is the latest finish slot of the bid b_i $(1 \leqslant e_i \leqslant l_i \leqslant T, e_i, l_i \in \mathbb{Z}^+)$,
  - d_i is the requested number of time slots for the bid b_i $(1 \leqslant d_i \leqslant l_i - e_i + 1, d_i \in \mathbb{Z}^+)$,
  - and p_i is the offered price for the bid b_i $(p_i \in \mathbb{R}^+)$.
- $\mathcal{E}_{idle}(m_a, t)$ denotes the energy cost of operating the physical machine m_a for the time slot t when it is idle;
- $\mathcal{E}_{full}(m_a, t)$ denotes the energy cost of operating the physical machine m_a for the time slot t when it is fully utilized;

- $\mathcal{P}_{res}(v_d, g_d)$ denotes *the reservation price*, that is the minimum price (excluding the energy costs) that the cloud provider requests for scheduling one instance of VM type v_d having size g_d for one time slot.

Note that in this definition,

- a is the index of each physical machine m_a;
- d is the index of each virtual machine v_d;
- i is the index of each bid b_i;
- j is the index of each subbid s_{ij} of bid b_i;
- k is the index of each pair $(v_{ijk},\, g_{ijk})$ of subbid s_{ij} of bid b_i;
- t is the index of each time slot in T.

A bid $b_i = (S_i, e_i, l_i, d_i, p_i)$ is *satisfied* when all of its subbids in S_i are satisfied. On the other hand, a subbid $s_{ij} = (H_{ij},\, q_{ij})$ is *satisfied* when q_{ij} VMs among the requested alternative VM types listed in $H_{ij} = \{(v_{ij1},\, g_{ij1}), (v_{ij2},\, g_{ij2}), \ldots (v_{ijr},\, g_{ijr})\}$, are allocated for d_i time slots between time slots e_i and l_i.

The outcome determination problem (ODP) of the ECO-CABS model is defined as determining a maximal set of mutually satisfiable bids and the corresponding schedule for VMs that maximizes the expected profit of the cloud provider while providing energy-aware mapping of VMs to physical machines during the defined scheduling period.

To formulate the ODP of the ECO-CABS model using integer linear programming, the following decision variables are introduced:

$$x_i = \begin{cases} 1, & \text{if bid } b_i \text{ is satisfied} \\ 0, & \text{otherwise} \end{cases}$$

$$y_i^t = \begin{cases} 1, & \text{if the reservation period of the VMs requested in bid } b_i \text{ begins at time slot } t \text{ if the bid is satisfied} \\ 0, & \text{otherwise} \end{cases}$$

$$o^{at} = \begin{cases} 1, & \text{if physical machine } m_a \text{ is active at time slot } t \text{ (i.e., if there exists a VM assigned to the physical machine } m_a \text{ at time slot } t) \\ 0, & \text{otherwise} \end{cases}$$

z_{ijk}^{at} = the number of VMs of VM type v_{ijk} requested in subbid s_{ij} of bid b_i that begins at time slot t on physical machine m_a

Then, the ODP is formulated as follows:

$$\begin{aligned} \max \quad & \sum_{b_i \in B} p_i \, x_i \\ & - \sum_{m_a \in M} \sum_{t=1}^{T} \mathcal{E}_{idle}(m_a, t) \, o^{at} \\ & - \sum_{t=1}^{T} \sum_{m_a \in M} \sum_{\substack{b_i \in B, \\ e_i \leqslant t \leqslant l_i}} \sum_{\substack{s_{ij} \in S_i, \\ (v_{ijk}, g_{ijk}) \in H_{ij}, \\ v_{ijk} = \Theta(m_a)}} \sum_{t'=max(e_i, t-d_i+1)}^{min(l_i-d_i+1, t)} \frac{(\mathcal{E}_{full}(m_a, t) - \mathcal{E}_{idle}(m_a, t)) \; g_{ijk}}{u_a} \, z_{ijk}^{at'} \\ & - \sum_{t=1}^{T} \sum_{m_a \in M} \sum_{\substack{b_i \in B, \\ e_i \leqslant t \leqslant l_i}} \sum_{\substack{s_{ij} \in S_i, \\ (v_{ijk}, g_{ijk}) \in H_{ij}, \\ v_{ijk} = \Theta(m_a)}} \sum_{t'=max(e_i, t-d_i+1)}^{min(l_i-d_i+1, t)} \mathcal{P}_{res}(v_{ijk, g_{ijk}}) \, z_{ijk}^{at'} \end{aligned} \tag{1}$$

$$\text{s.t.} \quad \left(\sum_{t=e_i}^{l_i-d_i+1} y_i^t \right) - x_i = 0 \quad (b_i \in B) \tag{2}$$

$$\left(\sum_{(v_{ijk}, g_{ijk}) \in H_{ij}} \sum_{m_a \in \Omega(v_{ijk})} z_{ijk}^{at} \right) - q_{ij}\, y_i^t = 0 \quad (b_i \in B, s_{ij} \in S_i, e_i \leqslant t \leqslant l_i - d_i + 1) \tag{3}$$

$$\sum_{\substack{b_i \in B, \\ e_i \leqslant t \leqslant l_i}} \sum_{\substack{s_{ij} \in S_i, \\ (v_{ijk}, g_{ijk}) \in H_{ij}, \\ v_{ijk} = \Theta(m_a)}} \sum_{t' = max(e_i, t-d_i+1)}^{min(l_i-d_i+1, t)} g_{ijk}\, z_{ijk}^{at'} \leqslant u_a\, o^{at} \quad (m_a \in M, 1 \leqslant t \leqslant T) \tag{4}$$

$$x_i,\, y_i^t,\, o^{at} \in \{0, 1\} \quad (\forall a, i, t) \tag{5}$$

$$z_{ijk}^{at} \in \mathbb{Z}^+ \cup \{0\} \quad (\forall a, i, j, k, t) \tag{6}$$

In this formulation, Eq.(1) is the objective function that has four components. The first component maximizes the expected revenue of the cloud provider that is the sum of the offered bid prices. The second and third components of the objective function minimize the energy cost of operating the physical servers by providing consolidation of VM instances. Note that if the electricity rate is constant throughout the scheduling period, then the parameter t can be omitted from both energy cost functions $\mathcal{E}_{idle}$ and $\mathcal{E}_{full}$. Finally, the fourth component ensures that the bids exceeding the minimum reservation price (excluding the energy costs) that the cloud provider requests for VM instances can be satisfied. Again, note that the costs other than the energy costs can be included in the reservation price for each VM instance.

Regarding the constraints, the first constraint in Eq.(2) ensures that if a bid is satisfied then the reservation period for allocated VMs should fit inside the requested time window. The second constraint in Eq.(3) enforces that all the subbids of a satisfied bid should also be satisfied, that is for each subbid the requested number of VMs listed inside the subbid should be scheduled such that all the VMs start at the same time slot which is inside the requested time window. Finally, the allocated VMs for all bids are assigned to the physical machines in the third constraint defined in Eq.(4). This constraint also determines the active physical machines to which at least one VM is assigned for proper consolidation of VMs.

The ODP of the ECO-CABS model is a generalization of the winner determination problem of the multi-unit nondiscriminatory combinatorial auction institution defined in

[16] which is proven to be NP-hard. Therefore, the ODP is also NP-hard.

5 Solution Methods

The optimization problem, the ODP, of the ECO-CABS model is defined using integer linear programming, which can be solved by general-purpose Mixed-Integer Programming (MIP) solvers. However, since the ODP is proven to be NP-Hard, it may not be possible to find optimal solutions in a reasonable time for medium and large-scale problem instances. Therefore, various heuristics are designed in this study to find high-quality solutions in a reasonable time. In this section, these heuristic methods are presented in detail.

The source code of the solution methods proposed in this study can be reached from [62].

5.1 Heuristic Methods

The following heuristic methods are proposed for the ODP of the ECO-CABS model:

i. Greedy Placement Heuristic

ii. Single Bid Placement Heuristic

iii. Multiple Bid Placement Heuristic

iv. Genetic Algorithm

In each of these methods, the bids submitted are processed one by one or in groups. For this reason, the order of the bids affects the performance of the methods. Therefore, the following sorting heuristics are also proposed to be used in these heuristic methods:

Sorting Heuristic H_1 - Average Profit Considering the Reservation Price: In this heuristic, the bids are sorted according to the amount of profit that the virtual machines requested in the bids will provide per time slot in descending order. Note that in this sorting heuristic only the virtual machine with the lowest reservation price listed in the subbids is included for determining the cost. The heuristic value H_1 of a bid b_i is calculated by subtracting this cost value from the offered price per time slot as follows:

$$H_1(b_i) = \frac{p_i}{d_i} - \sum_{\substack{s_{ij} \in S_i, \\ (v_{ijk}, g_{ijk}) \in H_{ij}, \\ min(\mathcal{P}_{res}(v_{ijk}, g_{ijk}))}} \mathcal{P}_{res}(v_{ijk}, g_{ijk}) * q_{ij} \tag{7}$$

