Received January 2, 2021, accepted January 17, 2021, date of publication January 25, 2021, date of current version February 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3054559

An Energy-Aware Combinatorial Virtual Machine Allocation and Placement Model for Green Cloud Computing

MUSTAFA GAMSIZ AND ALİ HAYDAR ÖZER, (Member, IEEE)
Department of Computer Engineering, Faculty of Engineering, Marmara University, 34722 Istanbul, Turkey

Corresponding author: Ali Haydar Özer (haydar.ozer@marmara.edu.tr)

This work was supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under Grant 119E007.

ABSTRACT Resource allocation is an important problem for cloud environments. This paper introduces an energy-aware combinatorial auction-based model for the resource allocation problem in clouds. The proposed model allows users of a cloud to submit their virtual resource requests as bids using the provided bidding language which allows complementarities and substitutabilities among those resources to be declared. The model finds the most profitable mutually satisfiable set of winning bids, and the corresponding allocation of virtual resources to the users while considering the placement of virtual resources to the available physical resources in the cloud by executing an optimization problem. During the optimization, the model also takes account of the non-linear energy requirements of the physical resources based on their utilization levels to find a placement with the lowest energy cost, thus, providing an energy-aware solution to the resource allocation problem. The associated optimization problem is formally defined and formulated using integer programming. Since the optimization problem is intractable, four heuristic methods are also proposed. To evaluate the performance of the model and the proposed heuristic methods, several experiments are conducted on a comprehensive test suite. The results demonstrate the benefits of the proposed model, and the high-quality solutions provided by the proposed methods.

INDEX TERMS Cloud computing, combinatorial auction, energy-aware, green cloud, heuristic method, resource allocation, linear integer programming, virtual machine consolidation, virtual machine placement.

I. INTRODUCTION

Cloud computing provides a flexible platform that offers various infrastructure, platform, and software services [1]–[3]. Supporting the pay-as-you-go pricing model, cloud computing is attractive for private and public institutions [1]. Cloud providers offer many different types of cloud services, from software to the artificial intelligence platform, from virtual machines to storage systems. These services are generally provided by using data centers with a large number of pieces of computing equipment [4]. Data centers are energy-intensive facilities with an average rack density of 8.2 kW as indicated in the 2020 State of the Data Center report [5] which can even reach 43 kW per rack using efficient water-cooling techniques [6]. Data centers located in the US are estimated to consume 135 billion kWh of energy annually [7], and worldwide annual data center energy usage is estimated between 200 TWh [8], [9] to 500 TWh [10] which corresponds to approximately 1% of the global electricity use [8]. Andrae and Edler [11] estimate that by the year 2030, data centers will use around 3–13% of global electricity.

Energy costs constitute a significant share of the cloud providers' budgets. For example, according to Amazon.com, the amount of energy-related costs, including direct power consumption and power consumption of the cooling infrastructure, accounts for 42% of the total budget [12]. Apart from the high cost of energy usage, its negative contribution to global warming should be taken into account because of the increased carbon emissions. The information technology sector is responsible for 2% of global carbon emissions [13], inside this, data centers alone are estimated to be responsible for 0.3% of the global carbon emissions [9]. Additionally, data centers are estimated to have the fastest-growing carbon footprint across the entire IT sector [14]. For sustainable cloud platforms, cloud providers should also consider the carbon footprint of cloud infrastructures.

The associate editor coordinating the review of this manuscript and approving it for publication was Fan-Hsun Tseng.

The most effective way of reducing energy usage of data centers, and hence their carbon emissions is to increase the energy efficiency of data centers. The Uptime Institute's Intelligence Report 2020 [10] states that the most effective way for increasing energy efficiency is to increase the utilization levels of servers by consolidating the workload to as few servers as possible. For instance, for a power usage effectiveness value (PUE) [15] of 2.0, increasing the utilization level of servers up to 1.5 years old from 5% to 25% causes a reduction in energy consumption by 65%. However, reducing the PUE from 2.0 to 1.5 results in only a 25% reduction also requiring additional investment. Based on the data from over 300 data centers studied under the EURECA project [16], it is estimated that the utilization levels of the servers are around 25% [10]. Thus, significant energy savings can be obtained by increasing the utilization levels of servers.

Therefore, efficient use of hardware resources by improving the utilization rate of resources in data centers plays an important role in increasing energy efficiency. However, this is not a straightforward task to implement. As stated by Yousafzai *et al.* [17], the resource pools in the data centers have a very heterogeneous structure, and as the technology advances, cloud providers have to adopt technological changes to data centers. Thus, as the variety of resource types in the data centers increases, increasing the efficiency of the resources requires more effort. Furthermore, geographically distributed heterogeneous resources owned by large cloud providers and demand elasticity provided by the cloud systems cause difficulty in resource allocation in cloud systems compared to the homogeneous clusters [18]. In the literature, the resource allocation problem is defined as the economical allocation of the resources requested by cloud users while satisfying the Service Level Agreements of the users [17].

In this study, an Energy-Aware Combinatorial Auction based Virtual Machine Allocation and Placement Model (ECO-VMAP), is proposed for the resource allocation problem in cloud systems to increase the energy efficiency of cloud infrastructures and to reduce their carbon footprint, thereby ensuring the efficient use of hardware resources and minimizing energy use. This model associates multiple virtual machine requests of users with a subset of the hardware that will use as little energy as possible, providing resource allocation which meets their service level agreements.

The proposed ECO-VMAP model is a combinatorial auction-based model, more specifically, is based on the multi-unit nondiscriminatory combinatorial auction mechanism presented in [19] which is designed especially for efficient resource allocation. In this model, by means of this mechanism users can declare their complex preferences for virtual machines offered in the cloud using the provided bidding language. The bidding language enables users to request a bundle of virtual machines in a single bid, and also to declare complementarities and substitutabilities among the requested virtual machines. Thus, this language allows users to request virtual machines flexibly for a variety of workloads such as web and database applications, high-performance computing, and data storage. Along with each bid declared, users offer an associated price value which indicates the gain to be obtained for satisfying the corresponding bid. For commercial cloud environments, this value may indicate the amount to be paid by the owner of the bid if the bid is satisfied, for non-commercial clouds, on the other hand, it may indicate the priority of a job.

The objective of the model is to determine the best possible allocation of virtual machines based on the bids of the users while considering the placement of the allocated virtual machines to the available physical servers to minimize the energy consumption and while providing the required quality of service. That is, the model simultaneously finds (i) the optimum allocation of virtual machines to the users based on their submitted bids, and (ii) the optimum placement of the allocated virtual machines to the physical servers such that the energy cost is minimized.

A typical physical server has a non-linear power usage function based on its utilization level since an idle server can draw from 17.5% (for the servers up to 1.5 years old) to more than 26% (for the servers that are 5 to 10 years old) of the power it draws when it is fully utilized [10].[1] Also, as will be introduced in Section III in detail, even if the idle power is ignored, the power usage function is still a non-linear function of the server utilization level. Therefore, the proposed model is designed to incorporate the non-linear power usage characteristics of physical servers for finding an energy-efficient placement of the VM instances to the physical servers. Thus, based on the allocated virtual machines, each physical server is either kept at its optimum utilization levels based on each server's power usage function or turned off. Although primarily designed for virtual machine allocation, i.e., for hardware-level virtualization, we consider that the model can also be used to allocate containers with well-defined resource limits, i.e., for operating system virtualization, which is also explained in Section III.

By means of the introduced mechanisms, the ECO-VMAP model aims to increase the expected profit of the cloud providers while satisfying the demands of the users in the cloud environment and reducing the energy usage of the physical resources for providing an environment-friendly and sustainable cloud computing service.

The paper is organized as follows. A brief literature survey is provided in the next section. In Section III, an example cloud computing scenario is given, and using this scenario, the ECO-VMAP model is explained in detail. Section IV introduces the mathematical definition of the model, the associated optimization problem, and its integer programming formulation. This section also includes the complexity analysis of this optimization problem. An alternative formulation for the optimization problem of the ECO-VMAP model which allows different-sized VMs or containers to be allocated on a single physical machine is provided in Section V.

[1]This study also indicates that approximately 40% of the servers deployed in the studied data centers are older than five years.

Since the corresponding optimization problem is proven to be NP-hard, several heuristic methods are proposed for the model which are explained in Section VI. To estimate the real-life performance of the model along with the proposed heuristic methods, a test case generator is implemented, and a comprehensive test suite is prepared. The results of the conducted experiments using this test suite are presented in Section VII. Finally, the paper is concluded in Section VIII.

II. RELATED WORK AND A BRIEF INTRODUCTION TO COMBINATORIAL AUCTIONS

There are many studies in the literature on virtual machine placement or resource allocation considering energy efficiency in clouds [17]. For example, Beloglazov *et al.* [20], Hasan and Huh [21], and Yazir *et al.* [22] proposed virtual machine allocation solutions to increase the utilization levels of servers and to decrease energy consumption based on a CPU power usage model. The proposed solutions are based on the consolidation of virtual machines dynamically during execution to reduce the power used by the infrastructure while satisfying service level agreements. Yin *et al.* [23], on the other hand, proposed that instead of using only the CPU power usage metric, an energy usage metric should be defined for physical servers based on the combined CPU-memory-network load of virtual machines and physical servers. Bessai *et al.* [24] examined the problems of mapping and scheduling user interactive workflows to resources on the cloud while considering environmental impacts. Lee and Zomaya [25], on the other hand, proposed a solution for geographically distributed data centers where parallel jobs which are defined as workflows of data center and service level algorithms are placed in the cloud with a focus on energy use and performance. In these studies, the heterogeneity of the cloud infrastructure has not been taken into consideration. The work of Lee *et al.* [26], however, considered heterogeneous resources, and an algorithm called the performance analysis-based VM Scheduling algorithm, which takes into account users' requests and utilization levels of servers, was proposed to place heterogeneous virtual machines on physical servers.

Zhao *et al.* [27] integrated a physical machine power consumption model and VM performance model in a multi-objective optimization problem for finding an optimal placement of VMs to physical servers. The proposed optimization problem is NP-Hard and thus they proposed an ant colony optimization-based solution for this problem. Similar to the work of Zhao *et al.* [27], Ye *et al.* [28] also define a multi-objective optimization problem for finding an optimal VM placement but using different objectives. They consider load balancing, resource utilization, and robustness in addition to energy consumption and the energy model used is a linear model based on CPU utilization. For solving this problem, an evolutionary algorithm is proposed. In the work in [29], a VM placement framework has been proposed which considers the quality of service agreements, performance goals, and security policies while reducing energy consumption. To reduce energy consumption, the VM placement framework aims to maximize the number of physical machines with no VMs running on them in a data center so that the idle servers can be turned off to save energy. A similar energy consumption model is also used in the work of Liu *et al.* [30] which aims to find a placement of VMs to a minimum number of active servers to reduce the energy consumption. For this purpose, an ant colony system is proposed to solve the introduced optimization model. In the work of Wang *et al.* [31], the authors propose a VM placement solution that considers the quality of service requirement along with an energy minimization objective. Different from the previous works, a particle swarm optimization-based heuristic solution is proposed for the proposed energy and quality of service-aware optimization approach. In a more recent study, Liu *et al.* [32] introduced a constrained optimization problem for an energy-aware and availability-aware solution to the VM allocation problem. The proposed model considers the following parameters in different objectives: energy consumption of physical servers, availability of virtual cluster, the mean resource utilization of the physical servers, and load-balance among the physical servers. They also proposed a population-based heuristic method to find a solution to the defined optimization problem. For further discussion about the VM placement or resource allocation problem considering energy efficiency in clouds, the reader is referred to the surveys in [33]–[40].

Auction-based approaches are also proposed for the resource allocation problem in clouds due to their greater allocative efficiency.[2] Sheikholeslami and Navimipour [35] provide a survey on auction-based resource allocation mechanisms which they categorize into four groups based on the auction mechanism used: one-sided, double-sided, combinatorial auction-based, and other types of auction-based mechanisms. Among these auction mechanisms, combinatorial auctions have particular importance for cloud resource allocation since it allows package bidding, that is, the bidders can bid not only on a single good but on combinations of different goods [42], [43]. In this model, all goods are open to all bidders, and bidders are free to express their valuations for any combination of goods. For example, a bidder can define a package containing a television and a satellite receiver, and submit a bid for that package. The amount proposed in the bid reflects the amount that the bidder will pay only if she can receive the entire package. Combinatorial auctions can be applied to many different areas such as radio spectrum rights [44], airport time slots [45], logistics services [46], and students' course registrations [47].

The combinatorial auction mechanism allows bidders to express their complex preferences which is important especially when there are complementarities among the goods sold [43]. This is widely seen in cloud-based resources. Usually, users of a cloud need not only a single resource but a

[2]Note that auctions provide higher expected revenue to the sellers compared to the fixed-price mechanism (see [41] for a review).

resource set (various virtual machines, storage space, network capacity, licenses, etc.) simultaneously. For example, if only a subset of the virtual machines required in a high-performance computing job is allocated, the job cannot be finished on time. Similarly, for a user to run a web application, several different servers may be required such as load balancing servers, front-end and back-end servers, database servers, and storage space simultaneously. For such users, these are complementary resources. These examples can be increased according to different usage scenarios. In complementary markets, combinatorial auctions increase economic efficiency and provide higher profit rates to resource providers [43].

There are several studies in the literature that utilize combinatorial auction techniques for cloud resource allocation. In their work, Zaman and Grosu [48] proposed a combinatorial auction-based model for VM allocation in clouds. The model requires a weight value to be assigned to each VM type available in the cloud, based on their respective computational power. Each potential user places a bid for a bundle of VMs with different weight values indicating how many VMs she needs. The proposed model does not have an aim to maximize the revenue of the cloud provider, and hence it does not have an objective function. The authors propose two truthful mechanisms called CA-LP (truthful in expectation) and CA-GREEDY to determine the winning bids finding a static allocation of VMs. In this model, it is assumed the bidders are single-minded, that is in one auction round they can only submit one bid. The model also does not consider the energy costs of the physical hardware. In their subsequent study, Zaman and Grosu [49] extended their work to support the dynamic provisioning of VM instances. The extended model also considers the costs of running or idle VMs for finding the allocation of VMs. They propose a truthful mechanism called CA-PROVISION which finds an approximate solution for the introduced resource allocation problem. The assumptions of weight-based definitions of VMs and single-minded users are also valid for this extended model. The model further assumes that the running cost and the idle cost of VMs are constant, independent of on which hardware they are being executed. Note that, the energy cost per unit time of each VM varies depending on the server hardware that the VM is currently mapped to. Huu and Tham [50] proposed another combinatorial auction-based model for resource allocation. They propose a mixed-integer program for maximizing the sum of users' bid prices the optimization of which is an NP-hard problem. Therefore, they provide three truthful greedy algorithms for determining the winning bids. Similar to the work of Zaman and Grosu [48], the bidders are assumed to be single-minded, the placement of the VMs to the physical servers is not considered, and hence, the costs of VMs are constant. Wang and Huang [51] proposed another combinatorial auction-based resource allocation method for clouds which they call the Multi-Attribute Auction Mechanism. In this method, the frameworks of the allocation mechanism are created first, and then auction descriptions are given. Secondly, with a Support Vector Regression based method, the request set in users' offers is converted into multiple single-source requests. Thirdly, these requests are processed, and the winning offers are determined. In this work, although the combinatorial auction-based mechanism is used, energy efficiency has not been taken into account in resource allocation. Finally, Tan *et al.* [52] proposed an optimal posted price-based mechanism online version of the combinatorial auction mechanism considering capacity limits and supply costs. The proposed mechanism is optimal in the way that no other mechanism can achieve a better competitive ratio, however, energy efficiency is not considered. A more detailed survey of auction-based mechanisms for the resource allocation problem can be found in [35].

Although the original combinatorial auction model which is used in the previous works allows users to bid for the resources they are interested in, it does not allow the expression of the resources that the users consider as alternatives. For example, one user may be interested in a set of virtual machines with high computational powers without differentiating which datacenter or region the virtual machines are located. Another user, on the other hand, may require a set of virtual machines located in a specific region for executing a web application aimed towards the clients located in the same region. However, she may not differentiate different virtual machine types as long as they are located in the specific data center and satisfy the required computational power, memory, and storage space. The original combinatorial auction does not have a direct mechanism for expressing these preferences of the users for alternative resources.[3] Formally, the original combinatorial auction model allows users to define logical AND operation between the resources they want, however, does not support logical OR operation in its bidding language.

This limitation may decrease the efficiency of the allocation of resources that will be determined by the combinatorial auction model. To overcome this limitation, Özer and Özturan [19] proposed a multi-unit nondiscriminatory combinatorial auction model that comprises a novel bidding language that supports both logical AND and OR operations so that users can express their preferences for equivalent or alternative goods.