Sorting Heuristic H_2 - Average Profit Considering the Total Cost: In this heuristic, the bids are again sorted in descending order of average profit, however, different from the heuristic H_1, in this heuristic the energy costs of the bids are considered in calculating the profit. The heuristic value H_2 of a bid b_i is calculated as follows:

$$\begin{aligned} H_2(b_i) = \frac{p_i}{d_i} - & \sum_{\substack{s_{ij} \in S_i, \\ (v_{ijk}, g_{ijk}) \in H_{ij}, \\ min(\mathcal{P}_{res}(v_{ijk}, g_{ijk}))}} \mathcal{P}_{res}(v_{ijk}, g_{ijk}) * q_{ij} \\ & - \sum_{\substack{s_{ij} \in S_i, \\ (v_{ijk}, g_{ijk}) \in H_{ij}, \\ min(\mathcal{P}_{res}(v_{ijk}, g_{ijk})), \\ v_{ijk} = \Theta(m_a)}} \frac{(\mathcal{E}_{full}(m_a, t) - \mathcal{E}_{idle}(m_a, t)) \; g_{ijk}}{u_a} * q_{ij} \end{aligned} \tag{8}$$

5.1.1 Greedy Placement Heuristic

The first heuristic proposed for the ECO-CABS model is the Greedy Placement (GP) method which is presented in Algorithm 1. In this method, the set of bids is sorted using the provided sorting heuristic and the bids are processed one by one. The virtual machines requested in the subbids of a bid are placed on the physical servers in order, starting from the virtual machines having the smallest reservation price. This placement process is done by taking into account the time constraints specified in the bid. Subbids are placed starting from the earliest available time slot. If the physical server instances do not have enough space for the VM sizes and the quantities specified in a subbid, the placements made so far are rolled back and the entire procedure is repeated for the next time slot. If the requested VMs in any of the subbids cannot be placed on physical servers for a feasible time slot, the bid is marked as rejected and the placement process is continued with the next bid. When all VMs requested in the subbids of the bid are placed on physical server instances, the bid is marked as accepted and the same steps are repeated for the subsequent bid.

Algorithm 1 Greedy Placement Heuristic

```
Input: Problem instance: TestInstance, Sorting Heuristic: SortMethod
Output: A feasible solution to TestInstance: solution
1: procedure GreedyPlacement
2:     Set $sortedBids \leftarrow Sort(TestInstance, SortMethod)$
3:     Set $currentObjectiveValue \leftarrow 0$
4:     Set $solution \leftarrow \emptyset$
5:     for each Bid $b_i \in sortedBids$ do
6:         Set $bidAccepted \leftarrow$ false
7:         for each Timeslot $t_t \in T$ do
8:             Set $subbidSolutionList \leftarrow \emptyset$
9:             for each Subbid $s_{ij} \in S_i$ do
10:                Set $subbidSol \leftarrow AllocateSubbid(s_{ij}, t_t, d_i)$
11:                if $subbidSol \neq \emptyset$ then
12:                    Add $subbidSolutionList \overset{+}{\leftarrow} subbidSol$
13:            if $subbidSolutionList.size() == s_{ij}.size()$ then ▷ All subbids are satisfied
14:                $objValSol \leftarrow CalculateObjectiveValue(solution, subbidSolutionList)$
15:                if $objValSol > currentObjectiveValue$ then
16:                    $currentObjectiveValue \leftarrow objValSol$
17:                    Add $solution \overset{+}{\leftarrow} subbidSolutionList$
18:                    Set $bidAccepted \leftarrow$ true
19:            if $bidAccepted$ then break
20:    return $solution$
21: procedure AllocateSubbid
22:    Parameters: $s_{ij}, t_t, d_i$
23:    Define $(H_{ij}, q_{ij}) = s_{ij}$
24:    Set $remainingQty \leftarrow q_{ij}$
25:    Set $vmSolutionList \leftarrow \emptyset$
26:    for each VM Alternative and Size Tuple $(v_{ijk}, g_{ijk}) \in H_{ij}$ do
27:        for each Physical Machine $m_a \in \Omega(v_{ijk})$ do
28:            while $remainingQty > 0$ do
29:                if $isAllocationFeasible(v_{ijk}, g_{ijk}, t_t, d_i)$ then
30:                    Set $remainingQty \leftarrow remainingQty - 1$
31:                    Set $vmSolutionList \leftarrow Allocate(v_{ijk}, t_t, a)$
32:                else
33:                    break
34:    if $remainingQty \neq 0$ then
35:        Set $vmSolutionList \leftarrow \emptyset$
36:    return $vmSolutionList$
```

Note that even when all the VMs in the subbids are placed, it may be the case that the cost of executing VMs in physical servers can be higher than the offered price of the bid itself. To prevent such cases, an objective value check is performed at each iteration. If the new objective value is less than the current solution, the placement of the VMs requested in the bid is rolled back and the bid is marked as rejected.

5.1.2 Single Bid Placement Heuristic

In the Single Bid Placement (SBP) method presented in Algorithm 2, the set of bids is again sorted using the provided sorting heuristic and the bids are processed one by one. However, in this algorithm, the ODP of the ECO-CABS model is solved for every single bid by using a MIP solver to find the optimal placement for the VMs requested in the bid. For this purpose, initially, the upper bounds of all decision variables are set to 0. Then, for each bid b_i being processed, the corresponding decision variable is reverted to its original state, i.e. its upper bound is set back to the original values specified in Eq. (5) and Eq. (6). This modified model is then solved by the MIP solver. This gives the best placement for the processed bid. If the bid is accepted, then the lower and upper bounds of the corresponding decision variables are fixed to their values found by the MIP solver. In the case of rejection, the lower and upper bounds of the corresponding decision variables are set to 0. The same procedure is repeated until all the bids are processed. This method aims to reduce the execution time compared to the optimal MIP solver by processing bids one by one while giving priority to bids with high sorting heuristic values.

5.1.3 Multiple Bid Placement Heuristic

The SBP described above places bids one by one on the physical servers optimally. However, approaching the optimum solution is quite difficult as it only considers a single bid when scheduling VMs. Another disadvantage is that for problem instances with a large number of bids, the total running time can be high since for each bid an optimization step is carried out. In Multiple Bid Placement (MBP) method, it is aimed to overcome these disadvantages by processing the bids in batches. In this method, the bid set is divided into fixed-size batches, and at each iteration bids inside a batch are processed together. This time, the same decision variable bound modifications defined in the SBP are applied

Algorithm 2 Single Bid Placement Heuristic

```
    Input: Problem instance: TestInstance, Sorting Heuristic: SortMethod
    Output: A feasible solution to TestInstance: solution
 1: procedure SINGLEBIDPLACEMENT
 2:     Set $sortedBids \leftarrow Sort(TestInstance, SortMethod)$
 3:     Set $currentObjectiveValue \leftarrow 0$
 4:     Prepare model defined in Eq.(1)-(6)
 5:     Set upper bounds of decision variables $x_i \leq 0$ ($b_i \in sortedBids$)
 6:     Set upper bounds of the relevant decision variables as $y_i^t \leq 0$, $z_{ijk}^{at} \leq 0$ ($b_i \in sortedBids$, $\forall a, j, k, t$ defined in $b_i$)
 7:     for each Bid $b_i \in sortedBids$ do
 8:         Set the upper bound of decision variable $x_i$ to 1: $0 \leq x_i \leq 1$
 9:         Set bounds of the relevant decision variables as $y_i^t \in \{0, 1\}$, $z_{ijk}^{at} \in \mathbb{Z}^+ \cup \{0\}$ ($\forall a, j, k, t$ defined in $b_i$)
10:         Optimize model using an optimization solver and set the curentSol as the set of values of the decision variables
11:         if $x_i == 1$ in currentSol then
12:             Set the lower bound of decision variable $x_i$ to 1: $1 \leq x_i$
13:             Set lower bounds of the relevant decision variables $y_i^t, z_{ijk}^{at}$ to their values in currentSol ($\forall a, j, k, t$ defined in $b_i$)
14:         else
15:             Set the upper bound of decision variable $x_i$ to 0: $x_i \leq 0$
16:             Set upper bounds of the relevant decision variables as $y_i^t \leq 0, z_{ijk}^{at} \leq 0$
17:     Set $solution \leftarrow model.getDecisionVariables()$
18:     return solution
```

to the corresponding decision variables in the processed batch. Then, the upper and lower bounds of the relevant decision variables are fixed accordingly, and the same process is carried out for the subsequent batches. The pseudocode of the MBP method is presented in Algorithm 3.

Algorithm 3 Multiple Bid Placement Heuristic

```
Input: Problem instance: TestInstance, Sorting Heuristic: SortMethod, Batch Size: bSize
Output: A feasible solution to TestInstance: solution
1: procedure MultipleBidPlacement
2:     Set sortedBids ← $Sort(TestInstance, SortMethod)$
3:     Prepare model defined in Eq.(1)-(6)
4:     Set upper bounds of decision variables $x_i \leq 0$ ($b_i \in$ sortedBids)
5:     Set upper bounds of the relevant decision variables as $y_i^t \leq 0$, $z_{ijk}^{at} \leq 0$ ($b_i \in$ sortedBids, $\forall a, j, k, t$ defined in $b_i$)
6:     for each Batch of bids batch ← sortedBids.nextBatch(bSize) do
7:         for each Bid $b_i \in$ batch do
8:             Set the upper bound of decision variable $x_i$ to 1: $0 \leq x_i \leq 1$
9:             Set bounds of the relevant decision variables as $y_i^t \in \{0,1\}$, $z_{ijk}^{at} \in \mathbb{Z}^+ \cup \{0\}$ ($\forall a, j, k, t$ defined in $b_i$)
10:        Optimize model using an optimization solver and set the curentSol as the set of values of the decision variables
11:        for each Bid $b_i \in$ batch do
12:            if $x_i == 1$ in currentSol then
13:                Set the lower bound of decision variable $x_i$ to 1: $1 \leq x_i$
14:                Set lower bounds of the relevant decision variables $y_i^t, z_{ijk}^{at}$ to their values in currentSol ($\forall a, j, k, t$ defined in $b_i$)
15:            else
16:                Set the upper bound of decision variable $x_i$ to 0: $x_i \leq 0$
17:                Set upper bounds of the relevant decision variables as $y_i^t \leq 0, z_{ijk}^{at} \leq 0$
18:    Set solution ← $model.getDecisionVariables()$
19:    return solution
```

5.1.4 Genetic Algorithm

The final heuristic method proposed for the ECO-CABS model is the Genetic Algorithm (GA) the pseudocode of which is presented in Algorithm 4. In the GA implementation, each individual corresponds to an order of bids, i.e. a permutation of bids, and the population consists of different bid orders. Thus, instead of the solution space, the genetic algorithm searches the space of all possible orders of bids. Since an order of bids does not define a solution, the order is converted to a solution using the Greedy Placement method presented in Algorithm 1. The objective value returned by this method for an individual, i.e. the order of bids, is used as the fitness value of the corresponding individual.

The initial population in the genetic algorithm is constructed using three predefined bid orders and $N-3$ random bid orders where N is the population size. The two of the predefined orders are the bid orders defined by the sorting heuristics H_1 and H_2. For the third predefined order, the linear relaxation of the ODP presented in Eq.(2)-(6) is solved. Then, the bids are sorted in descending order of the values of the *relaxed* decision variables x.

The selection of candidate parents is conducted using the binary tournament selection method [63]. The crossover and mutation methods used in this approach are described in detail below.

Crossover: Since this GA implementation searches the space of possible bid orders, a modified version of the uniform crossover method is used in this implementation. So, in the crossover method, the bid orders of both parents are preserved as much as possible when forming a new individual. In this method, bid orders of the two parents are traversed and at each step, a bid from one of the parents (uniformly chosen with equal probability) is added to the bid order of the newly created individual. Then, the other parent's bid is added to this order. If the selected bid has already been added to the order of the new individual before, it will not be added again.

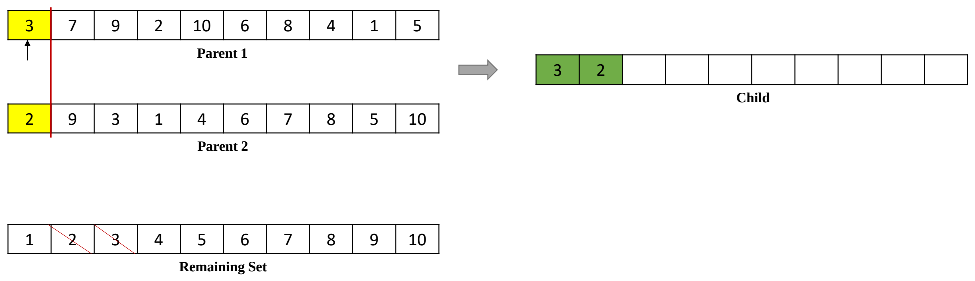
Figure 5a shows the bid orders of the two example parents. First, a uniform selection is made between the parents with equal probability. Suppose that Parent 1 is chosen. Then the first bid of Parent 1, i.e. Bid 3, is added to the order of the child. After that, the corresponding bid of the other parent, Bid 2 is added to the order of the child immediately after Bid 3.

In the next step, a choice is made between the second bids of both parents. Suppose that this time Parent 2 is selected. Thus, the second bid of Parent 2 is added to the order of the child first. Then, the second bid of the other parent is added immediately after this bid as can be seen in Figure 5b.

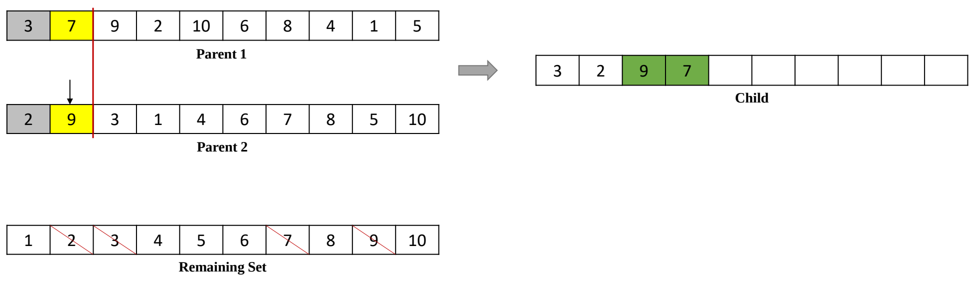
In the third step seen in Figure 5c, independent of the selected parent, no action is taken in this step since both parents' bids have already been added to the order of the child in the previous steps.

Algorithm 4 Genetic Algorithm

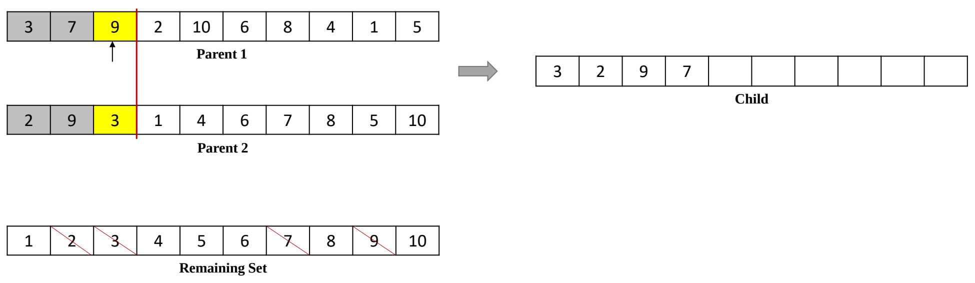
```
    Input: Problem instance: TestInstance, Population Size: $N$
    Output: A feasible solution to TestInstance: solution
 1: procedure GeneticAlgorithm - Main Method
 2:     Set $population \leftarrow InitializePopulation()$
 3:     Set $bestIndividualSoFar \leftarrow GetBestIndividual(population)$
 4:     repeat
 5:         Call $GenerationCycle()$
 6:         Set $bestIndividualInGeneration \leftarrow GetBestIndividual(population)$
 7:         if $bestIndividualInGeneration.fitness > bestIndividualSoFar.fitness$ then
 8:             Set $bestIndividualSoFar \leftarrow bestIndividualInGeneration$
 9:     until One of the stopping criteria is true
10:     Set $mbpSolution \leftarrow MultipleBidPlacement(bestIndividualSoFar)$
11:     if $bestIndividualInGeneration.fitness > mbpSolution.fitness$ then
12:         return $bestIndividualInGeneration$
13:     else
14:         return $mbpSolution$
15: procedure GenerationCycle
16:     Parameters: $population$
17:     Set $parent_1 \leftarrow BinaryTournamentSelection(population)$
18:     Set $parent_2 \leftarrow BinaryTournamentSelection(population)$
19:     Call $Crossover(population, parent_1, parent_2)$
20:     Call $Mutation(population)$
21: procedure Crossover
22:     Parameters: $population, parent_1, parent_2$
23:     Initialize lookup table: $remainingSet \leftarrow [1, numberOfBids]$
24:     Set $childBidOrder \leftarrow \emptyset$
25:     for each Tuple $(b_{i,p_1}, b_{i,p_2}) \in parent_1.bidOrder(), parent_2.bidOrder()$ do
26:         Set $firstPivotBid, secondPivotBid \leftarrow 0$
27:         if $RandomNumber(0.0, 1.0) \geq 0.5$ then
28:             $firstPivotBid \leftarrow b_{i,p_1}$
29:             $secondPivotBid \leftarrow b_{i,p_2}$
30:         else
31:             $firstPivotBid \leftarrow b_{i,p_2}$
32:             $secondPivotBid \leftarrow b_{i,p_1}$
33:         Set $childBidOrder \overset{+}{\leftarrow} firstPivotBid, secondPivotBid$
34:         Set $remainingSet \overset{-}{\leftarrow} firstPivotBid, secondPivotBid$
35:     Add $childBidOrder \overset{+}{\leftarrow} remainingSet$
36:     Set $child \leftarrow FitnessEvaluation(childBidOrder)$
37:     Call $ReplaceWorstSolution(child)$
```



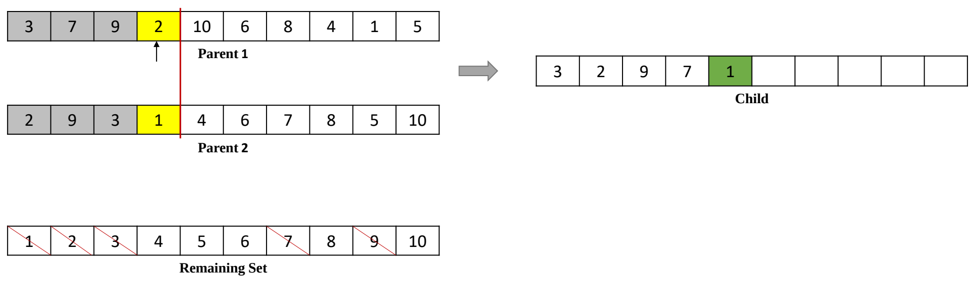
(a) Crossover Step 1: Parent 1 is selected. The first bid of Parent 1, and then the first bid of Parent 2 are added to the child.



(b) Crossover Step 2: Parent 2 is selected. The second bid of Parent 2, and then the second bid of Parent 1 are added to the child.