The proposed ECO-VMAP model[4] differentiates from the previous work as follows. The model is based on the multi-unit nondiscriminatory combinatorial auction model in which the bidders may express their valuations for any combination of VM types declaring complementarities and substitutabilities among the requested VMs. The model aims to find an allocation of VMs by maximizing the sum of the offered prices of the mutually satisfiable set of bids of the users while simultaneously finding the optimal placement of VMs to the physical servers such that their energy

[3]This kind of preferences can be encoded in the original combinatorial auctions using dummy items and exponential number of bids which is not feasible in practice. See [19] for detailed discussion.

[4]A preliminary version of the paper which includes only a rudimentary model description has been presented at the 2019 4th International Conference on Computer Science and Engineering (UBMK 2019) [53].

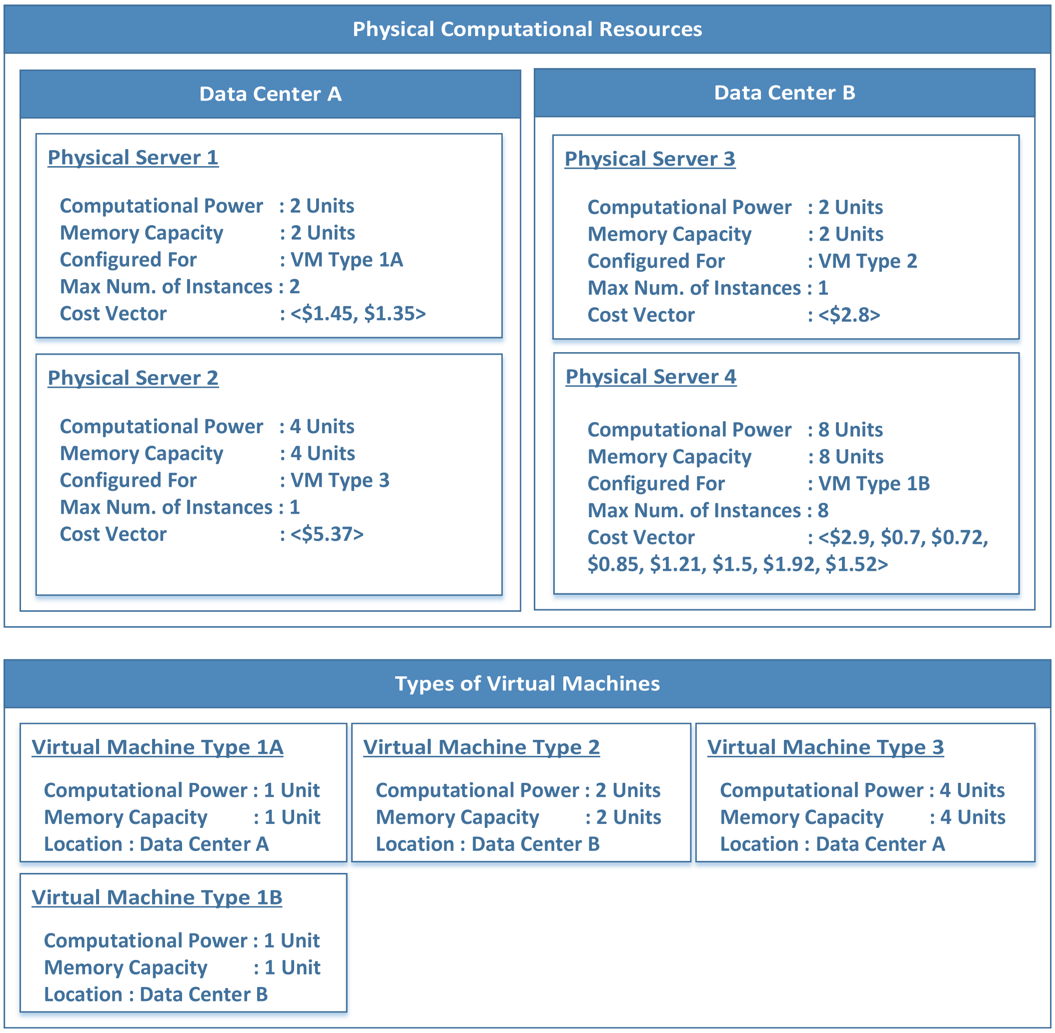

Physical Computational Resources
Data Center A
Data Center B
Physical Server 1
Computational Power : 2 Units
Memory Capacity : 2 Units
Configured For : VM Type 1A
Max Num. of Instances : 2
Cost Vector : <$1.45, $1.35>
Physical Server 2
Computational Power : 4 Units
Memory Capacity : 4 Units
Configured For : VM Type 3
Max Num. of Instances : 1
Cost Vector : <$5.37>
Physical Server 3
Computational Power : 2 Units
Memory Capacity : 2 Units
Configured For : VM Type 2
Max Num. of Instances : 1
Cost Vector : <$2.8>
Physical Server 4
Computational Power : 8 Units
Memory Capacity : 8 Units
Configured For : VM Type 1B
Max Num. of Instances : 8
Cost Vector : <$2.9, $0.7, $0.72, $0.85, $1.21, $1.5, $1.92, $1.52>
Types of Virtual Machines
Virtual Machine Type 1A
Computational Power : 1 Unit
Memory Capacity : 1 Unit
Location : Data Center A
Virtual Machine Type 2
Computational Power : 2 Units
Memory Capacity : 2 Units
Location : Data Center B
Virtual Machine Type 3
Computational Power : 4 Units
Memory Capacity : 4 Units
Location : Data Center A
Virtual Machine Type 1B
Computational Power : 1 Unit
Memory Capacity : 1 Unit
Location : Data Center B

FIGURE 1. Physical computational resources and virtual machine types defined in the example scenario demonstrating the ECO-VMAP model.

consumption is minimized. The model also incorporates the non-linear power usage characteristics of physical servers for energy calculation. A detailed explanation of the ECO-VMAP model is provided in the next section.

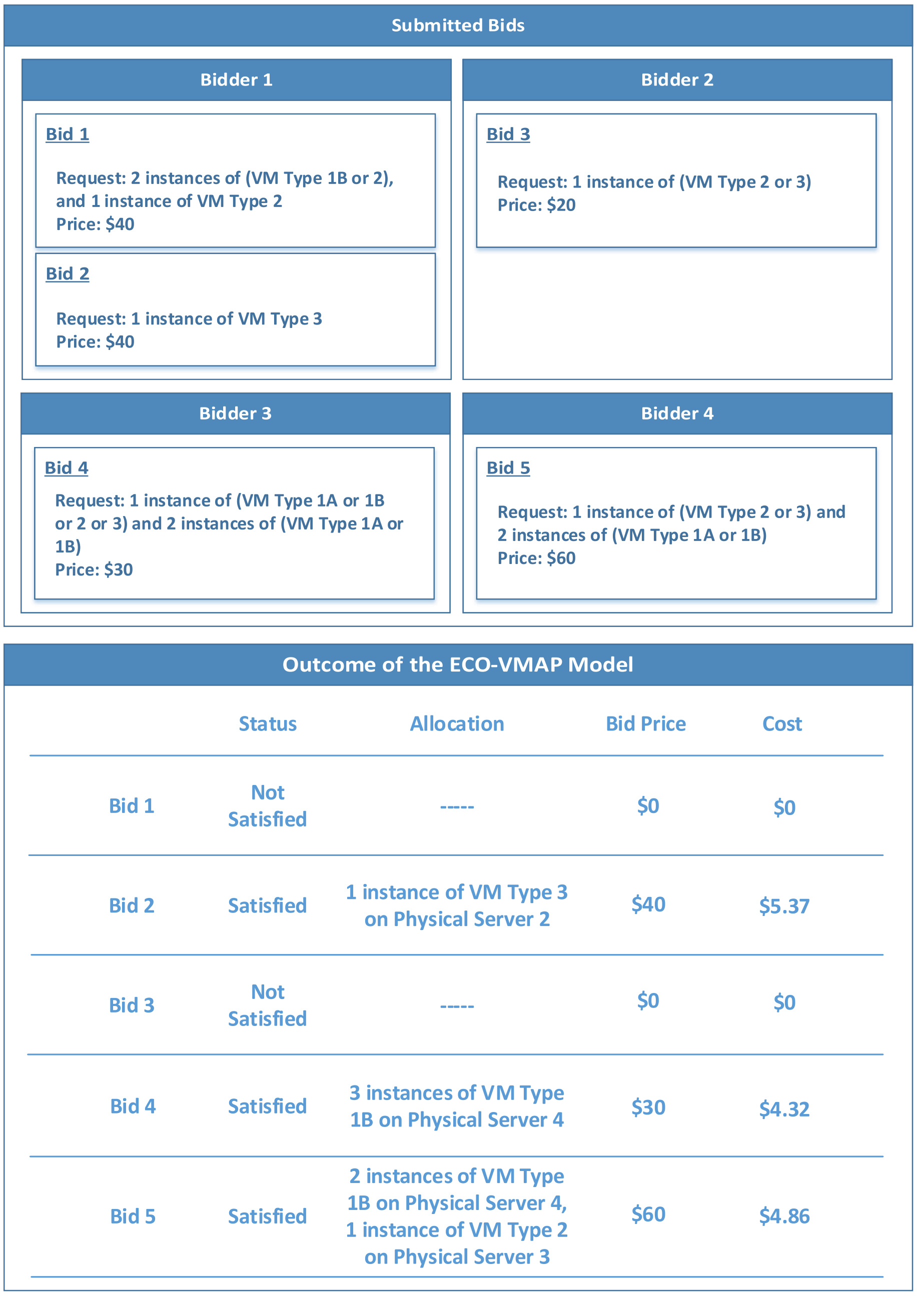
III. THE ECO-VMAP MODEL

In the ECO-VMAP model, a cloud infrastructure comprises one or more data centers, and each data center comprises a number of physical servers. To calculate the energy costs accurately, the ECO-VMAP model considers each physical server as unique, since even two identical servers do not draw the same amount power when their load levels are different. Each physical server is configured to execute a predefined VM type.

The ECO-VMAP model needs the specifications of neither the virtual machine types nor the physical servers. The model requires the following information:

(i) a list of physical servers (e.g. a list of unique identifiers of physical servers)
(ii) a list of VM types (e.g. a list of unique identifiers of VM types)
(iii) a mapping of which physical server is configured to execute which virtual machine,
(iv) for each physical server, the maximum number of VMs that can be executed simultaneously in that physical server,
(v) for each physical server, a cost vector defining the energy cost of physical servers under different load levels (based on the number of VMs executing simultaneously) during the allocation period, i.e., the period in which the VMs will be allocated to the users. This data can be obtained, for instance, by using a power meter measuring the power consumption of a server under different load levels.

If there are multiple data centers in the cloud, virtual machine types can be replicated for each data center. This allows users to specify the data center in which their VMs to be executed if they are willing to. If not, then this does not



Submitted Bids

Bidder 1

Bid 1

Request: 2 instances of (VM Type 1B or 2), and 1 instance of VM Type 2
Price: $40

Bid 2

Request: 1 instance of VM Type 3
Price: $40

Bidder 2

Bid 3

Request: 1 instance of (VM Type 2 or 3)
Price: $20

Bidder 3

Bid 4

Request: 1 instance of (VM Type 1A or 1B or 2 or 3) and 2 instances of (VM Type 1A or 1B)
Price: $30

Bidder 4

Bid 5

Request: 1 instance of (VM Type 2 or 3) and 2 instances of (VM Type 1A or 1B)
Price: $60

Outcome of the ECO-VMAP Model

	Status	Allocation	Bid Price	Cost
Bid 1	Not Satisfied	-----	$0	$0
Bid 2	Satisfied	1 instance of VM Type 3 on Physical Server 2	$40	$5.37
Bid 3	Not Satisfied	-----	$0	$0
Bid 4	Satisfied	3 instances of VM Type 1B on Physical Server 4	$30	$4.32
Bid 5	Satisfied	2 instances of VM Type 1B on Physical Server 4, 1 instance of VM Type 2 on Physical Server 3	$60	$4.86

FIGURE 2. **Bids of the users, the outcome of the ECO-VMAP model, and allocation of VMs for the example scenario.**

cause additional difficulty for the users who are not interested in the locations of the physical servers on which their VMs are executing by means of the nondiscriminatory combinatorial auction mechanism used in the model.

An example scenario for the ECO-VMAP model is provided in Figure 1 and Figure 2. There are two data centers, Data Center A and Data Center B each of which hosts two physical servers. There are three types of virtual machines, VM Type 1, 2, and 3, defined with different computational power and memory capacity. Furthermore, VM Type 1 is replicated for each data center A and B since the same VM can be executed in both data centers. This replication technique allows users to select data centers in which the requested VMs will be executed as explained above. If a virtual machine type cannot be executed in a data center X then no replication is necessary for that data center. Note that, the virtual machine types can further be diversified based on other possible features besides CPU and memory such as storage, GPU, and network without any limitation.

Each physical server is configured to execute a virtual machine type. The number of VM instances that a physical server can execute simultaneously depends on the specifications of the physical server and the VM type, and the quality of service requirements. For instance, Physical Server 1 is configured for VM Type 1 (VM Type 1A with location suffix). Since the Physical Server 1 has twice the computational power and memory capacity needed from the VM Type 1, it can execute 2 instances of VM Type 1 simultaneously (note that this number should be determined based on the quality of service requirements). Physical Server 4, on the other hand, is more powerful, and it can execute 8 instances of VM Type 1 (VM Type 1B with the location suffix) simultaneously.

When the physical computational resources and the virtual machine types are defined, the potential users of the cloud can request VMs to run their applications. The first objective of the ECO-VMAP model is to allow users to declare their complicated VM requests to the cloud provider to find the best suitable allocation of VMs. For this purpose, the ECO-VMAP model uses a nondiscriminatory variant of the well-known combinatorial auction mechanism proposed in [19]. The bidding language provided in this auction mechanism supports both logical AND and logical OR requests inside the bids of the user, allowing them to declare complementarities among their VM requests, i.e., they can bundle their VM requests in a single package. Furthermore, they are also allowed to declare substitutabilities among their VM requests.

In this scenario, four bidders submit five bids to the auctioneer (i.e., the cloud provider) as seen in Figure 2. For instance, in Bid 1, Bidder 1 requests 2 instances of VM Type 1B or 2 without differentiating these two VM types in the first subbid of her bid. However, she also simultaneously requests one instance of VM Type 2 in her same bid as the second subbid. These subbids in her bid are connected by logical AND relationship meaning that this bid can be satisfied if these two subbids can be satisfied together, that is the requested VMs in these two subbids can be allocated to the user. The bidder declares that she will pay \$40 if her bid is satisfied. If this bid is satisfied, the allocation outcome would be one of the following:

- One instance of VM Type 1B and two instances of VM Type 2; or
- Two instances of VM Type 1B and one instance of VM Type 2; or
- Three instances of VM Type 2 are allocated to the bidder during the allocation period.

In any other allocation outcome, the user does not pay the bid price.

The bids can be presented in a formal bidding format as follows:

- $B_1 = [\{\langle(V_{1B}, V_2), 2\rangle, \langle(V_2), 1\rangle\}, \$40]$
- $B_2 = [\{\langle(V_3), 1\rangle\}, \$40]$
- $B_3 = [\{\langle(V_2, V_3), 1\rangle\}, \$20]$
- $B_4 = [\{\langle(V_{1A}, V_{1B}, V_2, V_3), 1\rangle, \langle(V_{1A}, V_{1B}), 2\rangle\}, \$30]$
- $B_5 = [\{\langle(V_2, V_3), 1\rangle, \langle(V_{1A}, V_{1B}), 2\rangle\}, \$60]$

In this notation, [ ] encloses the bid itself, $\langle\ \rangle$ encloses the subbids separated with commas which are connected in AND relationship, ( ) encloses the requested alternative VMs (in OR relationship), and finally, the last value indicates the offered price for the bid.

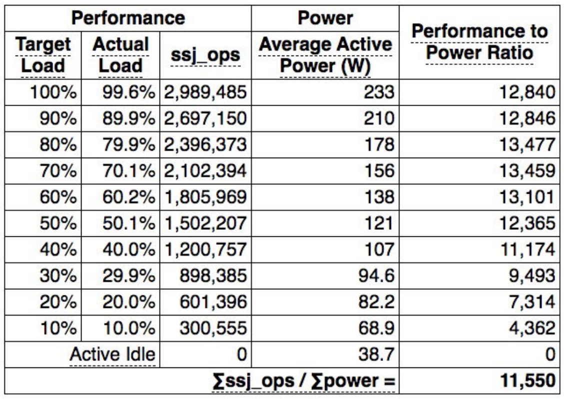
The benefit of replicating the VM types for each data center location can be seen in Bid 1 and Bid 5. In Bid 1, Bidder 1 requests three VMs such that the physical servers on which the VMs will be executed are located in Data Center B. The reason for such a request may be that Bidder 1 wants to run an e-commerce web application to serve the customers in the same region as Data Center B. However, in Bid 5, Bidder 4 requests 3 VMs without differentiating the location of the physical servers that will execute the VMs.