(c) Crossover Step 3: Parent 1 is selected. No action is taken since the bids of both parents have already been added to the child.



(d) Crossover Step 4: Parent 1 is selected. Since the fourth bid of Parent 1 has already been added, only the fourth bid of the Parent 2 is added to the child.

Figure 5: The steps of the crossover method of the genetic algorithm on an example scenario.

Finally, in the fourth step, suppose that Parent 1 is selected again. However, since the fourth bid in the order of Parent 1, Bid 2, has already been added to the child, this bid is skipped. Only the fourth bid in the order of Parent 2, Bid 1, is added to the order of the child. Details of this step are presented in Figure 5d. When these steps are completed, a complete bid ordering is obtained for the new individual.

Thus, after the crossover operation, a new individual is obtained, and the individual with the lowest fitness value is replaced by this new individual.

Mutation: In the mutation method, the order of a randomly selected individual is altered. First, the order of the bids for the individual to be mutated is divided into fixed-sized groups, and then two randomly selected bids from each group are swapped. Since the swap operation is between the bids within each group, the original bid order does not change dramatically. In this study, a mutation probability of 0.05 is used and each bid order to be mutated is divided into groups of five [1]. Three exemplary mutations performed on an example individual are shown in Figure 6.

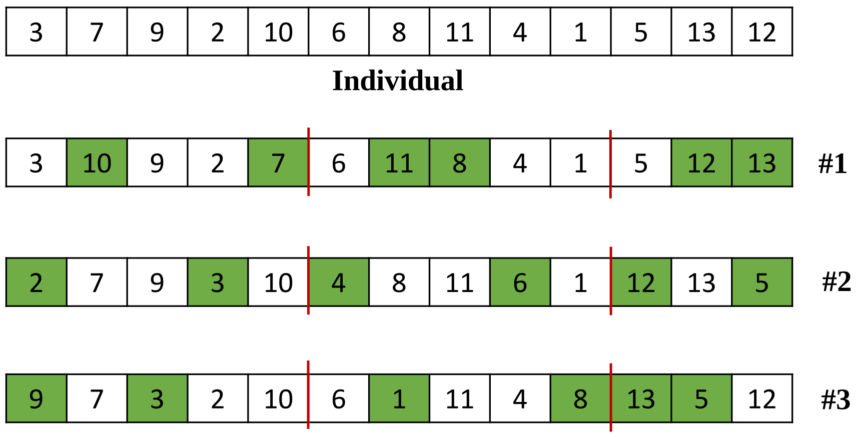


Figure 6: Three mutation possibilities for an example individual with a group size of 5.

6 Experimental Results

By virtue of its novel bidding language, the ECO-CABS model enables users to indicate their complex virtual resource requests using logical AND or OR relations along with the scheduling constraints. Although major cloud providers provide grouping functionality for

[1] These values are determined based on a preliminary test.

managing multiple *identical* VMs such as Amazon EC2 Placement Groups [64], Google Cloud Instance Groups [65], and Microsoft Azure Virtual Machine Scale Sets [66], to the best of our knowledge, no cloud provider offers a mechanism to obtain possibly complex preferences of the users as the ECO-CABS model. Without such a mechanism, it is not possible to capture the details required for assessing the real-life performance of the ECO-CABS model.

For this reason, to estimate the performance of the ECO-CABS model and the proposed heuristic methods, a test case generator has been developed. The generator has several parameters for generating test cases representing various scenarios, which are:

i. **Available VM Count** defines the sum of the VMs with a base configuration that can be run concurrently on all physical server instances, that is the sum of the u_a values of the physical server instances. The values $\{1024, 2048, 3072, 4096\}$ are used for this parameter.

ii. **Bid Density** defines the ratio of requested VM count to the available VM count considering the duration requested in the bids and the total scheduling period. The requested VM count is calculated as the sum of the requested sizes of the VMs in all subbids. This parameter takes the values of $\{0.25, 0.50, 0.75, 1, 2, 3, 4, 5\}$. Note that the bid density parameter represents the inverse of the supply-demand ratio. When the bid density is less than 1, the supply exceeds the demand, whereas, when it is greater than 1, the demand exceeds the supply.

iii. **Mean Number of Subbids** defines the mean number of subbids in a bid. Poisson distribution with mean values of $\{1, 2, 3\}$ is used for this parameter.

iv. **Mean Number of Alternative VMs in a Subbid** defines the mean number of alternative VM requests in a subbid. Poisson distribution with mean values of $\{1, 2, 3\}$ is used for this parameter.

v. **Mean Number of Quantity for a Subbid** defines the mean number of quantity values for VM requests in a subbid. Poisson distribution with mean values of $\{1, 2, 3\}$ is used for this parameter.

vi. **Scheduling Period** defines the total number of available time slots for the requested VMs to be scheduled. This parameter takes the values of $\{5, 10, 15, 20\}$ slots.

The first parameter determines the number of physical server instances in the cloud environment, while together with the first parameter, the bid density parameter determines the number of bids created in the test case. All defined parameters affect the complexity of the problems generated.

Using the stated value set for each parameter, 3456 test instances were created using the test case generator. To evaluate the model and estimate the performance of the proposed heuristic methods, each test instance was solved by a MIP solver, Gurobi Optimizer version 9.1 [67] with a time limit of one hour per test instance. Experiments were conducted on a workstation with a 16-core 3.10 GHz Intel Xeon CPU and 128 GB of RAM.

The reservation prices used in the study are determined based on AWS EC2 on-demand virtual machine instance prices [68]. One time slot is considered as 24 hours and the cost of energy per kWh is set to €0.21 which is the average electricity cost in the EU [69].

The source code of the test case generator proposed in this study can be reached from [62].

6.1 Evaluating the Performance of the Heuristic Methods

The ECO-CABS model is a novel scheduling model that combines the multi-unit nondiscriminatory combinatorial auction institution with an energy-aware virtual to physical resource mapping. The optimization problem (ODP) of the ECO-CABS model is significantly different than that of the proposed approaches in the literature as surveyed briefly in Section 2. Therefore, the generated test cases cannot be solved using the solution methods proposed in those studies.

For this reason, to evaluate the performance of the proposed heuristic methods, the results found by the proposed heuristic methods are compared to the results found by the MIP solver. For this purpose, each of the 3456 test instances is solved using all the proposed heuristic methods and also using the MIP solver for obtaining the optimal solutions for the ODP of the ECO-CABS model. However, although relatively small-sized test cases are generated, the MIP solver was able to find optimal solutions for only 547 test instances. For the 2849 instances, it was able to find a feasible solution in one hour which is not necessarily optimal. For the remaining 60 test instances, the MIP solver could not find a feasible solution at all within this one-hour time limit. The instances for which the MIP solver could not find a solution are not included in this experiment. Although the MIP solver could not find the optimal solutions for the majority of the test cases, it is observed that the optimality gaps for these instances are very small on average. The histogram presenting the optimality gap values can be seen in Figure 7. It is observed that the majority of solutions obtained with the MIP solver have optimality gap values less than 1%, that is the solutions are within at most 1% of the optimal results.

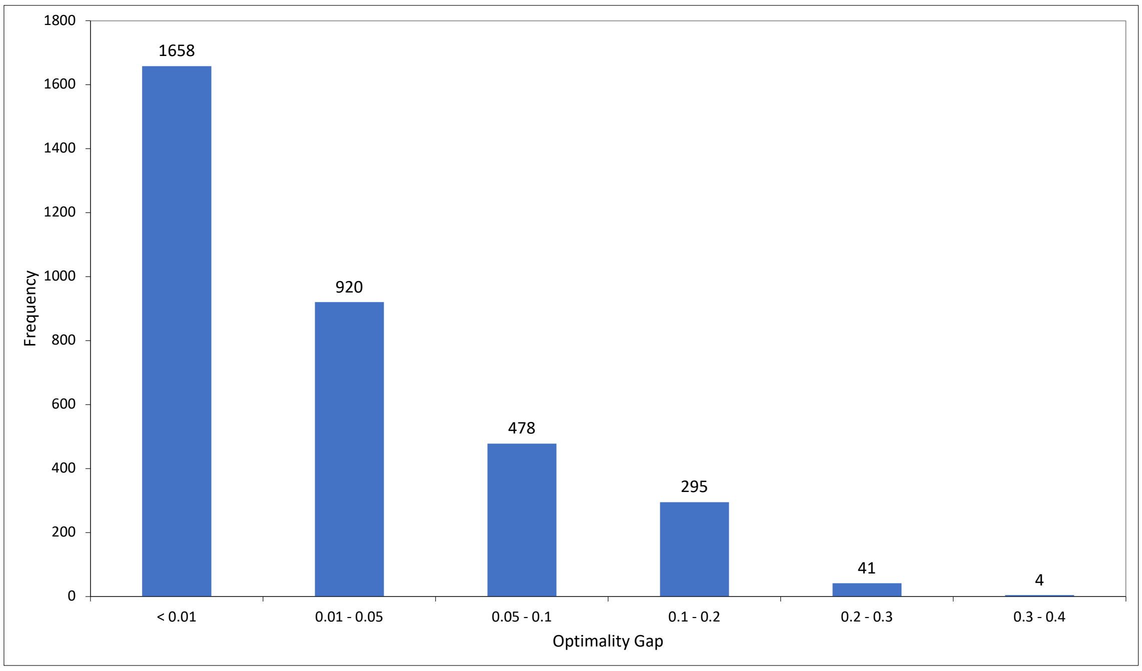


Figure 7: Histogram of the optimality gap values of the solutions found by the MIP solver for all test instances.