Additionally, the bidding language used in the ECO-VMAP model also supports the (OR*) bidding scheme [54]. It is also possible for users to declare the XOR relationship among their bids so that at most one of the bids can be satisfied. For instance, if she wants so, Bidder 1 may also declare that only one of the Bid 1 and Bid 2 to be satisfied.

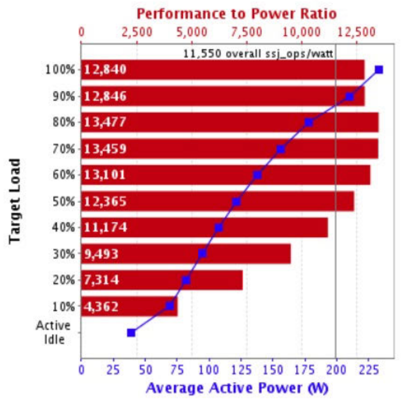
Besides finding an allocation of VMs based on the users' preferences, the second objective of the ECO-VMAP model is to provide an energy-aware mapping of the allocated VMs to the physical servers available in the cloud to reduce the total energy consumption. For this purpose, the power requirements of the physical servers should be defined in the model. The power consumption of a physical server is a function of the utilization level of the server. However, this function is not linear. Figure 3 shows the energy consumption characteristics of three different commercial servers (with a different number of execution cores) from a well-known world-wide vendor under different utilization levels. Even when a server is idle, i.e., its utilization level is 0%, it uses a considerable amount of power. When the utilization level is gradually increased by 10% increments, the power consumption increases in a non-linear fashion. For these selected servers, at lower utilization levels (between 10% and 30%), the rate of increase is lower, however, at higher utilization levels (between 70% and 90%), the rate of increase is higher.

As noted above, depending on its specifications, a physical server can execute more than one VM due to the virtualization technology. Thus, the utilization level of a physical server depends on the number of VMs being executed on the server. Therefore, in the ECO-VMAP model, each physical server is assumed to contain VM slots the number of which is the maximum number of VMs that the server can execute simultaneously. For instance, in the example scenario Physical Server 1 can execute two VMs simultaneously, and therefore it has two VM slots; whereas Physical Server 4 can execute 8 VM instances, and hence it has 8 VM slots.

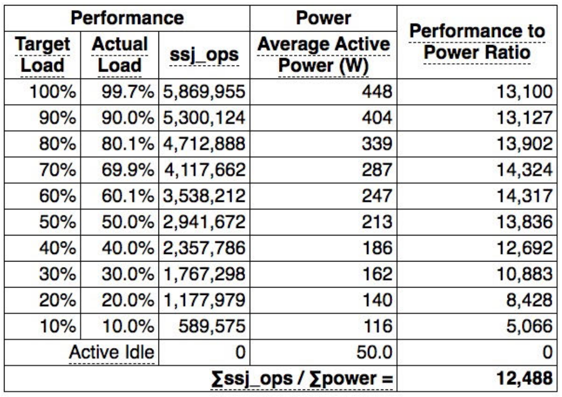
To define the energy cost in the ECO-VMAP model, a cost vector should be defined for each physical server based on the number of VMs that the server executes simultaneously. For instance, for Physical Server 1, the cost vector is defined as $< \$1.45, \$1.35 >$ indicating that the energy cost of the



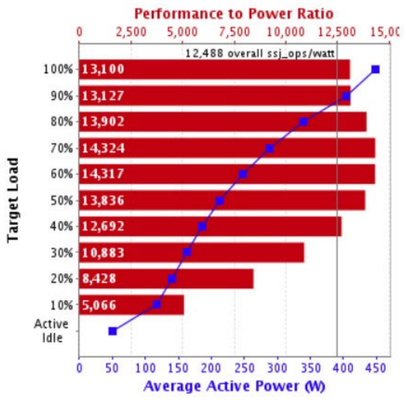
Performance			Power	Performance to Power Ratio
Target Load	Actual Load	ssj_ops	Average Active Power (W)	
100%	99.6%	2,989,485	233	12,840
90%	89.9%	2,697,150	210	12,846
80%	79.9%	2,396,373	178	13,477
70%	70.1%	2,102,394	156	13,459
60%	60.2%	1,805,969	138	13,101
50%	50.1%	1,502,207	121	12,365
40%	40.0%	1,200,757	107	11,174
30%	29.9%	898,385	94.6	9,493
20%	20.0%	601,396	82.2	7,314
10%	10.0%	300,555	68.9	4,362
Active Idle		0	38.7	0
		∑ssj_ops / ∑power =		11,550



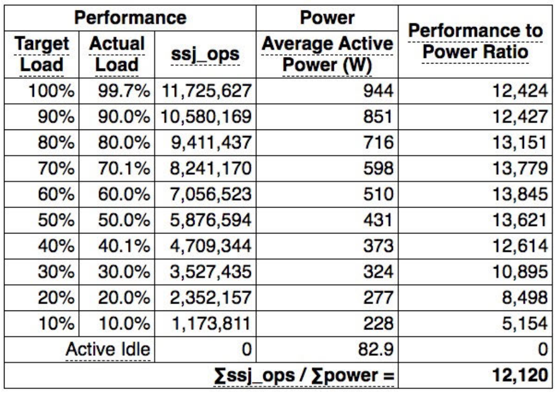
(a) SPECpower_ssj® 2008 results for a 28-core server hardware from a well-known vendor.



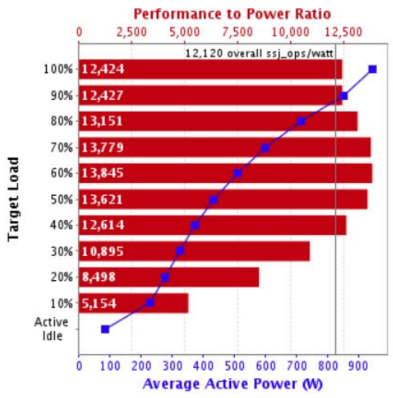
Performance			Power	Performance to Power Ratio
Target Load	Actual Load	ssj_ops	Average Active Power (W)	
100%	99.7%	5,869,955	448	13,100
90%	90.0%	5,300,124	404	13,127
80%	80.1%	4,712,888	339	13,902
70%	69.9%	4,117,662	287	14,324
60%	60.1%	3,538,212	247	14,317
50%	50.0%	2,941,672	213	13,836
40%	40.0%	2,357,786	186	12,692
30%	30.0%	1,767,298	162	10,883
20%	20.0%	1,177,979	140	8,428
10%	10.0%	589,575	116	5,066
Active Idle		0	50.0	0
		∑ssj_ops / ∑power =		12,488



(b) SPECpower_ssj® 2008 results for a 56-core server hardware from a well-known vendor.



Performance			Power	Performance to Power Ratio
Target Load	Actual Load	ssj_ops	Average Active Power (W)	
100%	99.7%	11,725,627	944	12,424
90%	90.0%	10,580,169	851	12,427
80%	80.0%	9,411,437	716	13,151
70%	70.1%	8,241,170	598	13,779
60%	60.0%	7,056,523	510	13,845
50%	50.0%	5,876,594	431	13,621
40%	40.1%	4,709,344	373	12,614
30%	30.0%	3,527,435	324	10,895
20%	20.0%	2,352,157	277	8,498
10%	10.0%	1,173,811	228	5,154
Active Idle		0	82.9	0
		∑ssj_ops / ∑power =		12,120



(c) SPECpower_ssj® 2008 results for a 112-core server hardware from a well-known vendor.

FIGURE 3. Benchmark results of three servers from a well-known vendor using SPECpower_ssj® benchmark [55].

physical server will be \$1.45 if only one VM is executed, and it will be \$2.8 if two VMs are executed simultaneously. On the other hand, the cost vector of Physical Server 4 has 8 cost values. If a single VM is executed on Physical Server 4, then the cost will only be \$2.9 whereas if all the VM slots are occupied the cost will be \$11.32. Note that the cost values indicate the cost of executing VM for the whole allocation period. These cost values in this example are calculated using the data available in Figure 3 for an allocation period of 24 hours and using a Power Usage Effectiveness (PUE) value of 2.4 which is calculated using Schneider's PUE calculator [56] for a data center that runs at 50% capacity. PUE value indicates the ratio of total power consumption of a data center to the power consumption of the computing equipment located in the data center. Note that the actual PUE values for hyper-scale data centers are smaller than this value [10], however, this value is just used for the example scenario for demonstration as obtained from Schneider's PUE calculator.

The outcome of the ECO-VMAP model for this example scenario can be seen in Figure 2. Bids 2, 4, and 5 are satisfied. One VM is allocated to Bidder 1, three VMs are allocated to Bidder 3, and finally, three VMs are allocated to Bidder 4. Physical Servers 2 and 3 runs at full capacity, Physical Server 4 uses 5 slots out of 8 slots available, and Physical Server 1 has no load, therefore it is deactivated (turned off). The sum of the bid prices is \$130, whereas the total cost is \$14.55. The benefit of the energy-saving feature of the ECO-VMAP model can also be seen in the outcome. If two instances of VM Type 1A were allocated to Bidder 4 instead of VM Type 1B, that is if the 2 VMs were placed to Physical Server 1 instead of Physical Server 4, the total cost would be \$15.27 instead of \$14.55.

An auction round consists of two periods in the ECO-VMAP model. In the first period, users submit their bids to the system indicating their preferences. During this period, they can modify or retract their bids. After the first period is over, the clearing period begins, and a market-clearing optimization process is carried out to determine the winning bids and the corresponding allocation of VMs to the users along with the placement of VMs to the physical servers. The optimization process will be explained in the next section. After an auction round is over, another round can begin. Users having unsatisfied bids at the end of a round can modify their bids and resubmit them for the next round. The length of the auction round may be determined in accordance with the allocation period of VMs.

A further note is that although the ECO-VMAP model is provided to find an allocation and placement of VM instances in a cloud, it also supports the allocation of other types of resources such as virtual storage, GPUs, and virtual network functions [57]. For instance, if a cloud infrastructure consists of a storage server with storage capacity X which is to be allocated to the VM instances as virtual storage, then a virtual storage type can be introduced in the model along with a special physical server that is configured for this virtual storage type. If the smallest amount of virtual storage that will be offered is Y, then the physical server should be defined to have X/Y slots.

Finally, although designed for VMs, we consider that the ECO-VMAP model can also be used for allocating containers. Relatively recent container technology offers a light-weight alternative for virtual machines [58]–[60]. VMs are emulated by a hypervisor that runs on a physical server, i.e., host machine. The hypervisor creates, runs, and destroys VMs. Each VM running on a physical server is isolated from the other VMs, uses its operating system, and can use the physical resources of the host machine under the control of the hypervisor. Containers, on the other hand, are deployed on a physical server and run on top of the operating system kernel instead of a hypervisor, that is containers are based on operating system virtualization instead of hardware virtualization. Since there is no operating system installed in a container, a container image is smaller than a VM image. Having small images and lacking a hypervisor layer, containers are considered more efficient and scalable comparing to VMs [61], [62]. However, the level of isolation is lower in containers. Therefore, containers are considered to be more vulnerable to possible attacks from malicious containers and prone to higher security risks [63], [64]. Although these risks can be reduced to a degree by using additional security mechanisms, these mechanisms would bring additional overhead reducing the performance and elasticity of container platforms [58], [63]. Another important issue yet to be solved within container technology is migration [65]. Besides the underlying operating system, the containers also share some libraries [66]. Thus, to migrate a number of containers to a destination host, the operating system of the host should support these libraries required by the containers. However, VMs can be migrated to a destination host as long as a compatible hypervisor exists in the host [58]. Thus, both VMs and containers have pros and cons, and although the container technology is promising, it is not dominant over the VM technology yet [67]. To have the best of two worlds, hybrid models can also be used in which containers are deployed on virtual machines for increased isolation and security [68], [69].

In order to use the ECO-VMAP model for container allocation, different container types can be defined instead of VM types. If containers are to be deployed directly on an operating system of a physical server, then the number of slots of the physical server can be defined as the number of containers to be allocated on the physical server considering the resources of the physical server. However, if instead of physical servers, containers are to be deployed on VMs, then each VM should be defined as a physical server. According to the capacity of each VM, the number of containers that can be hosted in the corresponding VM can be defined accordingly. Note that containers share the resources (e.g., CPU, memory, storage, and network) of the physical host similar to VMs but without dedicated reservations. This brings the possibility of resource-hungry processes to consume most of the resources of the host. In order to prevent such situations, container managers can limit the number of resources that a container can use, and can also pin containers to one or more physical or virtual CPUs available on the host [60], [69]. Finally, as will be introduced in Section V, it is also possible to place different sized containers, i.e., containers having different resource limits on a single physical server or VM using the ECO-VMAP model.

IV. THE MATHEMATICAL DEFINITION AND FORMULATION OF THE ECO-VMAP MODEL

The ECO-VMAP model is formally defined as follows. Let

- $P = \{p_1, p_2, \ldots, p_m\}$ be the set of m physical servers available in the cloud, where each server is considered as unique even though some of the servers have the same technical specifications;

- $V = \{v_1, v_2, \ldots, v_n\}$ be the set of n VM types to be allocated;
- $SV : V \rightarrow P$ be the function which indicates the VM type that server p_i is configured for ($1 \leq i \leq m, p_i \in P, SV(p_i) \in V$);
- $U = \{u_1, u_, \ldots, u_m\}$ be the list of VM capacities of physical servers where u_i is the maximum number of instances of VM Type $SV(p_i)$ that the physical server p_i can execute simultaneously considering the quality of service requirements, i.e., the number of slots in p_i ($1 \leq i \leq m, u_i \in \mathbb{Z}^+, p_i \in P$);
- $VS : V \rightarrow \mathcal{P}(P)$ be the function that returns the subset of the set of physical servers P that are configured for VM type v_d ($1 \leq d \leq n, v_d \in V,\ VS(v_d) \subseteq P$);
- C be the cost matrix where each element c^{ij} is the operating cost of the physical server p_i when the jth VM slot of the server p_i is occupied with a VM instance (note that the actual operating cost of the physical server p_i is the sum of the c^{ij} values which correspond to allocated VM slots on the server) ($c^{ij} \in \mathbb{R}^+$, $1 \leq i \leq m, 1 \leq j \leq u_i$));
- $B = \{B_1, B_2, \ldots, B_e\}$ be the set of user submitted bids such that each bid is defined as a pair $B_k = (R_k, o_k)$ where R_k is the set of subbids and o_k is the offered bid price;
- $R_k = \{\langle S_{k1}, q_{k1}\rangle, \langle S_{k2}, q_{k2}\rangle, \ldots\}$ be the set of subbids where $S_{kl} \subseteq V$ is the set of substitutable VM types and $q_{kl} \in \mathbb{Z}^+$ is the requested number of VMs from the set S_{kl} ($1 \leq k \leq e, 1 \leq l \leq f, f = max_{k,l}\,|S_{kl}|$).

A bid B_k is called *satisfiable* if all of its subbids are simultaneously satisfiable. A subbid $\langle S_{kl}, q_{kl}\rangle$, on the other hand, is called *satisfiable* if q_{kl} VMs can be allocated among the VM types listed in the set of substitutable VM types S_{kl}. Thus, there is a logical AND relationship among the subbids of a bid, and there is a logical OR relationship among the VM types listed in the set of substitutable VM types.

The winner determination problem (WDP) of the ECO-VMAP model is defined as finding the set $B' \in B$ of mutually satisfiable bids, the corresponding allocation of VMs, and the placement of these VMs to the physical servers while satisfying the required quality of service such that the difference between the sum of the bid prices of these satisfiable bids and the cost of the physical servers for the predetermined allocation period is maximized.

The WDP of the ECO-VMAP model can be formulated using linear integer programming. For this purpose, the following decision variables are defined:

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

$$y^{ij}_{kl} = \begin{cases} 1, & j^{th} \text{ VM slot of the physical server } p_i \text{ is allocated for the subbid } l \text{ of } R_k \\ 0, & \text{otherwise} \end{cases}$$

$$z^{ij} = \begin{cases} 1, & j^{th} \text{ VM slot of the physical server } p_i \text{ is allocated to a VM} \\ 0, & \text{otherwise} \end{cases}$$

Then, the WDP can be formulated as follows:

$$\max \sum_{B_k \in B} o_k x_k - \sum_{p_i \in P} \sum_{j=1}^{u_i} c^{ij} z^{ij} \tag{1}$$

$$\text{s.t} \sum_{B_k \in B} \sum_{S_{kl} \in R_k} y^{ij}_{kl} \leq z^{ij} \ (p_i \in P, 1 \leq j \leq u_i) \tag{2}$$

$$\sum_{v_d \in S_{kl}} \sum_{p_i \in VS(v_d)} \sum_{j=1}^{u_i} y^{ij}_{kl} = q_{kl} \cdot x_k \quad (B_k \in B, S_{kl} \in R_k) \tag{3}$$

$$z^{ij} \geq z^{i(j+1)} \ (p_i \in P, 1 \leq j \leq (u_i - 1)) \tag{4}$$

$$x_k, y^{ij}_{kl}, z^{ij} \in \{0, 1\} \ (\forall i, j, k, l) \tag{5}$$

In this formulation, (1) is the objective function that maximizes the difference between the sum of the prices of the satisfied bids and the cost of operating the physical servers on which the allocated VMs execute. Thus, the objective function maximizes the expected profit of the cloud provider based on the declared bid prices by the users. (2) ensures that each VM slot of a physical server can be allocated to a single subbid. (3) is the bid satisfaction constraint. For each satisfied bid, all of its subbids should be satisfied, that is, for each subbid of a satisfied bid, the requested number of VMs from the set of substitutable VM types should be allocated to the corresponding subbid. Finally, (4) defines the order in which VMs are placed in a physical server. For instance, a VM can be placed in the first slot of a currently idle server, and another VM, then, can be placed in the second VM slot of the physical server. This order is required to calculate the cost of operating the physical servers correctly.

Note that each element c_{ij} of the cost matrix C defines the *reservation price* of the j^{th} slot of the physical server p_i, that is, it denotes the minimum price that the cloud provider is willing to allocate the corresponding slot. This mechanism also enables the usage of the non-linear energy function of a physical server based on its utilization including the idle energy cost. For instance, the cost matrix C may be defined as:

$$c_{ij} = \begin{cases} \pi(v_d) + \varepsilon^i_{idle} + \varepsilon^i_{load}(\frac{1}{u_i}), & \text{for } j = 1 \\ \pi(v_d) + \left(\varepsilon^i_{load}(\frac{j}{u_i}) - \varepsilon^i_{load}(\frac{j-1}{u_i})\right), \\ \text{for } 1 < j \leq u_i \end{cases}$$

where v_d is the VM type that the physical server p_i is configured for ($v_d = SV(P_i)$), $\pi(v_d)$ denotes the minimum profit that the cloud provider is willing to earn by allocating one instance of v_d to a bidder (this value can be zero for non-commercial clouds), ε^i_{idle} denotes the idle energy cost of the physical server p_i during the allocation period, and $\varepsilon^i_{load}(x)$ is the energy cost the physical server p_i when its utilization level is x during the allocation period.

Finally, the WDP of the ECO-VMAP model is a generalization of the CAP problem of the combinatorial auction model [42] which is proven to be NP-hard. Therefore, the WDP of the ECO-VMAP model is also NP-hard.

TABLE 1. The features of the different-sized instances of the three VM types available from a major public cloud provider [70].

VM Type	Instance Size	# Virtual CPU	Memory (GiB)	Instance Storage (TB)	Disk Throughput (MiB/s)	Network Bandwidth (Gbps)
Compute Optimized (CO)	xlarge	2	4	Network Storage	N/A	Up to 10
	2xlarge	4	8	Network Storage	N/A	Up to 10
	4xlarge	8	16	Network Storage	N/A	Up to 10
	8xlarge	16	32	Network Storage	N/A	Up to 10
	18xlarge	36	72	Network Storage	N/A	10
	24xlarge	48	96	Network Storage	N/A	12
	36xlarge	72	144	Network Storage	N/A	25
	48xlarge	96	192	Network Storage	N/A	25
Memory Optimized (MO)	xlarge	2	16	Network Storage	N/A	up to 10
	2xlarge	4	32	Network Storage	N/A	up to 10
	4xlarge	8	64	Network Storage	N/A	up to 10
	8xlarge	16	128	Network Storage	N/A	up to 10
	16xlarge	32	256	Network Storage	N/A	10
	24xlarge	48	384	Network Storage	N/A	10
	32xlarge	64	512	Network Storage	N/A	20
	48xlarge	96	768	Network Storage	N/A	25
Storage Optimized (SO)	xlarge	4	16	2 x 14 HDD	500	Up to 25
	2xlarge	8	32	4 x 14 HDD	1,000	Up to 25
	4xlarge	16	64	8 x 14 HDD	2,000	25
	6xlarge	24	96	12 x 14 HDD	3,100	40
	8xlarge	32	128	16 x 14 HDD	4,100	50
	12xlarge	48	192	24 x 14 HDD	6,200	75

V. CONSOLIDATING DIFFERENT-SIZED VM INSTANCES ON A SINGLE PHYSICAL MACHINE - AN ALTERNATIVE FORMULATION FOR THE WDP OF THE ECO-VMAP MODEL

As introduced in Section III, the ECO-VMAP model assumes that each physical server is pre-configured for a VM type and can only host the instances of the corresponding VM type. For instance, consider the three of the VM types available from a well-known major public cloud provider that are listed in Table 1.

The *compute optimized* VM type (CO) has 8 instance sizes available, ranging from an instance with 2 virtual CPUs and 4 GiB of memory to an instance with 96 virtual CPUs and 192 GiB of memory. Similarly, the *memory optimized* VM type (MO) has also 8 instance sizes available. For this VM type, although the range of virtual CPUs is the same as the CO, the amount of memory reserved for the instances ranges from 16 to 768 GiB for memory-intensive workloads. Note that as stated in [70], the performance of a virtual CPU of the CO is higher than that of the MO. These two VM types can use network-based storage whereas the instances of the *storage optimized* VM type (SO) have local storage ranging from 28 TB to 336 TB.

In order to introduce these VM types in the ECO-VMAP model, each instance size of each VM type listed in Table 1 should be defined as a distinct VM type, that is, there would be 8+8+6=20 different VM types defined in the model (e.g., *CO.xlarge*, *CO.2xlarge*, and so on). Also, each physical server should be configured for executing one of these VM types. Thus, there should be at least one physical server for each of these 20 VM types in the cloud.

This assumption of the ECO-VMAP model prevents different-sized VM instances to be mapped to a single physical server although technically it is possible. For instance, an instance of *CO.xlarge* and an instance of *CO.2xlarge* cannot be mapped to a single physical server capable of running compute optimized VM types. This assumption of the ECO-VMAP model is not considered as a serious limitation, since the number of physical serves used by cloud providers far exceeds the number of VM types even when different-sized VM instance types are considered as separate VM types. For instance, the major public cloud provider mentioned above offers a total of 158 such VM types [70], whereas it is estimated that the provider has more than 1 million physical servers [71].

However, for smaller cloud systems, or for the cloud systems where this limitation causes an inefficient mapping, the ECO-VMAP model can be modified slightly so as to remove this limitation. In the modified (will be referred to as *alternative*) ECO-VMAP model, the bids should also include size information along with each VM type requested. For instance, if a bidder requests 3 VM instances of either CO or MO having 8 virtual CPUs, along with another VM instance of SO with 4 × 14 TB of local storage for a price of \$100, she would be submitting the following bid:

$$B_1 = [\{\langle (CO^{(4)}, MO^{(4)}), 3\rangle, \langle (SO^{(2)}), 1\rangle\}, \$100]$$

where the superscript of the VM type name indicates the size of the requested instance. This size value denotes the capacity of the requested instance as *a multiple of the smallest instance available for that VM type*. Thus, in this example, $CO^{(4)}$ refers to the instance *CO.4xlarge* which has four times the CPU and memory capacity of the smallest CO instance *CO.xlarge*.

Similarly, $MO^{(4)}$ and $SO^{(2)}$ refer to the VM types *MO.4xlarge* and *SO.2xlarge* in this example.

In accordance with the modified bidding language, let q_{kld} denote the size of the VM instance v_d requested in subbid l of R_k. Furthermore, a new integer decision variable t^i_{kld} is defined as the number of VM instances of type v_d that are allocated on the physical server p_i for the subbid l of R_k.

Then, the WDP of the alternative ECO-VMAP model can be formulated as follows:

$$\max \sum_{B_k \in B} o_k x_k - \sum_{p_i \in P} \sum_{j=1}^{u_i} c^{ij} z^{ij} \qquad (6)$$

$$\text{s.t} \sum_{B_k \in B} \sum_{S_{kl} \in R_k} y^{ij}_{kl} \leq z^{ij} \quad (p_i \in P, 1 \leq j \leq u_i) \qquad (7)$$

$$\sum_{v_d \in S_{kl}} \sum_{p_i \in VS(v_d)} t^i_{kld} = q_{kl} \cdot x_k \quad (B_k \in B, S_{kl} \in R_k) \qquad (8)$$

$$\sum_{j=1}^{u_i} y^{ij}_{kl} = q_{kld} \cdot t^i_{kld} \quad (B_k \in B, S_{kl} \in R_k, v_d \in S_{kl}, p_i \in P) \qquad (9)$$

$$z^{ij} \geq z^{i(j+1)} \; (p_i \in P, 1 \leq j \leq (u_i - 1)) \qquad (10)$$

$$x_k, y^{ij}_{kl}, z^{ij} \in \{0, 1\} \quad (\forall i, j, k, l) \qquad (11)$$

$$t^i_{kld} \in Z^+ \cup \{0\} \quad (\forall i, j, k, l) \qquad (12)$$

The objective function in (6), the constraints (7) and (10) are the same as the original formulation introduced in the previous section. As in the original formulation, the constraint in (8) functions as the bid satisfaction constraint, however, using the newly introduced variable t^i_{kld}. It ensures that for each satisfied bid, all of its subbids should be satisfied. It also ensures that every satisfied subbid gets exactly the number of VM instances requested in the corresponding subbid. For each VM allocated to a subbid, q_{kld} slots should be reserved in one of the physical servers that are configured for the VM type v_d, and this is enforced by the newly introduced constraint in (9). Note that in this formulation u_i denotes the maximum number of the smallest VM instance of type v_d that can be placed in the physical server p_i which is configured for the VM type v_d.

This alternative formulation will allow the same type of VM instances having different sizes to be mapped into a single physical server while ensuring that no over-utilization occurs in the physical server. Note that, although integer values are used for the size of the VM instances, this does not pose a restriction since any granularity can be obtained when the smallest instance is defined accordingly. For instance, a VM type using 3.7 virtual CPU can be defined if the smallest instance size is defined to have 0.1 virtual CPU. Then the size of the corresponding VM instance will be 37 which indicates that it has 37 times the resources required for the smallest VM instance with 0.1 virtual CPU. It is clear that the VM instance having two different types (any two of CO, MO and SO) should not be mapped to a single physical server, which is also ensured by the model.

VI. SOLUTION METHODS

Since the WDP of the ECO-VMAP model can be formulated using linear integer programming (see Section IV), the WDP can be solved to the optimality by using mixed-integer programming solvers. However, since this problem is NP-hard, finding the optimal solution, or even a feasible solution, may not be possible even for moderate problem instances. Therefore, the following heuristic methods are proposed for the WDP:

(i) Greedy Allocation and Placement Heuristic Method
(ii) Linear Relaxation-Based Heuristic Method
(iii) Network Flow-Based Heuristic Method
(iv) Partitioning-Based Heuristic Method

All the proposed heuristic methods process the bids in a given order, therefore, the bids submitted by the users should first be ordered before executing the heuristic methods. For this purpose, a sorting heuristic value is defined for each bid, and the bids are sorted in descending order based on these heuristic values of the bids.

The following four sorting heuristic functions are defined for determining the heuristic values of the bids:

Sorting Heuristic 1 The sorting heuristic value of a bid B_k is the *price* of the bid:

$$sh_1(B_k) = o_k$$

Sorting Heuristic 2 The sorting heuristic value of a bid B_k is the *mean profit* per virtual core requested:

$$sh_2(B_k) = \frac{o_k - \mu_c}{\mu_n}$$

where μ_c is the *mean* energy cost for all possible placements of VMs requested in the bid B_k to the available physical servers, μ_n is the mean number of cores required for the requested VMs in the bid B_k.

Sorting Heuristic 3 The sorting heuristic value of a bid B_k is the *maximum profit* per virtual core requested:

$$sh_3(B_k) = \frac{o_k - \eta_c}{\eta_n}$$

where η_c is the *minimum* energy cost that is required to place all the VMs requested in the bid B_k to the available physical servers, and η_n is the *minimum* number of cores required for the requested VMs in the bid B_k.

Sorting Heuristic 4 The sorting heuristic value of a bid B_k is the value of the decision variable x_k obtained after solving the linear relaxation of the integer program given in Section IV:

$$sh_4(B_k) = x_k$$

After sorting the bids in descending order based on sorting heuristic values, the actual heuristic methods can be executed.

A. GREEDY ALLOCATION AND PLACEMENT HEURISTIC METHOD

The Greedy Allocation and Placement Heuristic (GAPH) method is proposed to find a good solution to the WDP of the ECO-VMAP model in a short time. The pseudocode for GAPH method can be seen in Algorithm 1.

In this method, the bids which are sorted in descending order of sorting heuristic values are processed one by one. When a bid B_k is processed, the method tries to allocate the requested VMs such that when they are placed in the available slots of the physical servers, the total cost is the minimum among all possible placements. For this purpose, the GAPH method executes the uniform cost search on the available slots of the physical servers using a priority queue data structure (*minHeap*) for selecting the minimum cost slots suitable for the requested VMs. For more information about the uniform cost search, see [72].

If all of the subbids of the bid B_k can be satisfied, i.e., all the requested VMs can be allocated, then, the total cost of the corresponding allocation is calculated. If the bid price o_k exceeds the total cost of the allocation for the bid, then the slots are allocated for the bid B_k, the bid B_k is marked as satisfied, and it is added to the winning bid set B_{sol}. If there are not enough VM slots, or the cost of the allocation is higher than the bid price o_k, then the bid B_k is rejected, and temporarily allocated slots for the bid B_k are deallocated. The method works in a greedy manner such that when a bid is satisfied, the corresponding allocation is fixed. The method then continues with the next bid in the sorted bid set B. The method terminates when all the bids in the sorted bid set B are processed.