To assess the performance of a heuristic method, a metric named *success rate* is used which indicates the ratio of the solution found by a heuristic method to the solution found by the MIP solver. Thus, a success rate of 100% indicates that the corresponding heuristic has found the same solution as the MIP solver in terms of objective values. Each test instance is solved with 12 different heuristic and sorting method combinations. The box plot for the success rates for all proposed heuristic methods can be seen in Figure 8.

To understand if the differences in the mean success rates of the proposed heuristic methods presented in Figure 8 are significant, the one-way Welch analysis of variance

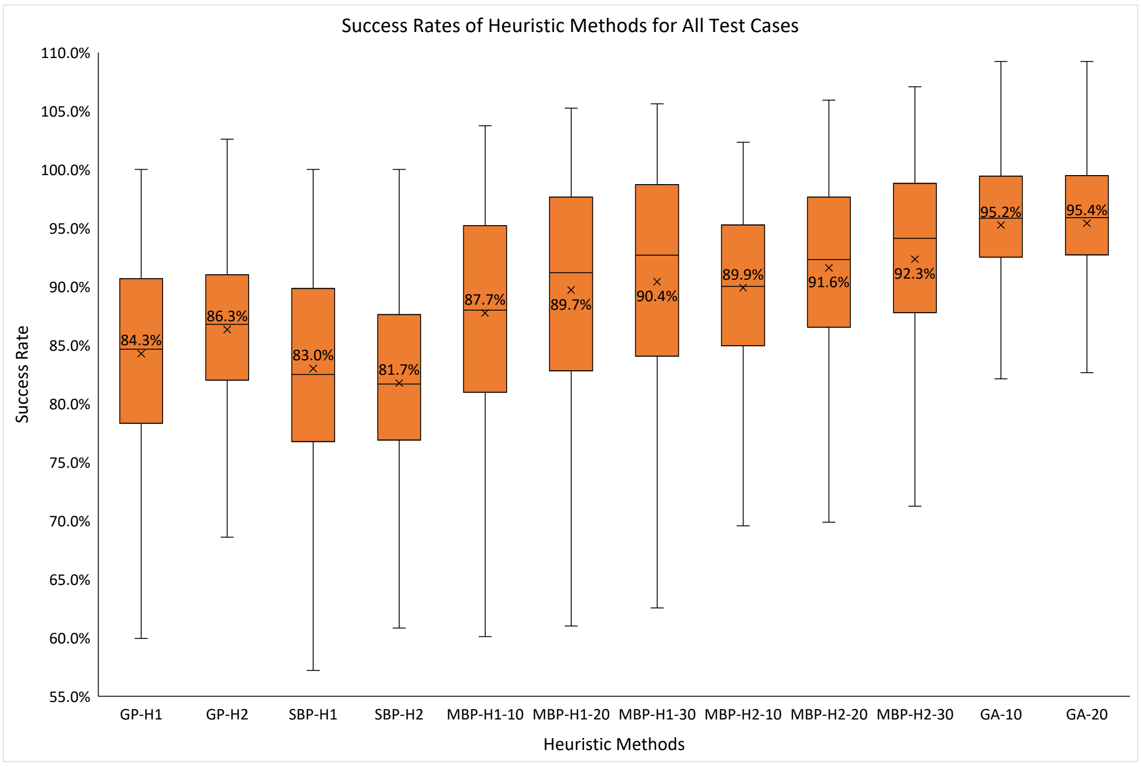


Figure 8: Box plot representing the success rates of the proposed heuristic methods for all test cases.

(Welch ANOVA) [70] test is conducted at an $\alpha = 0.05$ significance level. The success rates are found to be statistically different for different heuristic configurations (Welch's $F(11, 15999.868) = 1430.234, p < 0.0005$). Since the assumption of homogeneity of variances is violated, as assessed by Levene's test for equality of variances ($p < 0.0005$), to interpret the results of the Welch ANOVA and the results from the Games-Howell post hoc test are used for multiple comparisons. The results of multiple comparisons are presented in Table 3. In the following text, the term *significant* is used for a difference of mean values that is statistically significant at the $\alpha = 0.05$ significance level.

The Greedy Placement Heuristic (GP) is observed to have approximately 84.3% and 86.3% success rates on average using the proposed sorting heuristics H_1 and H_2, respectively. It is observed that using H_2 instead of H_1 contributes to the quality of solutions significantly for this heuristic.

Although more complicated, the Single Bid Placement Heuristic (SBP), provides a significantly lower quality of solutions on average than the GP method with mean success rates of approximately 83.0% and 81.7% using H_1 and H_2, respectively. However, unlike

Table 3: Pairwise comparsions of the mean success rates of the heuristic methods provided by one-way Welch ANOVA with Games-Howell post hoc test. Each cell at the coordinate (X, Y) represents the difference between the mean success rate of the heuristic methods at the row X and at the column Y. Values written in bold indicates that the corresponding mean difference is significant at the $\alpha = 0.05$ significance level.

	GP-H1	GP-H2	SBP-H1	SBP-H2	MBP-H1-10	MBP-H1-20	MBP-H1-30	MBP-H2-10	MBP-H2-20	MBP-H2-30	GA-10
GA-20	**11.1%**	**9.1%**	**12.4%**	**13.6%**	**7.7%**	**5.7%**	**5.0%**	**5.5%**	**3.8%**	**3.1%**	0.2%
GA-10	**11.0%**	**8.9%**	**12.3%**	**13.5%**	**7.5%**	**5.5%**	**4.8%**	**5.4%**	**3.6%**	**2.9%**	
MBP-H2-30	**8.1%**	**6.0%**	**9.3%**	**10.6%**	**4.6%**	**2.6%**	**1.9%**	**2.4%**	**0.7%**		
MBP-H2-20	**7.3%**	**5.3%**	**8.6%**	**9.9%**	**3.9%**	**1.9%**	**1.2%**	**1.7%**			
MBP-H2-10	**5.6%**	**3.6%**	**6.9%**	**8.1%**	**2.2%**	0.2%	-0.5%				
MBP-H1-30	**6.1%**	**4.1%**	**7.4%**	**8.7%**	**2.7%**	**0.7%**					
MBP-H1-20	**5.4%**	**3.4%**	**6.7%**	**7.9%**	**2.0%**						
MBP-H1-10	**3.5%**	**1.4%**	**4.7%**	**6.0%**							
SBP-H2	**-2.5%**	**-4.6%**	**-1.2%**								
SBP-H1	**-1.3%**	**-3.3%**									
GP-H2	**2.1%**										

the GP method, the SBP method using sorting heuristic H_1 provides significantly better solutions than the SBP method using the sorting heuristic H_2.

The GP and the SBP methods do not have any input parameters. However, as explained in the previous section the Multiple Bid Placement Heuristic (MBP) has a batch-size parameter that defines the number of bids to be processed together in a single run. To estimate the effect of this parameter, three batch-size values, 10, 20, and 30, are used in this experiment. Thus, along with the two sorting heuristics, 3x2=6 configurations were executed for each test case. For each batch, a time limit of 120 seconds is enforced.

Compared to the previous methods, a significant improvement in the quality of the solutions is observed using the MBP method. The best results are obtained when the batch size of 30 and the sorting heuristic H_2 are used where the mean success rate is 92.3% . It is also seen that there exists a positive correlation between the batch size and the mean success rate, and the sorting heuristic H_2 provides significantly better solutions than the sorting heuristic H_1. Note that since a time limit of 120 seconds is enforced for each batch, the maximum batch size is set to 30 in this experiment. The batch size parameter may further be increased in accordance with this time limit which will, of course, increase the running time of the method.

The most promising method proposed in this study is the Genetic Algorithm (GA). Similar to the MBP method, the GA has a population size parameter and the population sizes of 10 and 20 are used in this experiment. The GA has provided the best results with mean success rates of 95.2% and 95.4% for the population sizes, 10 and 20, respectively. However, the increase in the mean success rate when a population size of 20 is used has not been found statistically significant, therefore higher population sizes were not considered in this experiment.

As noted at the beginning of this section, the available VM count parameter directly affects the size of the problem instances. Therefore, the performances of the heuristic methods are also analyzed with respect to this parameter. The box plot in Figure 9 presents the mean success rates of the proposed heuristics grouped by the available VM count parameter. As the number of available VMs increases, it is observed that the quality of solutions with respect to the MIP solver tends to increase for the GP methods, whereas it tends to decrease for the SBP methods. The reason for the decrease is that for the test instances containing a high number of bids, the SBP method cannot find optimal placement for even a single bid because of the time limit set per bid. The changes in the quality of solutions found by the remaining methods are found to be insignificant.

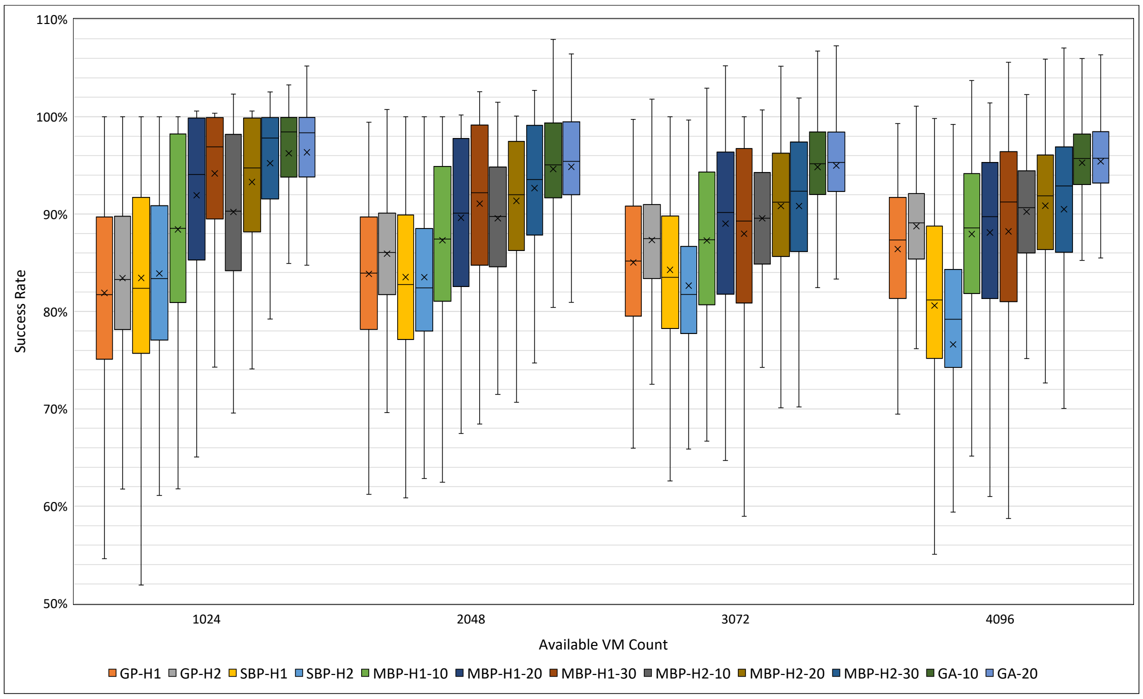


Figure 9: Box plot representing success rates of the proposed heuristic methods grouped by the available VM count paremeter .

Another important metric for evaluating the performances of heuristic methods is the running time of the methods. Table 4 presents the mean and the standard deviations of the running times of the MIP solver and the proposed heuristic methods based on the available VM count parameter. The mean running time of the MIP solver for all test instances increases from 2485 seconds to 3410 seconds as the number of available (also requested) VMs increases with an overall mean value of 3141 seconds and standard deviation of 998 seconds. Recall that among the 3396 test instances (discarding the unsolved instances), only 547 of them are solved within 1 hour time limit, and the mean running time would be much higher if a time limit of 3600 seconds was not enforced.

Table 4: Mean running times (in seconds) of the proposed heuristic methods grouped by the available VM count paremeter.

Heuristic	**Available VM Count**									
	1024		2048		3072		4096		**Overall**	
	mean	std	mean	std	mean	std	mean	std	**mean**	**std**
GP-H1	< 1	< 1	2	3	6	11	11	16	**5**	**8**
GP-H2	< 1	< 1	2	4	6	10	11	16	**5**	**8**
SBP-H1	21	32	239	470	1116	2190	1870	3375	**811**	**1517**
SBP-H2	76	96	652	827	1315	1774	2568	3587	**1153**	**1571**
MBP-H1-10	37	49	75	79	225	400	423	674	**190**	**300**
MBP-H1-20	71	54	101	72	161	164	501	707	**209**	**249**
MBP-H1-30	82	52	104	52	210	208	326	346	**181**	**165**
MBP-H2-10	40	49	79	95	194	302	456	766	**192**	**303**
MBP-H2-20	73	54	107	85	202	206	384	480	**192**	**206**
MBP-H2-30	82	52	103	59	172	148	303	378	**165**	**159**
GA-10	73	64	199	258	427	490	637	634	**334**	**362**
GA-20	82	79	274	388	608	700	871	865	**459**	**508**
MIP Solver	2485	1568	3290	923	3378	768	3410	732	**3141**	**998**

Being the least complicated method, the GP method provides the fastest solutions for all test instances with a mean running time of approximately 5 seconds. Therefore, it can be used when the problem sizes are huge or when a solution is required in a very short interval.

The longest-running method is the SBP method with a mean running time of 811 seconds when H_1 is used and 1153 seconds when H_2 is used. The reason for longer running times is that an optimization step is carried out for each bid in the problem instance. Thus, it is not feasible to use this method for large problem instances.

The mean running times of the MBP method range between 165 and 209 seconds depending on the sorting heuristic used and the batch size parameter. The running time of the method is observed to be lower when the sorting heuristic H_2 is used. The effect of the batch size on the running time cannot be deduced simply, however, increasing the batch size to 30 causes a decrease in the mean running times for both sorting heuristic methods.

The GA executes slower than both the GP and MBH methods having a mean running time of 334 seconds when a population size of 10 is used and 459 seconds when a population size of 20 is used. The extra running time is used to search the solution space efficiently. Recall that since there is no significant improvement observed in the quality of solutions when the population size is increased to 20, the population size of 10 would be enough for this method and the mean running time is 334 seconds in this case. Thus, it can be concluded that the GA executes significantly faster than the MIP solver on average.

6.2 Estimating the Improvement in Outcomes Using the ECO-CABS Model

The features of the ECO-CABS model and its benefits have been explained in detail in the previous sections. To estimate the improvement to be achieved using the ECO-CABS model on a cloud system, we simulated each test case using the First-Come-First-Served approach of the current cloud systems in order to compare the outcomes. More specifically, for each test case, the following steps were carried out:

i. The bids in the given test case are shuffled uniformly to represent a random arrival order.

ii. The bids are processed one by one in the arrival order to simulate the First-Come-First-Served approach of the current cloud systems.

iii. Each bid is checked whether it can be scheduled or not based on the current utilization of the physical servers. If a bid can be scheduled, then the VMs are mapped to the physical server such as the energy consumption is minimized, and the bid is included in the solution of the simulation.

iv. The objective value of the simulation solution is calculated after all the bids are processed.

Each of the 3396 test instances was simulated 20 times with different random arrival orders and the mean objective value of 20 runs was calculated for each test instance. These mean objective values are compared to the objective values obtained by solving the ODP of the ECO-CABS model for the same instances.

The mean improvement percentages grouped according to the bid density parameter are presented in Figure 10. It is observed that the ECO-CABS model provides approximately 9% to 66% better results compared to the simulation results for different bid density values. The overall improvement for all test cases is observed as approximately 37%.

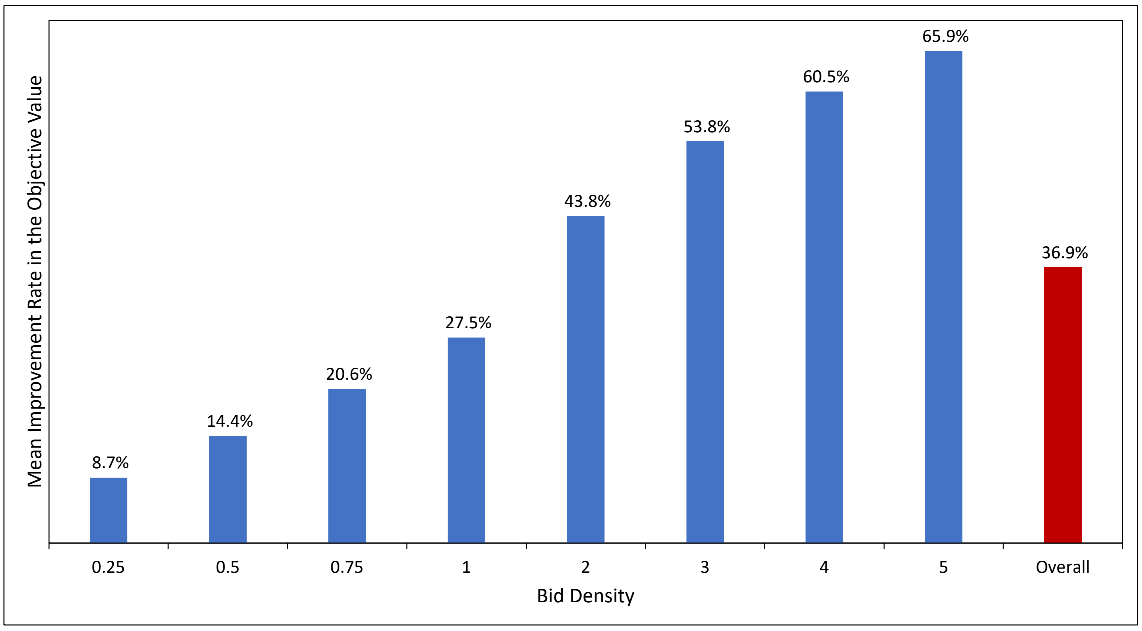


Figure 10: Mean improvement percentage in the objective values using the ECO-CABS model over the simulation results.

Note that when the bid density values less than one, i.e. the number of available VM slots is greater than the number of requested VMs, most of the bids can be scheduled. Therefore, the improvement to be observed when the ECO-CABBS model is used is relatively small, obtained mostly by the energy-efficient placement of VMs to the physical servers. However, when the bid density is more than one, the model's feature of selecting the optimal set of bids becomes effective, further increasing the improvement rate.

7 CONCLUSION

In this study, an energy-aware combinatorial auction-based virtual machine scheduling model for cloud environments is presented. The model utilizes the multi-unit nondiscriminatory combinatorial auction institution for determining the efficient scheduling of VMs. The bidding language of the ECO-CABS model allows cloud users to declare their complex VM scheduling preferences in their bids along with the duration and the time frame for the allocation. The model also incorporates an energy model based on the physical server utilization levels. Together with the auction features and the energy model, the ECO-CABS model finds the optimal schedule of VMs and the energy-efficient placement of VMs to the physical servers based on the bids of the users.