B. LINEAR RELAXATION-BASED HEURISTIC METHOD

The Linear Relaxation-Based Heuristic (LRBH) method presented in Algorithm 2 consists of two phases. In the first phase, the method builds a winning bid set B_{sol} consisting of mutually satisfiable bids, starting from scratch. It starts with the first bid in the sorted bid set B, and adds the bids to the solution set B_{sol} one at a time. When a bid is added to the set B_{sol}, the method checks whether the required VMs in the set B_{sol} can be allocated simultaneously. If so, the method keeps the recently added bid in the set B_{sol}, and moves to the next bid in the sorted bid set B. Otherwise, it removes the recently added bid from the set B_{sol}, and again continues with the next bid. To check whether the bids in the set B_{sol} are mutually satisfiable or not, the following linear program is solved:

$$\max \sum_{B_k \in B_{sol}} o_k - \sum_{p_i \in P} \sum_{j=1}^{u_i} c^{ij} z^{ij} \tag{13}$$

$$\text{s.t.} \sum_{B_k \in B_{sol}} \sum_{S_{kl} \in R_k} y_{kl}^{ij} \leq z^{ij} \quad (1 \leq i \leq m, 1 \leq j \leq u_i) \tag{14}$$

$$\sum_{v_d \in S_{kl}} \sum_{p_i \in VS(v_d)} \sum_{j=1}^{u_i} y_{kl}^{ij} = q_{kl} \quad (B_k \in B_{sol}, S_{kl} \in R_k) \tag{15}$$

Algorithm 1: Pseudocode for the Greedy Allocation and Placement Heuristic Method

```
ECO-VMAP
Require: problem instance, sorting
    heuristic function $sh(x)$.
Ensure: Winning bid set $B_{sol}$ including the allocation of
    VMs and their placement to the slots of the physical
    servers.
 1: Sort the bids in the bid set $B$ according to the sorting
    heuristic function value $sh(x)$ in descending order;
 2: Set the winning bids set $B_{sol} \leftarrow \emptyset$;
 3: Set the bid index $k \leftarrow 1$;
 4: while $k \leq |B|$ do
 5:   Set the subbid index $l \leftarrow 1$;
 6:   while $l \leq |R_k|$ do
 7:     Initialize minHeap as a binary heap
        data structure with the smallest key values at the
        root;
 8:     for all VM Type $v$ in $S_{kl}$ do
 9:       for all physical server $p_i$ which is
          configured for the VM Type $v$ ($p_i \in VS(v)$) do
10:         Insert the first available slot $j$ of $p_i$ to
            minHeap where the key value is the $c^{ij}$ (the
            cost of the corresponding slot);
11:       end for
12:     end for
13:     while the number of allocated VMs for the
        subbid $\langle S_{kl}, q_{kl} \rangle < q_{kl}$ and minHeap is
        not empty do
14:       Extract the slots with the minimum cost from
          minHeap and allocate to the subbid $\langle S_{kl}, q_{kl} \rangle$;
15:     end while
16:     if the number of allocated VMs for subbid
        $\langle S_{kl}, q_{kl} \rangle < q_{kl}$ then
17:       if l = 1 then
18:         Deallocate the VMs for the bid $B_k$
19:         Set the bid index $k \leftarrow k + 1$;
20:         continue
21:       end if
22:       Backtrack to the previous subbid ($l \leftarrow l - 1$)
          and change the allocation of VM slots to the
          next available slot in the corresponding
          minHeap data structure;
23:     end if
24:     Set the bid index $l \leftarrow l + 1$;
25:   end while
26:   if all the subbids in the bid $B_k$ are satisfied and the
      total cost of the allocated slots for the bid
      $B_k$ is lower than the bid price $o_k$ then
27:     Add the bid $B_k$ to the winning bid set $B_{sol}$
28:   end if
29:   Set the bid index $k \leftarrow k + 1$;
30: end while
31: return The winning bid set $B_{sol}$ including the
    allocation and placement information;
```

$$z^{ij} \geq z^{i(j+1)} \quad (p_i \in P, 1 \leq j \leq (u_i - 1)) \tag{16}$$
$$0 \leq y_{kl}^{ij}, z^{ij} \leq 1 \quad (\forall i, j, k, l) \tag{17}$$
$$y_{kl}^{ij}, z^{ij} \in \mathbb{R} \quad (\forall i, j, k, l) \tag{18}$$

After all the bids are traversed, the maximal set of mutually satisfiable bids B_{sol} are found, however, the allocation found in this step may not be feasible because of relaxing the integrality constraints. Therefore, in the second phase of the LRBH method, the integral version of the linear program defined above is solved only for the final winning bid set B_{sol} to find the allocation and placement of VMs for this set. That is, the linear integer program in (13)-(16), and additionally with the following integrality constraint is solved:

$$y_{kl}^{ij}, z^{ij} \in \{0, 1\} \quad (\forall i, j, k, l) \tag{19}$$

C. NETWORK FLOW-BASED HEURISTIC METHOD

The LRBH method presented in Algorithm 2 solves the linear relaxation of the linear integer program defined in (1)-(5) to find a maximal mutually satisfiable set of bids in the first phase and also solves another linear integer program to find an allocation VMs and their placements to the physical servers in the second phase. Although solving the linear relaxation in the first phase is relatively fast, the second phase may require exponential time in the size of the winning bid set B_{sol}.

To overcome this issue, in the Network Flow-Based Heuristic (NFBH) method presented in Algorithm 3, the requirement of filling the VM slots of physical servers in order is relaxed. That is, the constraint in (4) is removed. Since the VM slots can be filled in any order, the cost of each slot of a physical server is also redefined as the mean cost value of all the slots in that server. That is, the new cost value $\bar{c}^{ij}$ of the jth VM slot of the physical server p_i is defined as:

$$\bar{c}^{ij} = \frac{\sum_{j=1}^{u_i} c^{ij}}{u_i} \tag{20}$$

Similar to the LRBH method, the NFBH method has also two phases. In the first phase, it builds a winning bid set B_{sol}, starting with an empty set. Then, the method adds each bid in the sorted bid set B to B_{sol} one at a time, and checks whether the bids in the set B_{sol} are mutually satisfiable or not, as in the LRBH method. However, as noted above, instead of the linear program defined in (1)-(5), the following program is solved:

$$\max \sum_{B_k \in B_{sol}} o_k - \sum_{p_i \in P} \sum_{j=1}^{u_i} \bar{c}^{ij} z^{ij} \tag{21}$$
$$\text{s.t.} \sum_{B_k \in B_{sol}} \sum_{S_{kl} \in R_k} y_{kl}^{ij} \leq z^{ij} \quad (1 \leq i \leq m, 1 \leq j \leq u_i) \tag{22}$$
$$\sum_{v_d \in S_{kl}} \sum_{p_i \in VS(v_d)} \sum_{j=1}^{u_i} y_{kl}^{ij} = q_{kl} \quad (B_k \in B_{sol}, S_{kl} \in R_k) \tag{23}$$
$$0 \leq y_{kl}^{ij}, z^{ij} \leq 1 \quad (\forall i, j, k, l) \tag{24}$$
$$y_{kl}^{ij}, z^{ij} \in \mathbb{R} \quad (\forall i, j, k, l) \tag{25}$$

Algorithm 2: Pseudocode for the Linear Relaxation-Based Heuristic Method

```
ECO-VMAP
Require: problem instance, sorting heuristic function $sh(x)$.
Ensure: The winning bid set $B_{sol}$ including the allocation
    of VMs and their placement to the slots of the physical
    servers.
 1: Sort the bids in the bid set $B$ according to the sorting
    heuristic function value $sh(x)$ in descending order;
 2: Set the winning bids $B_{sol} \leftarrow \emptyset$;
 3: Set $objVal \leftarrow 0$;
 4: Set the bid index $k \leftarrow 1$;
 5: Set $availableSlots \leftarrow$ the total number slots in all
    physical servers;
 6: while $k \leq |B|$ do
      {The First Phase: Find the mutually satisfiable set of
      bids $B_{sol}$.}
 7:   if the number of requested VMs in $B_k <$
      $availableSlots$ then
 8:     Add $B_k$ to $B_{sol}$;
 9:     Solve the linear program defined in (13)-(18) for
        the bid set $b_{sol}$ using the simplex algorithm;
10:     Set $newObjVal \leftarrow$ the objective value of the
        relaxed linear integer program;
11:     if the solution is feasible and $objVal < newObjVal$
        then
12:       Set $objVal \leftarrow newObjVal$;
13:       Set $availableSlots \leftarrow availableSlots-$ the
          number of allocated slots for $B_k$;
14:     else
15:       Remove the bid $B_k$ from the solution set $B_{sol}$;
16:     end if
17:   end if
18:   Set the bid index $k \leftarrow k + 1$;
19: end while
20: {The Second Phase: Find the allocation of VMs and
    their placement for the winning bid set $B_{sol}$.}
21: Optimize the linear integer program defined in (13)-(16)
    and (19);
22: return The winning bid set $B_{sol}$ including the
    allocation and placement information;
```

The main difference between this program and the linear program used in the LRBH method is that this program has a network structure (the constraint matrix is totally unimodular) and can be solved as a Minimum Cost Network Flow Problem [73]. This kind of problem instances can be solved faster than the general linear programs using the linear program solvers that exploit the network structure or using the specialized network flow algorithms such as Goldberg and Tarjan's algorithm [74].

Different from the LRBH method, in the NFBH method when the mutually satisfiable winning bid set B_{sol} is determined, the corresponding allocation of VMs is also found.

This difference comes from the fact that since the constraint matrix is totally unimodular, the decision variables y and z take integer values resulting in a feasible allocation of VMs. However, since the new cost values $\overline{c}^{ij}$ do not reflect the real cost, the placement of VMs to the physical servers may further be improved to reduce the cost.

For this purpose, in the second phase of the method, the allocated VMs are consolidated to reduce the costs. Since an allocation and the corresponding placement is found after the first phase, the total number of allocated slots in the physical servers that execute a VM type v can also be found, which is

$$s_v = \sum_{p_i \in VS(v)} \sum_{j=1}^{u_i} z^{ij} \tag{26}$$

These allocated s_v VMs are placed to the physical servers in the set $VS(v)$, starting from the one with the smallest mean cost value, $\overline{c}^{ij}$, and filling all its slots. Then, the next available physical server with the smallest mean cost value is filled with VMs. This process continues until all the VMs are consolidated in the slots of physical servers. Note that the second phase of the NFBH method ensures that for each VM type v, there can be at most one physical server that is neither fully utilized nor idle, and each remaining physical servers in the set $VS(v)$ is either fully utilized or idle, i.e., turned off.

D. PARTITIONING-BASED HEURISTIC METHOD

Since the WDP of the ECO-VMAP model is NP-hard, its worst-case time complexity increases exponentially based on the number of bids in the problem instance. Thus, it may not be possible to solve even moderate size problem instances.

In the Partitioning-Based Heuristic (PBH) method presented in Algorithm 4, to reduce the time complexity, the bid set B is divided into fixed-size partitions (i.e., subsets), and a linear integer program is solved for each partition. The size of the partitions *parSize* is an input parameter of the method. As in the previous heuristic methods, initially, the bids are sorted according to the sorting heuristic function in descending order. Secondly, all the bids in the set B are disabled. Disabling a bid B_k means that the upper limits of all the decision variables linked with the bid B_k, that is $x_k, y_{kl}^{ij} (\forall l, i, j)$, are set to 0. After that, the iteration phase begins. The bids in the first partition, i.e., the first *parSize* bids in the sorted bid set B, are enabled. Enabling a bid B_k is the opposite of the disabling operation, that is the upper limits of all the decision variables linked with the bid B_k, that is $x_k, y_{kl}^{ij} (\forall l, i, j)$, are set to 1. Note that disabling a bid fixes the values of the corresponding decision variables to 0, whereas, enabling a bid does not fix the corresponding decision variables to 1. The reason is that the lower limits of the binary decision variables are not changed in both operations which are set to 0.

After the bids are enabled in the current partition, the linear integer program for the WDP of the ECO-VMAP model defined in (1)-(5) is solved. The satisfied bids in the optimal solution is added to the winning bid set B_{sol}. Also, for all these satisfied bids, the corresponding allocation of VMs and their

Algorithm 3: Pseudocode for the Network Flow-Based Heuristic Method

```
ECO-VMAP
Require: problem instance, sorting heuristic function $sh(x)$.
Ensure: The winning bid set $B_{sol}$ including the allocation
        of VMs and their placement to the slots of the physical
        servers.
 1: Sort the bids in the bid set $B$ according to the sorting
    heuristic function value $sh(x)$ in descending order;
 2: Set the winning bids $B_{sol} \leftarrow \emptyset$;
 3: Set $objVal \leftarrow 0$;
 4: Set the bid index $k \leftarrow 1$;
 5: Set $availableSlots \leftarrow$ the total number slots in all
    physical servers;
 6: while $k \leq |B|$ do
      {The First Phase: Find the mutually satisfiable set of
      bids $B_{sol}$ and the allocation and placement of VMs.}
 7:   if the number of VMs requested in $B_k <$
      $availableSlots$ then
 8:     Add $B_k$ to $B_{sol}$;
 9:     Solve the network model defined in (21)-(24) for
        the bid set $b_{sol}$;
10:     Set $newObjVal \leftarrow$ the objective value of the
        network model;
11:     if the solution is feasible and $objVal < newObjVal$
        then
12:       Set $objVal \leftarrow newObjVal$;
13:       Set $availableSlots \leftarrow availableSlots -$ the
          number of allocated slots for $B_k$;
14:     else
15:       Remove the bid $B_k$ from the solution set $B_{sol}$;
16:     end if
17:   end if
18:   Set the bid index $k \leftarrow k + 1$;
19: end while
20: {The Second Phase: Improve the placement of VMs by
    consolidation.}
21: for all VM Type $v$ in $V$ do
22:   Set $s_v \leftarrow \sum_{p_i \in VS(v)} \sum_{j=1}^{u_i} z^{ij}$ {$z^{ij}$ values are calculated in
      the first phase}
23:   Set the set $availableSrv_v \leftarrow VS(v)$;
24:   Sort the set $availableSrv_v$ according to the mean cost
      per slot in ascending order;
25:   for all physical server $p_i \in availableSrv_v$ do
26:     if $s_v > u_i$ then
27:       Allocate all the $u_i$ slots on ther server $p_i$;
28:       Set $s_v \leftarrow s_v - u_i$
29:     else
30:       Allocate all the $s_v$ slots on ther server $p_i$;
31:       break
32:     end if
33:   end for
34: end for
35: return The winning bid set $B_{sol}$ including the
    allocation and placement information;
```

placement are fixed, that is the values of the decision variables linked with the satisfied bid B_k are fixed to their value in the optimum solution. Fixing a decision variable means that both upper and lower values of it are set to its current value. Then, the remaining unsatisfied bids are disabled again. This completes the processing of the first partition. The remaining partitions are also processed in the same way iteratively allocating available VM slots for the remaining subsets of bids.

Despite the worst-case running time complexity of the WDP of the ECO-VMAP model being exponential, this method runs in polynomial time when a constant partition size is defined. Also, a time limit can be employed for processing each partition if larger partition sizes are used. In this study, a partition size of 25 is used.

Note that the heuristic methods that use mathematical programming, i.e., LRBH, NFBH, and PBH, are explained with respect to the original formulation of the WDP provided in Section IV, in order to use these heuristics with the alternative formulation of the WDP introduced in Section V, program in (6)-(12) should be considered instead of program in (1)-(5).

Algorithm 4: Pseudocode for the Partitioning-Based Heuristic Method

```
ECO-VMAP
Require: problem instance, sorting heuristic function $sh(x)$,
    partition size $parSize$.
Ensure: The winning bid set $B_{sol}$ including the allocation
    of VMs and their placement to the slots of the physical
    servers.
 1: Sort the bids in the bid set $B$ according to the sorting
    heuristic function value $sh(x)$ in descending order;
 2: Set the winning bids $B_{sol} \leftarrow \emptyset$;
 3: Set $numOfPartitions \leftarrow \left\lceil \frac{|B|}{parSize} \right\rceil$;
 4: for all bid $B_k$ in $B$ do
 5:   Disable the bid $B_k$;
 6: end for
 7: for $partitionNo \leftarrow 1$ to $numOfPartitions$ do
 8:   Set $startIndex \leftarrow (partitionNo - 1) * parSize$;
 9:   Set $endIndex \leftarrow startIndex + parSize - 1$;
10:   for $k \leftarrow startIndex$ to $endIndex$ do
11:     Enable the bid $B_k$;
12:   end for
13:   Solve the linear integer program defined in (1)-(5)
14:   for $k \leftarrow startIndex$ to $endIndex$ do
15:     if $x_k = 1$ then
          {If the bid $B_k$ is satisfied}
16:       Fix the values of the decision variables $x, y, z$
          for the bid $B_k$; {Fix the allocated VMs and the
          corresponding slots for the bid $B_k$}
17:       Add $B_k$ to $B_{sol}$;
18:     else
19:       Disable the bid $B_k$;
20:     end if
21:   end for
22: end for
23: return The winning bid set $B_{sol}$ including the
    allocation and placement information;
.
```

VII. EXPERIMENTAL RESULTS

To evaluate the performance of the ECO-VMAP model and the proposed solution methods, a test case generator suitable for factorial testing has been developed to generate problem instances. The parameters used in the test suite are as follows:

(i) **The Number of Physical CPU Cores** defines the sum of the physical CPU cores of all the physical servers in the cloud. This parameter takes 4 different values between 2592 and 10368.

(ii) **The Bid Density** is the ratio of all virtual cores requested inside all the bids to the number of physical CPU cores. A bid density value less than 1 indicates that the number of physical CPU cores is greater than the requested virtual cores, that is the supply exceeds the demand. A bid density value greater than 1 indicates the opposite, the demand exceeds the supply. This parameter takes 8 different values between 0.25 and 5.

(iii) **The Number of Data Centers** indicates the number of data centers in the cloud where the physical servers are located. This parameter takes values of 1, 2, and 3.

(iv) **The Average Number of Subbids in a Bid** is the mean of the number of subbids in every bid generated. This parameter takes values of 1, 2, and 3.

(v) **The Average Requested Number of VMs in a Subbid** Indicates the mean of the requested number of VMs in every subbid generated. This parameter takes 7 different values between 2 and 8.

Note that the parameter (i) also indirectly defines the number of physical servers and their slots available in the cloud, and the parameters (i) and (ii) together determine the number of bids generated. The remaining parameters (iii)-(v) determine the complexity of the generated bids.

Using this generator, a comprehensive test suite consisting of 4032 ECO-VMAP model instances was prepared for the following experiments. To evaluate the performances of the proposed heuristic methods, these instances were solved using Gurobi Optimizer version 8 [75] on a Linux workstation with two 8-cores 3.1 GHz Intel Xeon processors and 128 GB memory. For each instance, a wall-clock time limit of 1 hour was set. Note that large problem instances were not included in the generated test suite to be able to find the optimal solutions. Among the 4032 test instances, 3985 instances could be solved to the optimality within the given time limit. The remaining 47 test instances were not used in the experiments to avoid a possible bias.