The mathematical formulation of the ECO-CABS model is provided and the corresponding optimization problem of the model is formulated using integer linear programming. Since the optimization problem of the model is NP-Hard, several VM scheduling and placement heuristic methods along with two different sorting heuristic methods have been proposed. To demonstrate the performance of the model and the heuristic methods, a comprehensive test suite of 3456 test instances with different complexities has been prepared using a test case generator. Two experiments have been conducted. In the first experiment, the performances of the proposed heuristic methods are measured. The test suite is solved using 12 different heuristic configurations and a general-purpose MIP solver. It is observed that the simplest method proposed, the greedy placement method, provides solutions within 15% of the optimum in a few seconds. The most complicated method proposed, the genetic algorithm-based heuristic method, has outperformed all other heuristic methods providing solutions within only 5% of the optimum while executing significantly faster than the MIP solver. In the second experiment, the improvement to be obtained when the proposed ECO-CABS model is used in data centers is studied. It is estimated that approximately a 37% improvement in the profit of the cloud providers can be obtained when the ECO-CABS model is used on average. This improvement rate further increases as the number of bids submitted by the user increases.

The ECO-CABS model provides efficient VM scheduling even in real-world cloud environments within reasonable times using the proposed heuristic methods. The model is expected to contribute to a sustainable cloud computing platform with reduced energy usage while offering higher profits to cloud providers.

Acknowledgments

This work was supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under Grant 119E007. The authors would like to thank the reviewers for their helpful comments that have helped to improve the paper.

References

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy H Katz, Andrew Konwinski, Gunho Lee, David A Patterson, Ariel Rabkin, Ion Stoica, et al. Above the clouds: A berkeley view of cloud computing. Technical report, UCB/EECS-2009-28, EECS Department, University of California, 2009.

[2] Peter Mell and Tim Grance. The nist definition of cloud computing. *NIST Special Publications*, 2011.

[3] Arman Shehabi, Sarah Smith, Dale Sartor, Richard Brown, Magnus Herrlin, Jonathan Koomey, Eric Masanet, Nathaniel Horner, Inês Azevedo, and William Lintner. United states data center energy usage report. Technical report, Lawrence Berkeley National Laboratory, 2016. URL `https://eta.lbl.gov/publications/united-states-data-center-energy`.

[4] Arman Shehabi, Sarah J Smith, Eric Masanet, and Jonathan Koomey. Data center growth in the united states: decoupling the demand for services from electricity use. *Environmental Research Letters*, 13(12):124030, 2018.

[5] Eric Masanet, Arman Shehabi, Nuoa Lei, Sarah Smith, and Jonathan Koomey. Recalibrating global data center energy-use estimates. *Science*, 367(6481):984–986, 2020.

[6] Anders Andrae. Total consumer power consumption forecast. *Nordic Digital Business Summit*, 10, 2017.

[7] Fei Teng, Lei Yu, Tianrui Li, Danting Deng, and Frédéric Magoulès. Energy efficiency of vm consolidation in iaas clouds. *The Journal of Supercomputing*, 73(2):782–809, 2017.

[8] Nicola Jones. How to stop data centres from gobbling up the world's electricity. *Nature*, 561(7722):163–167, 2018.

[9] Lotfi Belkhir and Ahmed Elmeligi. Assessing ict global emissions footprint: Trends to 2040 & recommendations. *Journal of cleaner production*, 177:448–463, 2018.

[10] Google. Google environmental report 2020, 2020. URL `https://www.gstatic.com/gumdrop/sustainability/google-2020-environmental-report.pdf`. [Online; accessed 8 Aug 2022].

[11] Amazon. Amazon sustainability 2021 report, 2022. URL `https://sustainability.aboutamazon.com/pdfBuilderDownload?name=amazon-sustainability-2020-report`. [Online; accessed 8 Aug 2022].

[12] The Equipment Energy Efficiency (E3) Program. Energy efficiency policy options for australian and new zealand data centres, 2014. URL `https://www.energyrating.gov.au/sites/default/files/documents/Energy-Efficiency-Policy-Options-for-AUSNZ-Data-Centres_April-2014_0.pdf`. [Online; accessed 8 Aug 2022].

[13] Rabih Bashroush and Andy Lawrence. Beyond pue: tackling it's wasted terawatts. *Uptime Institute*, 2020. URL `https://uptimeinstitute.com/beyond-pue-tackling-it\OT1\textquoterights-wasted-terawatts`.

[14] Najet Hamdi and Walid Chainbi. A survey on energy aware vm consolidation strategies. *Sustainable Computing: Informatics and Systems*, 23:80–87, 2019. ISSN 2210-5379. doi:10.1016/j.suscom.2019.06.003.

[15] Saikishor Jangiti and Shankar Sriram VS. Emc2: Energy-efficient and multi-resource- fairness virtual machine consolidation in cloud data centres. *Sustainable Computing: Informatics and Systems*, 27:100414, 2020. ISSN 2210-5379. doi:10.1016/j.suscom.2020.100414.

[16] Ali Haydar Özer and Can Özturan. A model and heuristic algorithms for multi-unit nondiscriminatory combinatorial auction. *Computers & Operations Research*, 36(1): 196–208, 2009.

[17] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755 – 768, 2012. ISSN 0167-739X. doi:10.1016/j.future.2011.04.017. Special Section: Energy efficiency in large-scale distributed systems.

[18] Soha Rawas, Ahmed Zekri, and Ali El-Zaart. Lecc: Location, energy, carbon and cost-aware vm placement model in geo-distributed dcs. *Sustainable Computing: Informatics and Systems*, 33:100649, 2022. ISSN 2210-5379. doi:10.1016/j.suscom.2021.100649.

[19] Chaima Ghribi, Makhlouf Hadji, and Djamal Zeghlache. Energy efficient vm scheduling for cloud data centers: Exact allocation and migration algorithms. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pages 671–678. IEEE, 2013.

[20] Wei Zhu, Yi Zhuang, and Long Zhang. A three-dimensional virtual resource scheduling method for energy saving in cloud computing. *Future Generation Computer Systems*, 69:66–74, 2017.

[21] Youwei Ding, Xiaolin Qin, Liang Liu, and Taochun Wang. Energy efficient scheduling of virtual machines in cloud with deadline constraint. *Future Generation Computer Systems*, 50:62–74, 2015.

[22] Xiaojun Ruan, Haiquan Chen, Yun Tian, and Shu Yin. Virtual machine allocation and migration based on performance-to-power ratio in energy-efficient clouds. *Future Generation Computer Systems*, 100:380–394, 2019.

[23] The Standard Performance Evaluation Corporation. The Standard Performance Evaluation Corporation, SPECpower_ssj® 2008, 08 2022. URL `https://www.spec.org/power_ssj2008`. [Online; accessed 8 Aug 2022].

[24] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50, 2011.

[25] Zhongjin Li, Jidong Ge, Haiyang Hu, Wei Song, Hao Hu, and Bin Luo. Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds. *IEEE Transactions on Services Computing*, 11(4):713–726, 2015.

[26] Aftab Ahmed Chandio, Nikos Tziritas, Muhammad Saleem Chandio, and Cheng-Zhong Xu. Energy efficient vm scheduling strategies for hpc workloads in cloud data centers. *Sustainable Computing: Informatics and Systems*, 24:100352, 2019. ISSN 2210-5379. doi:10.1016/j.suscom.2019.100352.

[27] Manojit Ghose, Aryabartta Sahu, and Sushanta Karmakar. Urgent point aware energy-efficient scheduling of tasks with hard deadline on virtualized cloud system. *Sustainable Computing: Informatics and Systems*, 28:100416, 2020. ISSN 2210-5379. doi:10.1016/j.suscom.2020.100416.

[28] Xiangming Dai, Jason Min Wang, and Brahim Bensaou. Energy-efficient virtual machines scheduling in multi-tenant data centers. *IEEE Transactions on Cloud Computing*, 4(2):210–221, 2015.

[29] Sambit Kumar Mishra, Deepak Puthal, Bibhudatta Sahoo, Prem Prakash Jayaraman, Song Jun, Albert Y Zomaya, and Rajiv Ranjan. Energy-efficient vm-placement in cloud data center. *Sustainable computing: informatics and systems*, 20:48–55, 2018.

[30] Hancong Duan, Chao Chen, Geyong Min, and Yu Wu. Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems. *Future Generation Computer Systems*, 74:142–150, 2017.

[31] Fei Tao, Ying Feng, Lin Zhang, and T Warren Liao. Clps-ga: A case library and pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling. *Applied Soft Computing*, 19:264–279, 2014.

[32] Damián Fernández-Cerero, Agnieszka Jakóbik, Daniel Grzonka, Joanna Kołodziej, and Alejandro Fernández-Montes. Security supportive energy-aware scheduling and energy policies for cloud environments. *Journal of Parallel and Distributed Computing*, 119:191–202, 2018.

[33] Hongtao Lei, Rui Wang, Tao Zhang, Yajie Liu, and Yabing Zha. A multi-objective co-evolutionary algorithm for energy-efficient scheduling on a green data center. *Computers & Operations Research*, 75:103–117, 2016.

[34] Mohan Sharma and Ritu Garg. An artificial neural network based approach for energy efficient task scheduling in cloud data centers. *Sustainable Computing: Informatics and Systems*, 26:100373, 2020. ISSN 2210-5379. doi:10.1016/j.suscom.2020.100373.