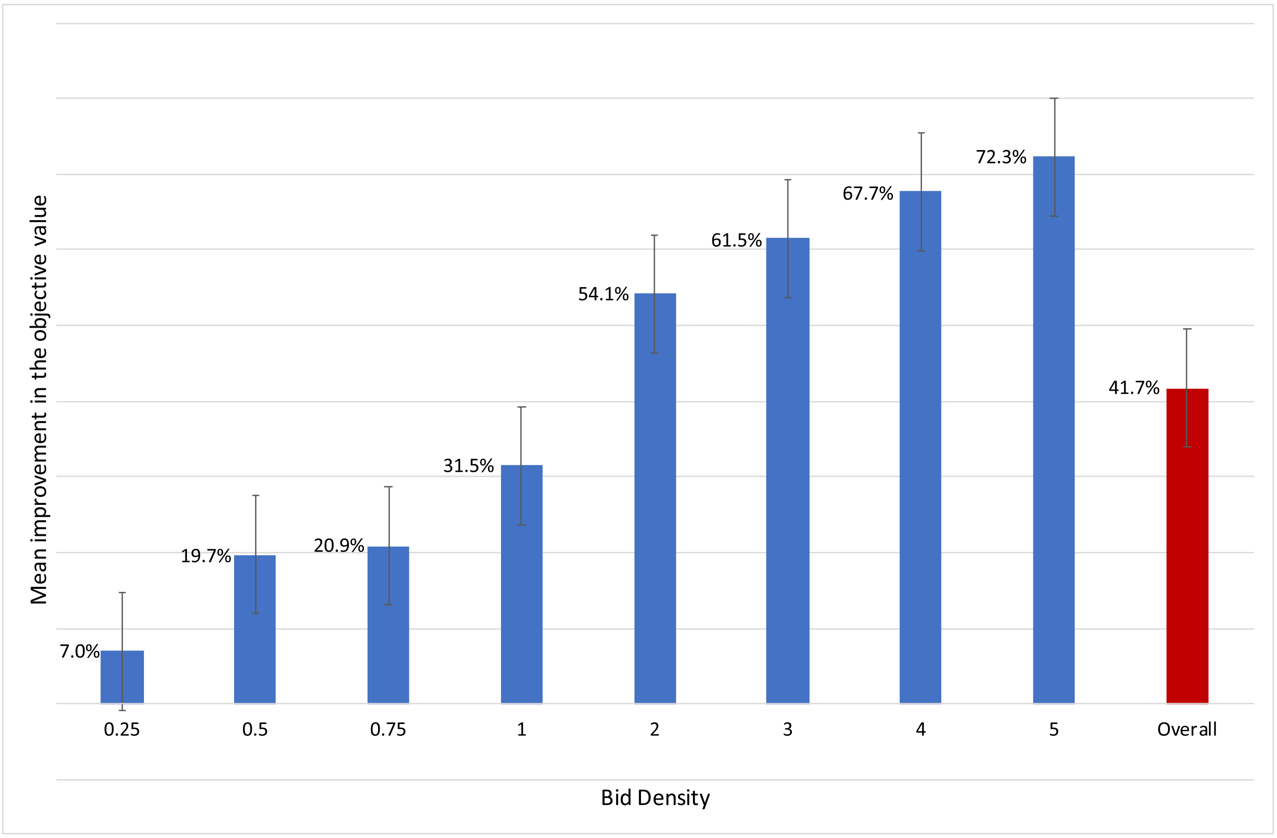


FIGURE 4. Mean improvement in the objective value using the ECO-VMAP model over the FCFS-based simulation grouped into the bid density values. Horizontal bars indicate the standard error of the corresponding sample.

A. ESTIMATING THE IMPROVEMENT TO BE OBTAINED USING THE ECO-VMAP MODEL

The features and the benefits of the ECO-VMAP model were introduced in the previous sections qualitatively. To estimate the benefit of the ECO-VMAP model qualitatively in terms of the objective function with respect to the widely used first-come, first-served (FCFS) based system, a simulation experiment was conducted. For each test instance, the simulation method presented in Algorithm 5 was executed to simulate the FCFS-based system.

The simulation method was executed 100 times for each test instance, and the mean objective value was calculated. These values are compared to the results of the ECO-VMAP model. The results of this experiment which are grouped into the bid density values can be seen in Figure 4. It is seen that processing the requests of the users as they arrive at the system, that is allocating the VMs requested by the users and placing them to the least cost slots available at that time, cannot provide an efficient allocation compared to the ECO-VMAP model. The ECO-VMAP model provides approximately 7% to 72% better allocations of VMs and their placements in terms of the objective value with respect to FCFS based systems depending on the bid density, i.e., the

Algorithm 5: Pseudocode for the Simulation Method for the FCFS Based System

```
ECO-VMAP
Require: problem instance, the number of simulations to
      be conducted $numSimul$.
Ensure: Mean objective value $meanSimulObjVal$ found
      after the simulation.
1: Set $meanSimulObjVal \leftarrow 0$;
2: for $simulNo \leftarrow 1$ to $numSimul$ do
3:     Shuffle the bids in the bid set $B$; {To simulate the
       random arrival order of bids.}
4:     Set $B_{sol} \leftarrow$ GAPH($B$); {Try to allocate VMs for
       each bid iteratively in the shuffled bid set $B$, and
       place them to the slots of the available servers such
       that the cost is the minimum using the GAPH
       method.}
5:     Set $meanSimulObjVal \leftarrow meanSimulObjVal +$ the
       objective value of the solution set $B_{sol}$;
6: end for
7: Set
   $meanSimulObjVal \leftarrow meanSimulObjVal / numSimul$;
8: return The mean objective value $meanSimulObjVal$;
```

TABLE 2. Mean and standard deviation of the success rates of the proposed heuristic methods using different sorting heuristics for each bid density value.

Heuristic	Bid Density	Sorting Heuristic 1		Sorting Heuristic 2		Sorting Heuristic 3		Sorting Heuristic 4	
		mean	stdev	mean	stdev	mean	stdev	mean	stdev
GAPH	0.25	95.4%	5.8%	95.3%	5.9%	95.3%	5.9%	94.8%	7.1%
	0.5	93.0%	9.1%	92.7%	8.9%	92.7%	9.1%	91.8%	10.5%
	0.75	90.2%	10.1%	89.8%	10.3%	89.6%	10.4%	89.2%	10.9%
	1	87.3%	9.6%	86.7%	10.0%	86.8%	9.7%	86.3%	10.3%
	2	81.3%	8.0%	83.5%	8.2%	83.3%	8.5%	83.4%	9.5%
	3	80.7%	8.0%	83.8%	7.6%	83.7%	7.6%	83.8%	9.0%
	4	80.7%	7.6%	83.7%	7.1%	83.7%	7.1%	84.6%	9.1%
	5	80.5%	7.3%	83.5%	7.0%	83.5%	6.8%	85.1%	8.8%
	Overall	**86.1%**	**10.1%**	**87.3%**	**9.4%**	**87.3%**	**9.4%**	**87.3%**	**10.2%**
LRBH	0.25	99.7%	0.3%	99.7%	0.8%	99.6%	0.9%	99.4%	2.7%
	0.5	99.7%	1.1%	99.4%	2.3%	99.4%	2.2%	99.5%	2.0%
	0.75	99.5%	1.5%	99.1%	3.8%	99.1%	3.8%	99.3%	3.5%
	1	98.5%	2.6%	98.6%	4.3%	98.8%	2.8%	99.1%	2.4%
	2	92.6%	5.4%	95.9%	3.3%	96.1%	3.2%	99.0%	2.2%
	3	90.4%	5.7%	94.4%	3.5%	94.3%	3.5%	98.8%	2.4%
	4	89.6%	5.8%	93.2%	4.0%	93.2%	4.0%	98.8%	2.1%
	5	88.9%	5.8%	92.2%	4.7%	92.3%	4.4%	98.7%	2.3%
	Overall	**94.9%**	**6.2%**	**96.6%**	**4.5%**	**96.6%**	**4.3%**	**99.1%**	**2.5%**
NFBH	0.25	97.0%	2.9%	96.5%	4.8%	96.4%	4.8%	95.4%	8.4%
	0.5	97.9%	2.3%	97.3%	4.1%	97.3%	4.0%	95.7%	10.1%
	0.75	98.5%	2.1%	97.4%	4.9%	97.3%	5.5%	96.9%	7.0%
	1	97.7%	2.8%	97.5%	4.6%	97.6%	3.4%	97.6%	3.7%
	2	92.3%	5.4%	95.4%	3.4%	95.6%	3.3%	98.3%	3.0%
	3	90.2%	5.6%	94.1%	3.6%	94.0%	3.6%	98.4%	3.1%
	4	89.5%	5.7%	92.9%	3.9%	92.9%	3.9%	98.6%	2.5%
	5	88.8%	5.6%	92.1%	4.3%	92.1%	4.4%	98.5%	2.5%
	Overall	**94.0%**	**5.8%**	**95.4%**	**4.7%**	**95.4%**	**4.6%**	**97.4%**	**5.9%**
PBH	0.25	99.6%	0.3%	99.6%	0.3%	99.6%	0.3%	99.6%	0.3%
	0.5	99.6%	0.4%	99.6%	0.5%	99.6%	0.5%	99.6%	0.6%
	0.75	99.4%	0.9%	99.5%	0.9%	99.5%	0.9%	99.4%	1.0%
	1	98.9%	1.6%	98.8%	1.6%	98.9%	1.6%	98.9%	1.6%
	2	94.9%	5.0%	96.7%	3.0%	96.8%	2.8%	97.9%	2.4%
	3	92.8%	5.2%	95.3%	3.5%	95.4%	3.5%	97.9%	2.2%
	4	91.3%	5.8%	94.4%	4.0%	94.5%	4.0%	97.8%	2.3%
	5	90.6%	5.6%	93.9%	4.2%	93.9%	4.3%	97.9%	2.1%
	Overall	**95.9%**	**5.4%**	**97.2%**	**3.5%**	**97.3%**	**3.5%**	**98.6%**	**1.9%**

supply-demand ratio. The mean improvement observed in all test cases is observed as approximately 42%.

B. PERFORMANCES OF THE PROPOSED HEURISTIC METHODS

To measure the performances of the proposed heuristic methods, all the 3985 test instances whose optimal solutions were found, further solved with the proposed four heuristic methods. For each heuristic method, again the proposed four sorting heuristic functions are used. Therefore, each test instance is solved for a total of 16 times for each heuristic method - sorting heuristic function pair. The objective values found by the heuristic methods are compared to the optimal objective values found by the MIP solver using the *success rate* measure which is defined as the ratio of the objective value found by the heuristic method to the optimal solution.

The means and standard deviations of the success rates of the solutions found by the proposed heuristic methods are presented in Table 2, and the box plot for the success rates can be seen in Figure 5.

To analyze whether the differences in the mean success rates of these 16 heuristic configurations are significant or not, the two-way analysis of variance (ANOVA) test is conducted at an $\alpha = 0.05$ significance level. The success rate is used as the dependent variable, and the heuristic method and the sorting heuristic function are used as two independent variables. The results indicate that there is a statistically significant interaction for the success rate between the heuristic method and the sorting heuristic function, $F(9, 63744) = 33.481, p < 0.0005$, partial $\eta^2 = 0.005$.

Because of the significant interaction effect, the simple main effects are analyzed [76]. The pairwise comparison results can be seen in Table 3 and Table 4. In the remaining text, the term *significant* used for a mean difference means that it is statistically significant at the $\alpha = 0.05$ significance level.

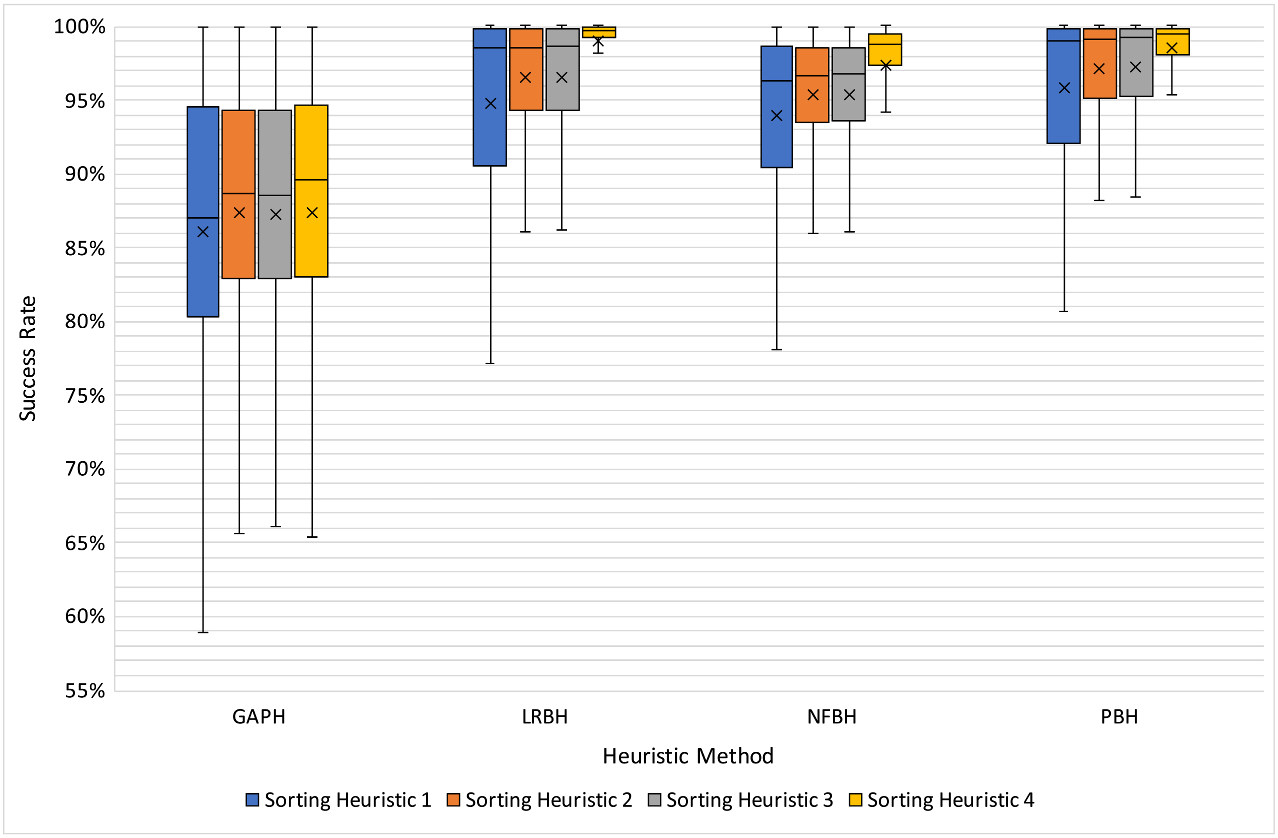


FIGURE 5. Box plot for the success rates of the proposed heuristic methods using different sorting heuristics for all test instances. (x) indicates the mean success rate of the corresponding heuristic method.

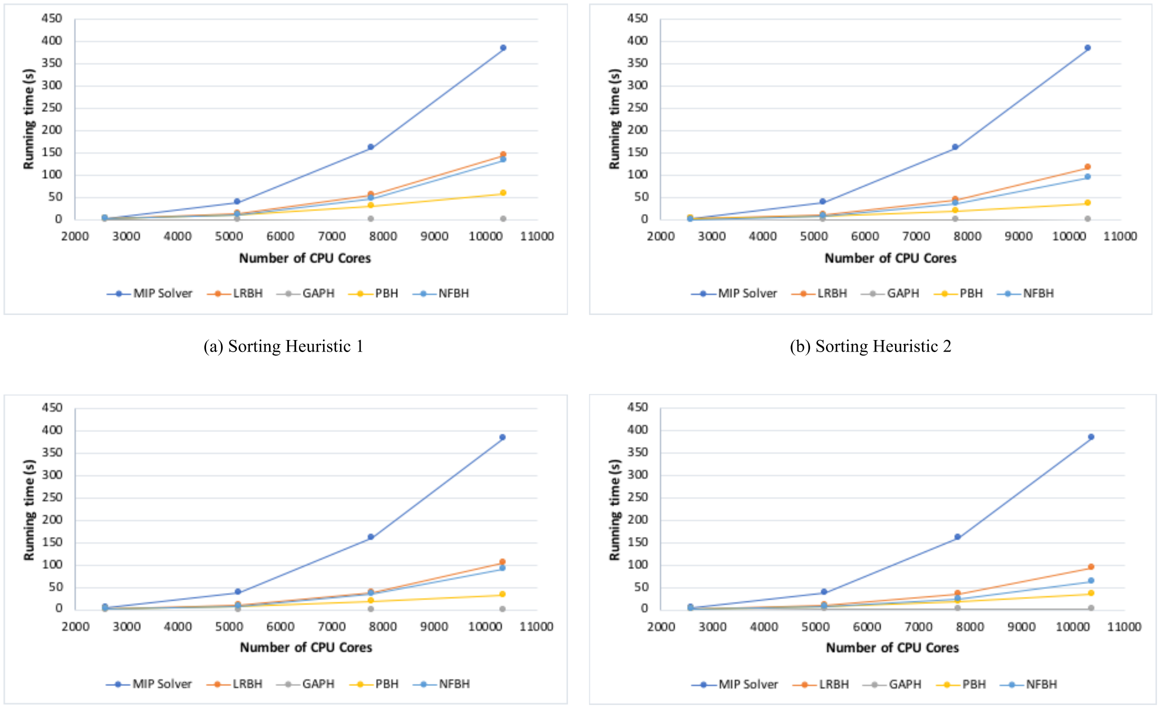
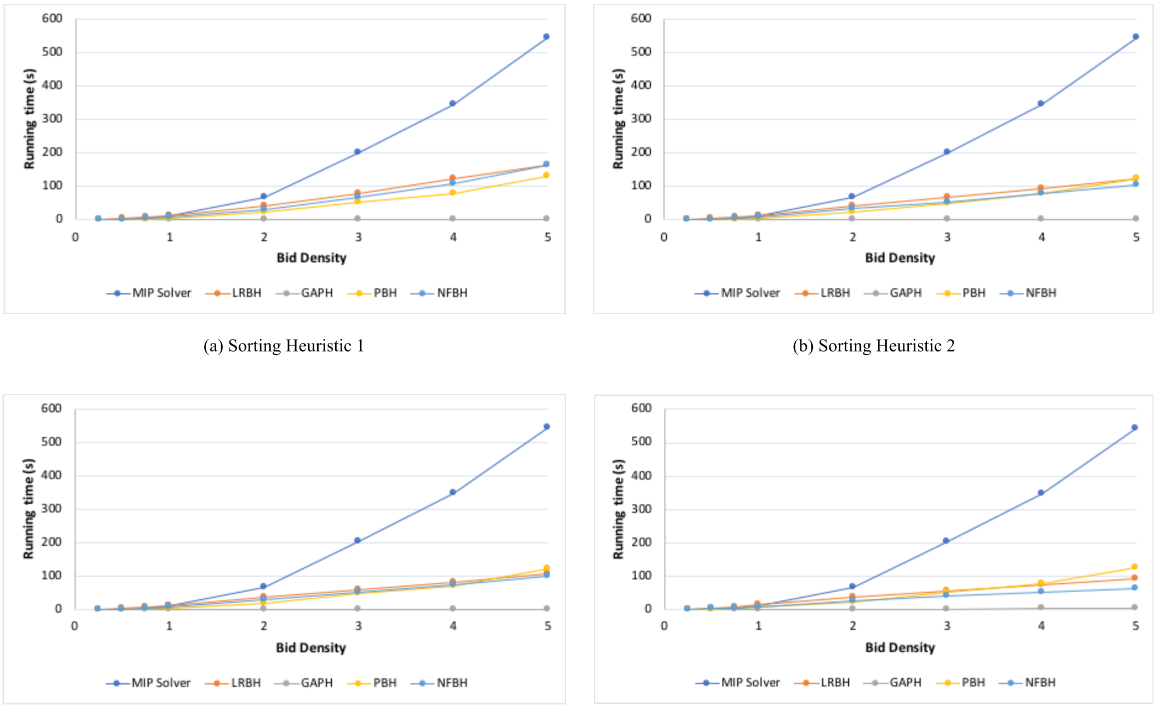


FIGURE 6. Mean running times of the MIP solver (optimal solutions) and the proposed heuristic methods versus the number of CPU cores for all test instances.



(a) Sorting Heuristic 1

(b) Sorting Heuristic 2

(c) Sorting Heuristic 3

(d) Sorting Heuristic 4

FIGURE 7. Mean running times of the MIP solver (optimal solutions) and the proposed heuristic methods versus the bid density value for all test instances.

Being the simplest but the fastest greedy heuristic, the GAPH method provides the lowest success rates among all the heuristic methods with mean success values ranging between 86.1% and 87.3% depending on the sorting heuristic function used. The NFBH method provides solutions that are significantly better than the GAPH method on average for all sorting heuristics. The range for overall mean values is between 94.0% and 97.4%. The PBH and LRBH methods are the best performing methods for the prepared test suite. However, neither is dominant to the other one for all sorting heuristics. The PBH method produces significantly better solutions with mean success rates of 95.9%, 97.2%, and 97.3% than the LRBH method which produces mean success rates of 94.9%, 96.6%, and 96.6% for the Sorting Heuristics 1, 2, and 3, respectively. The LRBH method, on the other hand, produces the significantly best solutions among all the heuristics when the Sorting Heuristic 4 is used with a mean success rate of 99.1% and a standard deviation of 2.5%.

Table 2 also presents the mean success rates for each bid density value used in the test suite. For Sorting Heuristics 1, 2, and 3, the mean success rates of all the heuristics tend to decrease as the bid density increases. Note that the bid density determines the number of bids in the test instances, meaning that a test instance with a high bid density contains more bids than a test instance with a lower bid density. For the Sorting Heuristic 4, on the other hand, the mean success rates of the LRBH and PBH methods again decreases as the bid density increases. However, the amount of the decrease is considerably small, approximately only 0.8% for the LRBH method and 1.7% for the PBH method. The only different trend is observed in only one configuration of the NFBH method using the Sorting Heuristic 4. The mean success rate obtained by this configuration starts at 95.4% when the bid density is 0.25 and increases to 98.5% when the bid density is 5.

The significance of the effect of the sorting heuristic functions depends on the heuristic method used. Referring to Table 4, for the GAPH method, the Sorting Heuristics 2, 3, and 4 produces significantly better solutions than the Sorting Heuristic 1 which considers only the price of a bid, however, there is no significant difference among the solution produced by these three sorting heuristic functions. For the remaining heuristic methods, LRBH, NFBH, and PBH, the Sorting Heuristic 2 and 3 produces significantly better solutions than the Sorting Heuristic 1. However, there is no significant difference observed in the mean success rate of the solutions when Sorting Heuristic 2 (the average profit per bid) or 3 (the maximum profit per bid) is used. Sorting Heuristic 4 (using the linear relaxation values) dominates all other sorting heuristic functions producing the best mean success rates for the LRBH, NFBH, and PBH methods.

The plots of the mean running times of the MIP solver (for finding the optimal solutions) and the proposed heuristic methods versus the number of CPU cores for each sorting heuristic function are presented in Figure 6. The mean

TABLE 3. Pairwise comparisons of the heuristic methods for each sorting heuristic function provided by the simple main effect analysis of the two-way ANOVA test. Mean Diff. (A-B) column presents the difference between the mean success rates of the heuristic methods at columns A and B using the Bonferroni correction. (*) indicates that the corresponding mean difference is significant at the $\alpha = 0.05$ significance level.

Sorting Heuristic	A	B	Mean Diff. (A-B)
Sorting Heuristic 1	GAPH	LRBH	**-8.747***
		NFBH	**-7.884***
		PBH	**-9.780***
	LRBH	NFBH	**0.863***
		PBH	**-1.032***
	NFBH	PBH	**-1.895***
Sorting Heuristic 2	GAPH	LRBH	**-9.208***
		NFBH	**-8.059***
		PBH	**-9.874***
	LRBH	NFBH	**1.149***
		PBH	**-0.666***
	NFBH	PBH	**-1.815***
Sorting Heuristic 3	GAPH	LRBH	**-9.304***
		NFBH	**-8.116***
		PBH	**-9.978***
	LRBH	NFBH	**1.187***
		PBH	**-0.674***
	NFBH	PBH	**-1.861***
Sorting Heuristic 4	GAPH	LRBH	**-11.724***
		NFBH	**-10.088***
		PBH	**-11.279***
	LRBH	NFBH	**1.636***
		PBH	**0.445***
	NFBH	PBH	**-1.191***

TABLE 4. Pairwise comparisons of the sorting heuristic functions for each heuristic method provided by the simple main effect analysis of the two-way ANOVA test. Mean Diff. (A-B) column presents the difference between the mean success rates of the sorting heuristic functions at columns A and B using the Bonferroni correction. (*) indicates that the corresponding mean difference is significant at the $\alpha = 0.05$ significance level.

Heur. Met.	A	B	Mean Diff. (A-B)
GAPH	Sort. Heur. 1	Sort. Heur. 2	**-1.234***
		Sort. Heur. 3	**-1.184***
		Sort. Heur. 4	**-1.234***
	Sort. Heur. 2	Sort. Heur. 3	0.050
		Sort. Heur. 4	0.000
	Sort. Heur. 3	Sort. Heur. 4	-0.050
LRBH	Sort. Heur. 1	Sort. Heur. 2	**-1.695***
		Sort. Heur. 3	**-1.741***
		Sort. Heur. 4	**-4.211***
	Sort. Heur. 2	Sort. Heur. 3	-0.046
		Sort. Heur. 4	**-2.516***
	Sort. Heur. 3	Sort. Heur. 4	**-2.470***
NFBH	Sort. Heur. 1	Sort. Heur. 2	**-1.409***
		Sort. Heur. 3	**-1.416***
		Sort. Heur. 4	**-3.438***
	Sort. Heur. 2	Sort. Heur. 3	-0.007
		Sort. Heur. 4	**-2.029***
	Sort. Heur. 3	Sort. Heur. 4	**-2.021***
PBH	Sort. Heur. 1	Sort. Heur. 2	**-1.329***
		Sort. Heur. 3	**-1.382***
		Sort. Heur. 4	**-2.734***
	Sort. Heur. 2	Sort. Heur. 3	-0.053
		Sort. Heur. 4	**-1.405***
	Sort. Heur. 3	Sort. Heur. 4	**-1.351***

running time for the MIP solver is approximately 3 seconds for the test instances with 2592 cores and goes all the way up to approximately 380 seconds for the instances with 10368 cores. Note that the running time of the MIP solver is independent of the sorting heuristic function used, therefore the plot is the same for all sorting heuristic functions. It is included in all the plots for easy comparison to the heuristic methods.

Compared to the other sorting heuristic functions, the Sorting Heuristic 4 causes all the heuristic methods to produce the solutions faster. Recall that the Sorting Heuristic 4 uses the value of the decision variable x_k obtained after solving the linear relaxation of the integer program of the WDP. Therefore, each heuristic method using the Sorting Heuristic 4, first solves a linear model which requires some CPU time. However, it is seen that this step pays off, and it causes the overall running time to be less for all heuristic methods.

Among the proposed heuristics, the LRBH method is the slowest of them all. Solving a linear model for each bid and the placement phase of this heuristic contributes to the increased running times. For the largest test instances with 10384 cores, the mean running time of this heuristic using Sorting Heuristic 4 is approximately 95 seconds. Solving a network model instead of a general linear model, and a faster placement method results in faster execution times for the NFBH method. Instead of 95 seconds, each largest instance requires approximately 63 seconds on average. In the LRBH and NFBH methods, a general linear or network model is solved for each bid in the test instance. However, in the PBH method, the bid set is divided into fixed-size partitions, and a fixed-size integer model is solved for each partition. Although each optimization step in the PBH method may require more time than the optimization steps of the LRBH and the NFBH methods, the number of optimization steps is smaller by a factor of the partition size in the PBH method. Therefore, it is observed that the PBH method runs faster than the LRBH and the NFB methods with a mean running time of 34 seconds per instance using the Sorting Heuristic 4. The pure greedy method GAPH is the fastest method among all the proposed heuristic methods. The mean running time for the largest test cases is less than two seconds when the Sorting Heuristic 4 is used because of the linear relaxation solution overhead, less than 2 milliseconds when the other sorting heuristic functions are used.

Note that although large instances are avoided in the test suite in order to be able to find the optimal solutions, the gap in the running times of the MIP solver and the proposed heuristics increase exponentially as the size of the problem instances increases.

Bid density is another parameter that is related to the difficulty of the problem instances. The plots of the mean running times of the MIP solver and the proposed heuristic methods versus the bid density for each sorting heuristic function are presented in Figure 7. For the bid densities lower than 1, the mean running times of the LRBH and the NFBH methods are on par with the mean running times of the MIP solver, whereas the PBH and the GAPH methods are still faster than the MIP solver even for these lower densities. As the bid density increases, it is observed that the gap between the mean running time of the MIP solver and that of the heuristic methods increases exponentially.

VIII. CONCLUSION

In this study, we proposed an energy-aware combinatorial auction-based model for the resource allocation problem in clouds. The proposed ECO-VMAP model allows users to declare their complex virtual machine (VM) requirements in a cloud using the provided bidding language. Users can bundle their VM requests in a single bid by declaring complementarities and substitutabilities among the requested VMs. Based on the collective information obtained from the bids of the users, the model finds an optimum allocation of VMs to the users while also finding an optimum placement of the allocated VMs to the physical servers such that the energy usage is minimized. Thus, the model aims to increase the expected profit of the cloud provider while satisfying as many users as possible and minimizing the energy cost of the physical resources. The model also incorporates the non-linear energy usage characteristics of the physical servers to calculate the energy cost accurately. The ECO-VMAP model is expected to contribute reducing the carbon footprint of the cloud infrastructures for providing sustainable cloud services.

The study includes the mathematical definition of the ECO-VMAP model and a linear integer program for the optimization problem of the model. However, since the optimization problem is proven to be NP-hard, four heuristic methods with different complexities were proposed. Each heuristic method also uses a sorting heuristic function, and therefore, four different sorting heuristic functions were also proposed resulting in a total of 16 heuristic configurations. A test case generator for generating ECO-VMAP model instances was developed, and a comprehensive test suite containing approximately 4000 test instances were prepared. Several experiments were conducted to estimate the real-life performance of the model and the proposed heuristics.

In the first experiment, the ECO-VMAP model was compared to a first-come, first-served (FCFS) system. A simulation conducted for the FCFS-based system that processes the requests of the users as they arrive and places the allocated VMs to the currently available physical servers such that the energy cost is minimized. The simulation results were then compared to the allocations found by the ECO-VMAP model. The results indicate that the expected profit increase is approximately 42% on average when the ECO-VMAP model is used.

In the second experiment, the performances of the proposed heuristic methods were measured and statistically analyzed using the prepared test instances. The GAPH method using the Sorting Heuristic 2 or 3, provides solutions within 13% of the optimum objective value on average in a fraction of a second even for the largest test cases which makes it suitable for very large real-life problem instances when the solutions are required in a short time frame. The best results are obtained when the LRBH method is used with the Sorting Heuristic 4. This method provides solutions within only 1% of the optimal objective value on average, however, for the expense of the increased running time. The PBH method is considerably faster than the LRBH method while providing slightly worse solutions than the LRBH method which are within 1.5% of the optimal solutions on average. The NFBH method, in contrast with the other methods, has an interesting feature of providing the best results when the bid density value is high which also makes this method suitable when the number of virtual resources requested is much greater than the available physical resources in the cloud. Thus, depending on the problem instance size, the time limit for obtaining a solution, and the bid density of the problem instance, it is considered that each proposed heuristic method has a utility. Although the standard deviations of the success rates of the proposed heuristics are relatively low, especially for the LRBH, NFBH, and PBH methods, depending on the available computational resources all the proposed heuristics can be executed simultaneously, and the best solution provided by them can be used to reduce the chance of obtaining a low-quality solution.

By means of the features provided the ECO-VMAP model and the proposed efficient heuristic methods, it is considered that the ECO-VMAP model can be used for resource allocation even in large cloud environments while providing higher expected profits for the cloud provider, satisfying VM allocations for the users, and providing reduced carbon footprints for the environment.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their helpful comments that have helped to improve the paper.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. (2009). *Above the Clouds: A Berkeley View of Cloud Computing*. Accessed: Dec. 2020. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[2] A. Weiss, "Computing in the clouds," *netWorker*, vol. 11, no. 4, pp. 16–25, 2007.

[3] P. M. Mell and T. Grance. (2011). *The NIST Definition of Cloud Computing*. Accessed: Dec. 2020. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-145/final

[4] Y. Joshi and P. Kumar, "Introduction to data center energy flow and thermal management," in *Energy Efficient Thermal Management of Data Centers*, Y. Joshi and P. Kumar, Eds. Boston, MA, USA: Springer, 2012, pp. 1–38.