[35] Wanyuan Wang, Yichuan Jiang, and Weiwei Wu. Multiagent-based resource allocation for energy minimization in cloud computing systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(2):205–220, 2016.

[36] AS Ajeena Beegom and MS Rajasree. A particle swarm optimization based pareto optimal task scheduling in cloud computing. In *International Conference in Swarm Intelligence*, pages 79–86. Springer, 2014.

[37] Mohit Kumar and SC Sharma. Pso-cogent: Cost and energy efficient scheduling in cloud environment with deadline constraint. *Sustainable Computing: Informatics and Systems*, 19:147–164, 2018.

[38] Yacine Kessaci, Nouredine Melab, and El-Ghazali Talbi. A multi-start local search heuristic for an energy efficient vms assignment on top of the opennebula cloud manager. *Future Generation Computer Systems*, 36:237–256, 2014.

[39] Shashikant Ilager, Kotagiri Ramamohanarao, and Rajkumar Buyya. Etas: Energy and thermal-aware dynamic virtual machine consolidation in cloud data center with proactive hotspot mitigation. *Concurrency and Computation: Practice and Experience*, 31(17):e5221, 2019.

[40] Dujing Chen and Yanyan Zhang. Diversity-aware marine predators algorithm for task scheduling in cloud computing. *ENTROPY*, 25(2), FEB 2023. doi:10.3390/e25020285.

[41] Roshni Pradhan and Suresh Chandra Satapathy. Energy aware genetic algorithm for independent task scheduling in heterogeneous multi-cloud environment. *JOURNAL OF SCIENTIFIC & INDUSTRIAL RESEARCH*, 81(7):776–784, JUL 2022. ISSN 0022-4456.

[42] Xiaojian He, Junmin Shen, Fagui Liu, Bin Wang, Guoxiang Zhong, and Jun Jiang. A two-stage scheduling method for deadline-constrained task in cloud computing. *CLUSTER COMPUTING-THE JOURNAL OF NETWORKS SOFTWARE TOOLS AND APPLICATIONS*, 25(5):3265–3281, OCT 2022. ISSN 1386-7857. doi:10.1007/s10586-022-03561-y.

[43] Anurina Tarafdar, Mukta Debnath, Sunirmal Khatua, and Rajib K. Das. Energy and makespan aware scheduling of deadline sensitive tasks in the cloud environment. *JOURNAL OF GRID COMPUTING*, 19(2), JUN 2021. ISSN 1570-7873. doi:10.1007/s10723-021-09548-0.

[44] Shuaishuai Liu, Xinyu Ma, Yuanfei Jia, and Yue Liu. An energy-saving task scheduling model via greedy strategy under cloud environment. *WIRELESS COMMUNICATIONS & MOBILE COMPUTING*, 2022, APR 15 2022. ISSN 1530-8669. doi:10.1155/2022/8769674.

[45] Mehboob Hussain, Lian-Fu Wei, Abdullah Lakhan, Samad Wali, Soragga Ali, and Abid Hussain. Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing. *SUSTAINABLE COMPUTING-INFORMATICS & SYSTEMS*, 30, JUN 2021. ISSN 2210-5379. doi:10.1016/j.suscom.2021.100517.

[46] Navpreet Kaur Walia, Navdeep Kaur, Majed Alowaidi, Kamaljeet Singh Bhatia, Shailendra Mishra, Naveen Kumar Sharma, Sunil Kumar Sharma, and Harsimrat Kaur. An energy-efficient hybrid scheduling algorithm for task scheduling in the cloud computing environments. *IEEE ACCESS*, 9:117325–117337, 2021. ISSN 2169-3536. doi:10.1109/ACCESS.2021.3105727.

[47] Lingjuan Ye, Yuanqing Xia, Siyuan Tao, Ce Yan, Runze Gao, and Yufeng Zhan. Reliability-aware and energy-efficient workflow scheduling in iaas clouds. *IEEE*

TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, 2022. ISSN 1545-5955. doi:10.1109/TASE.2022.3195958.

[48] J. Kok Konjaang, John Murphy, and Liam Murphy. Energy-efficient virtual-machine mapping algorithm (evima) for workflow tasks with deadlines in a cloud environment. *JOURNAL OF NETWORK AND COMPUTER APPLICATIONS*, 203, JUL 2022. ISSN 1084-8045. doi:10.1016/j.jnca.2022.103400.

[49] Emmanuel Bugingo, Wei Zheng, Zhenfeng Lei, Defu Zhang, Samuel Rene Adolphe Sebakara, and Dongzhan Zhang. Deadline-constrained cost-energy aware workflow scheduling in cloud. *CONCURRENCY AND COMPUTATION-PRACTICE & EXPERIENCE*, 34(6), MAR 10 2022. ISSN 1532-0626. doi:10.1002/cpe.6761.

[50] Anurina Tarafdar, Kamalesh Karmakar, Rajib K. Das, and Sunirmal Khatua. Multi-criteria scheduling of scientific workflows in the workflow as a service platform. *COMPUTERS & ELECTRICAL ENGINEERING*, 105, JAN 2023. ISSN 0045-7906. doi:10.1016/j.compeleceng.2022.108458.

[51] Deafallah Alsadie. A metaheuristic framework for dynamic virtual machine allocation with optimized task scheduling in cloud data centers. *IEEE ACCESS*, 9:74218–74233, 2021. ISSN 2169-3536. doi:10.1109/ACCESS.2021.3077901.

[52] Sampa Sahoo, Bibhudatta Sahoo, and Ashok Kumar Turuk. A learning automata-based scheduling for deadline sensitive task in the cloud. *IEEE TRANSACTIONS ON SERVICES COMPUTING*, 14(6):1662–1674, NOV 1 2021. ISSN 1939-1374. doi:10.1109/TSC.2019.2906870.

[53] Radu Prodan, Marek Wieczorek, and Hamid Mohammadi Fard. Double auction-based scheduling of scientific applications in distributed grid and cloud environments. *Journal of Grid Computing*, 9(4):531–548, 2011.

[54] Weiwei Kong, Yang Lei, and Jing Ma. Virtual machine resource scheduling algorithm for cloud computing based on auction mechanism. *Optik*, 127(12):5099–5104, 2016.

[55] Mustafa Gamsız and Ali Haydar Özer. An auction based mathematical model for

energy-aware virtual machine allocation in clouds. In *2019 4th International Conference on Computer Science and Engineering (UBMK)*, pages 1–6. IEEE, 2019.

[56] Mustafa Gamsız and Ali Haydar Özer. An energy-aware combinatorial virtual machine allocation and placement model for green cloud computing. *IEEE Access*, 9: 18625–18648, 2021.

[57] R. Ghafari, F. Hassani Kabutarkhani, and N. Mansouri. Task scheduling algorithms for energy optimization in cloud environment: a comprehensive review. *CLUSTER COMPUTING-THE JOURNAL OF NETWORKS SOFTWARE TOOLS AND APPLICATIONS*, 25(2):1035–1093, APR 2022. ISSN 1386-7857. doi:10.1007/s10586-021-03512-z.

[58] Miyuru Dayarathna, Yonggang Wen, and Rui Fan. Data center energy consumption modeling: A survey. *IEEE Communications Surveys & Tutorials*, 18(1):732–794, 2015.

[59] Enrique Jaureguialzo. Pue: The green grid metric for evaluating the energy efficiency in dc (data center). measurement method using the power demand. In *2011 IEEE 33rd International Telecommunications Energy Conference (INTELEC)*, pages 1–8, 2011. doi:10.1109/INTLEC.2011.6099718.

[60] Luiz André Barroso and Urs Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture*, 4(1):1–108, 2009.

[61] Amazon. Amazon elestic compute cloud, 2023. URL `https://aws.amazon.com/ec2/`. [Online; accessed 22 Mar 2023].

[62] Erbil Öner and Ali Haydar Özer. Software repository for the eco-cabs model, 2023. URL `https://github.com/ahozer/eco-sched`. [Online; accessed 1 Apr 2023].

[63] David E Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of genetic algorithms*, volume 1, pages 69–93. Elsevier, 1991.

[64] Amazon. Amazon ec2 user guide - placement groups, 2023. URL https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/placement-groups.html. [Online; accessed 22 Mar 2023].

[65] Google. Google cloud compute engine user guide - instance groups, 2023. URL https://cloud.google.com/compute/docs/instance-groups#managed_instance_groups. [Online; accessed 22 Mar 2023].

[66] Microsoft. Microsoft azure user guide - virtual machine scale sets, 2023. URL https://learn.microsoft.com/en-us/azure/virtual-machine-scale-sets/flexible-virtual-machine-scale-sets-portal. [Online; accessed 22 Mar 2023].

[67] Gurobi Optimization. Gurobi Optimizer Reference Manual, 2023. URL https://www.gurobi.com.

[68] Amazon Web Services. EC2 On-Demand Instance Pricing – Amazon Web Services, 8 2022. URL https://aws.amazon.com/ec2/pricing/on-demand/?nc1=h_ls. [Online; accessed 8 Aug 2022].

[69] Strom-Report. Electricity Prices in Europe - Who pays the most? [2010 - 2021], 08 2022. URL https://strom-report.de/electricity-prices-europe. [Online; accessed 8 Aug 2022].

[70] B. L. Welch. On the comparison of several mean values: An alternative approach. *Biometrika*, 38(3/4):330–336, 1951.