[5] R. Miller. (2020). *Rack Density Keeps Rising at Enterprise Data Centers*. Accessed: Dec. 2020. [Online]. Available: https://datacenterfrontier.com/rack-density-keeps-rising-at-enterprise-data-centers/

[6] I. Cooperation, "Intel IT: Extremely energy-efficient, high-density data centers," 2016. Accessed: Dec. 2020. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/best-practices/intel-it-extremely-energy-efficient-high-density-data-centers-paper.pdf

[7] A. Shehabi, S. J. Smith, E. Masanet, and J. Koomey, "Data center growth in the united states: Decoupling the demand for services from electricity use," *Environ. Res. Lett.*, vol. 13, no. 12, Dec. 2018, Art. no. 124030.

[8] E. Masanet, A. Shehabi, N. Lei, S. Smith, and J. Koomey, "Recalibrating global data center energy-use estimates," *Science*, vol. 367, no. 6481, p. 984, 2020.

[9] N. Jones, "How to stop data centres from gobbling up the world's electricity," *Nature*, vol. 561, no. 7722, pp. 163–166, 2018. Accessed: Dec. 2020. [Online]. Available: https://www.nature.com/articles/d41586-018-06610-y

[10] R. Bashroush and A. Lawrence. Beyond PUE: Tackling IT's Wasted Terawatts. Uptime Institute, 2020. Accessed: Dec. 2020. [Online]. Available: https://uptimeinstitute.com/beyond-pue-tackling-it%E2%80%99s-wasted-terawatts

[11] A. Andrae and T. Edler, "On global electricity usage of communication technology: Trends to 2030," *Challenges*, vol. 6, no. 1, pp. 117–157, Apr. 2015.

[12] F. Teng, L. Yu, T. Li, D. Deng, and F. Magoulès, "Energy efficiency of VM consolidation in IaaS clouds," *J. Supercomput.*, vol. 73, no. 2, pp. 782–809, Feb. 2017.

[13] M. Avgerinou, P. Bertoldi, and L. Castellazzi, "Trends in data centre energy consumption under the European code of conduct for data centre energy efficiency," *Energies*, vol. 10, no. 10, p. 1470, Sep. 2017.

[14] B. Whitehead, D. Andrews, A. Shah, and G. Maidment, "Assessing the environmental impact of data centres part 1: Background, energy use and metrics," *Building Environ.*, vol. 82, pp. 151–159, Dec. 2014.

[15] R. C. Zoie, R. D. Mihaela, and S. Alexandru, "An analysis of the power usage effectiveness metric in data centers," in *Proc. 5th Int. Symp. Electr. Electron. Eng. (ISEEE)*, Oct. 2017, pp. 1–6.

[16] EURECA Project. (2020). *EU Resource Efficiency Coordination Action*. Accessed: Dec. 2020. [Online]. Available: https://www.dceureca.eu/

[17] A. Yousafzai, A. Gani, R. M. Noor, M. Sookhak, H. Talebian, M. Shiraz, and M. K. Khan, "Cloud resource allocation schemes: Review, taxonomy, and opportunities," *Knowl. Inf. Syst.*, vol. 50, no. 2, pp. 347–381, Feb. 2017.

[18] Á. L. García, E. Fernández-del-Castillo, P. O. Fernández, I. C. Plasencia, and J. M. de Lucas, "Resource provisioning in science clouds: Requirements and challenges," *Softw. Pract. Exper.*, vol. 48, no. 3, pp. 486–498, Mar. 2018.

[19] A. H. Özer and C. Özturan, "A model and heuristic algorithms for multi-unit nondiscriminatory combinatorial auction," *Comput. Oper. Res.*, vol. 36, no. 1, pp. 196–208, Jan. 2009.

[20] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, May 2012.

[21] M. S. Hasan and E.-N. Huh, "Heuristic based energy-aware resource allocation by dynamic consolidation of virtual machines in cloud data center," *KSII Trans. Internet Inf. Syst.*, vol. 7, no. 8, pp. 1825–1842, 2013.

[22] Y. O. Yazir, C. Matthews, R. Farahbod, S. Neville, A. Guitouni, S. Ganti, and Y. Coady, Eds., "Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis," in *Proc. IEEE 3rd Int. Conf. Cloud Comput.*, Jul. 2010, pp. 91–98.

[23] B. Yin, Y. Wang, L. Meng, and X. Qiu, "A multi-dimensional resource allocation algorithm in cloud computing," *J. Inf. Comput. Sci.*, vol. 9, no. 11, pp. 3021–3028, 2012.

[24] K. Bessai, S. Youcef, A. Oulamara, C. Godart, and S. Nurcan, "Multi-objective resources allocation approaches for workflow applications in cloud environments," in *On the Move to Meaningful Internet Systems: OTM 2012 Workshops*, P. Herrero, H. Panetto, R. Meersman, and T. Dillon, Eds. Berlin, Germany: Springer, 2012, pp. 654–657.

[25] Y. C. Lee and A. Y. Zomaya, Eds., *Resource Allocation for Energy Efficient Large-Scale Distributed Systems: Information Systems, Technology and Management*. Berlin, Germany: Springer, 2010.

[26] H. M. Lee, Y.-S. Jeong, and H. J. Jang, "Performance analysis based resource allocation for green cloud computing," *J. Supercomput.*, vol. 69, no. 3, pp. 1013–1026, Sep. 2014.

[27] H. Zhao, J. Wang, F. Liu, Q. Wang, W. Zhang, and Q. Zheng, "Power-aware and performance-guaranteed virtual machine placement in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 6, pp. 1385–1400, Jun. 2018.

[28] X. Ye, Y. Yin, and L. Lan, "Energy-efficient many-objective virtual machine placement optimization in a cloud computing environment," *IEEE Access*, vol. 5, pp. 16006–16020, 2017.

[29] M. Gaggero and L. Caviglione, "Model predictive control for energy-efficient, quality-aware, and secure virtual machine placement," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 1, pp. 420–432, Jan. 2019.

[30] X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 113–128, Feb. 2018.

[31] S. Wang, A. Zhou, C.-H. Hsu, X. Xiao, and F. Yang, "Provision of data-intensive services through energy-and QoS-aware virtual machine placement in national cloud data centers," *IEEE Trans. Emerg. Topics Comput.*, vol. 4, no. 2, pp. 290–300, Apr. 2016.

[32] X. Liu, B. Cheng, and S. Wang, "Availability-aware and energy-efficient virtual cluster allocation based on multi-objective optimization in cloud datacenters," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 2, pp. 972–985, Jun. 2020.

[33] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," in *Advances in Computers*, vol. 82, M. V. Zelkowitz, Ed. Elsevier, 2011, ch. 3, pp. 47–111.

[34] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P. P. Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q. M. Malluhi, N. Tziritas, A. Vishnu, S. U. Khan, and A. Zomaya, "A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems," *Computing*, vol. 98, no. 7, pp. 751–774, Jul. 2016.

[35] F. Sheikholeslami and N. J. Navimipour, "Auction-based resource allocation mechanisms in the cloud environments: A review of the literature and reflection on future challenges," *Concurrency Comput. Pract. Exper.*, vol. 30, no. 16, p. e4456, Aug. 2018.

[36] Z.-H. Zhan, X.-F. Liu, Y.-J. Gong, J. Zhang, H. S.-H. Chung, and Y. Li, "Cloud computing resource scheduling and a survey of its evolutionary approaches," *ACM Comput. Surv.*, vol. 47, no. 4, pp. 1–33, Jul. 2015.

[37] M. C. S. Filho, C. C. Monteiro, P. R. M. Inácio, and M. M. Freire, "Approaches for optimizing virtual machine placement and migration in cloud environments: A survey," *J. Parallel Distrib. Comput.*, vol. 111, pp. 222–250, Jan. 2018.

[38] M. Zakarya and L. Gillam, "Energy efficient computing, clusters, grids and clouds: A taxonomy and survey," *Sustain. Computing: Informat. Syst.*, vol. 14, pp. 13–33, Jun. 2017.

[39] M. Zakarya, "Energy, performance and cost efficient datacenters: A survey," *Renew. Sustain. Energy Rev.*, vol. 94, pp. 363–385, Oct. 2018.

[40] R. Zolfaghari and A. M. Rahmani, "Virtual machine consolidation in cloud computing systems: Challenges and future trends," *Wireless Pers. Commun.*, vol. 115, no. 3, pp. 2289–2326, Dec. 2020.

[41] P. Bajari and A. Hortaçsu, "Economic insights from Internet auctions," *J. Econ. Literature*, vol. 42, no. 2, pp. 457–486, 2004.

[42] S. de Vries and R. V. Vohra, "Combinatorial auctions: A survey," *INFORMS J. Comput.*, vol. 15, no. 3, pp. 284–309, Aug. 2003.

[43] P. Cramton, Y. Shoham, and R. Steinberg, Eds., *Combinatorial Auctions*. Cambridge, MA, USA: MIT Press, 2006.

[44] C. Jackson, "Technology for spectrum markets," Ph.D. dissertation, Dept. Elect. Eng., Massachusetts Institute Technology, Cambridge, MA, USA, 1976.

[45] S. J. Rassenti, V. L. Smith, and R. L. Bulfin, "A combinatorial auction mechanism for airport time slot allocation," *Bell J. Econ.*, vol. 13, no. 2, pp. 402–417, 1982.

[46] Y. Sheffi, "Combinatorial auctions in the procurement of transportation services," *Interfaces*, vol. 34, no. 4, pp. 245–252, Aug. 2004.

[47] R. L. Graves, L. Schrage, and J. Sankaran, "An auction method for course registration," *Interfaces*, vol. 23, no. 5, pp. 81–92, Oct. 1993.

[48] S. Zaman and D. Grosu, "Combinatorial auction-based allocation of virtual machine instances in clouds," *J. Parallel Distrib. Comput.*, vol. 73, no. 4, pp. 495–508, Apr. 2013.

[49] S. Zaman and D. Grosu, "A combinatorial auction-based mechanism for dynamic VM provisioning and allocation in clouds," *IEEE Trans. Cloud Comput.*, vol. 1, no. 2, pp. 129–141, 2013.

[50] T. T. Huu and C.-K. Tham, "An auction-based resource allocation model for green cloud computing," in *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, Mar. 2013, pp. 269–278.

[51] X.-Y. Wang, X.-W. Wang, and M. Huang, "A multiple attribute decision and bidding based cloud resource dynamic allocation method," in *Proc. 8th ChinaGrid Annu. Conf.*, Aug. 2013, pp. 22–27.

[52] X. Tan, A. Leon-Garcia, Y. Wu, and D. H. K. Tsang, "Online combinatorial auctions for resource allocation with supply costs and capacity limits," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 4, pp. 655–668, 2020.

[53] M. Gamsız and A. H. Özer, "An auction based mathematical model for energy-aware virtual machine allocation in clouds," in *Proc. 4th Int. Conf. Comput. Sci. Eng. (UBMK)*, Sep. 2019, pp. 1–6.

[54] N. Nisan, "Bidding and allocation in combinatorial auctions," in *Proc. 2nd ACM Conf. Electron. Commerce (EC)*, New York, NY, USA: Association for Computing Machinery, 2000, pp. 1–12.

[55] The Standard Performance Evaluation Corporation. (2017). *Comparison Based on Results for the Named Systems as Published at www.spec.org as of 25 October 2017 and 8 November 2017. Spec and the Benchmark Name Specpower_ssj are Registered Trademarks of the Standard Performance Evaluation Corporation*. [Online]. Available: https://www.spec.org/power_ssj2008

[56] Schneider Electric Data Center Science Center. (2020). *Data Center Efficiency and PUE Calculator*. Accessed: Dec. 2020. [Online]. Available: https://www.se.com/ww/en/work/solutions/system/s1/data-center-and-network-systems/trade-off-tools/data-center-efficiency-and-pue-calculator/

[57] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1409–1434, 2nd Quart., 2019.

[58] P.-J. Maenhaut, B. Volckaert, V. Ongenae, and F. De Turck, "Resource management in a containerized cloud: Status and challenges," *J. Netw. Syst. Manage.*, vol. 28, no. 2, pp. 197–246, Apr. 2020.

[59] C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, "Cloud container technologies: A State-of-the-Art review," *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 677–692, Jul. 2019.

[60] M. A. Rodriguez and R. Buyya, "Container-based cluster orchestration systems: A taxonomy and future directions," *Softw. Pract. Exp.*, vol. 49, no. 5, pp. 698–719, 2019.

[61] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and Linux containers," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Mar. 2015, pp. 171–172.

[62] T. Adufu, J. Choi, and Y. Kim, "Is container-based technology a winner for high performance scientific applications?" in *Proc. 17th Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Aug. 2015, pp. 507–510.

[63] E. G. Young, P. Zhu, T. Caraza-Harter, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "The true cost of containing: A gVisor case study," in *Proc. 11th USENIX Workshop Hot Topics Cloud Comput. (HotCloud)*, Renton, WA, USA: USENIX Association, 2019. Accessed: Dec. 2020. [Online]. Available: https://www.usenix.org/conference/hotcloud19/presentation/young

[64] M. Mattetti, A. Shulman-Peleg, Y. Allouche, A. Corradi, S. Dolev, and L. Foschini, "Securing the infrastructure and the workloads of linux containers," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Sep. 2015, pp. 559–567.

[65] R. Stoyanov and M. J. Kollingbaum, "Efficient live migration of Linux containers," in *High Performance Computing*, R. Yokota, M. Weiland, J. Shalf, and S. Alam, Eds. Cham, Switzerland: Springer, 2018, pp. 184–193.

[66] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A survey on virtual machine migration: Challenges, techniques, and open issues," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1206–1243, 2nd Quart., 2018.

[67] S. K. Tesfatsion, C. Klein, and J. Tordsson, "Virtualization techniques compared: Performance, resource, and power usage overheads in clouds," in *Proc. ACM/SPEC Int. Conf. Perform. Eng.* Berlin, Germany: Association for Computing Machinery, 2018, pp. 145–156.

[68] P. Sharma, L. Chaufournier, P. Shenoy, and Y. C. Tay, "Containers and virtual machines at scale: A comparative study," in *Proc. 17th Int. Middleware Conf.* Trento, Italy: Association for Computing Machinery, 2016, p. 1.

[69] C. Prakash, P. Prashanth, U. Bellur, and P. Kulkarni, "Deterministic container resource management in derivative clouds," in *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, Apr. 2018, pp. 79–89.

[70] *Amazon Web Services*. (2020). *Amazon EC2 Instance Types—Amazon Web Services*. Accessed: Dec. 2020. [Online]. Available: https://aws.amazon.com/ec2/instance-types/

[71] L. Dignan. (2016). *AWS Cloud Computing OPS, Data Centers, 1.3 Million Servers Creating Efficiency Flywheel*. Accessed: Dec. 2020. [Online]. Available: https://www.zdnet.com/article/aws-cloud-computing-ops-data-centers-1-3-million-servers-creating-efficiency-flywheel/

[72] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach* (Pearson Series in Artificial Intelligence), 4th ed. Hoboken, NJ, USA: Pearson, 2020.

[73] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, Eds., *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1993.

[74] A. V. Goldberg and R. E. Tarjan, "Finding minimum-cost circulations by successive approximation," *Math. Oper. Res.*, vol. 15, no. 3, pp. 430–466, Aug. 1990.

[75] (2020). *Gurobi Optimizer*. Accessed: Dec. 2020. [Online]. Available: https://www.gurobi.com

[76] S. E. Maxwell and H. D. Delaney, *Designing Experiments Analyzing Data: A Model Comparison Perspective*, 2nd ed. London, U.K.: Lawrence Erlbaum Associates, 2004.

MUSTAFA GAMSIZ received the B.S. degree in computer engineering from Ege University, Izmir, Turkey, in 2010, and the M.S. degree in computer engineering from Marmara University, Istanbul, Turkey, in 2019 under the supervision of Prof. Ali Haydar Özer.

From 2011 to 2017, he worked as a Senior Researcher at TÜBİTAK. Since 2017, he has been working as a Senior Software Engineer at NETAŞ, Istanbul. His research interests include cloud computing, combinatorial optimization, heuristic design, evolutionary algorithms, and communication systems.

ALİ HAYDAR ÖZER (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer engineering from Boğaziçi University, Istanbul, Turkey, in 2002, 2004, and 2011, respectively.

From 2002 to 2011, he worked as a Research Assistant with the Department of Computer Engineering, Boğaziçi University. Since 2011, he has been working as an Assistant Professor with the Department of Computer Engineering, Marmara University, Istanbul, Turkey. His research interests include electronic market design, auctions and barter exchanges, cloud computing, combinatorial optimization, and heuristic design.