The Journal of Supercomputing
https://doi.org/10.1007/s11227-023-05505-8

An approximation algorithm for virtual machine placement in cloud data centers

Zahra Mahmoodabadi[1] · **Mostafa Nouri-Baygi**[1]

Accepted: 14 June 2023

Abstract
This study addresses the energy efficiency challenge in cloud data centers by examining the Virtual Machine Placement (VMP) problem. VMP involves mapping virtual machines (VMs) to physical machines (PMs) under capacity constraints. The paper focuses on the bin packing with linear usage cost (BPLUC) variant of bin packing, which includes fixed and variable costs in the calculation of the cost of a used bin. We prove that every approximation algorithm for the bin and vector bin packing can be used for BPLUC and VBPLUC, respectively. We propose a more power-efficient approach to VMP by applying a vector bin packing algorithm to minimize power consumption in data centers. We test the proposed algorithm on various synthetic and real workloads, and the experimental results demonstrate that it is more power-efficient than existing algorithms for VMP. The findings suggest that the proposed algorithm has significant implications for energy-efficient strategies in cloud data centers. Generally, this study makes contributes to the development of energy-efficient approaches to VMP that can help reduce power consumption and improve the sustainability of cloud data centers.

Keywords Approximation algorithm · VM placement · Vector bin packing

1 Introduction

One of the primary concerns of cloud providers is the efficient management of available resources. Minimizing power consumption and improving performance is a hot topic these days. Idle server static power consumption is more than 60% of server peak power consumption [1]. Virtualization is one of the proposed solutions for

✉ Mostafa Nouri-Baygi
nouribaygi@um.ac.ir

Zahra Mahmoodabadi
za.mahmoodabadi@mail.um.ac.ir

[1] Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran

Published online: 05 July 2023

optimal resource utilization. This technology allows cloud providers to create multiple VMs on a single PM, thus improving resource efficiency.

Deciding how to allocate VMs to PMs is called virtual machine placement(VMP)and is an NP-hard optimization problem. A common strategy to minimize data center energy consumption is to minimize the number of active PMs [2, 3]. Considering VMs as items and PMs as bins leads to the bin packing problem. This problem is strongly NP-hard [4]. The problem is assigning items to bins to minimize cost. The cost of bin packing is the number of bins used to pack items.

There is a variant of bin packing called Bin Packing with Linear Usage Cost (BPLUC) [5] that accounts for the cost in a different way. BPLUC bin costs consist of two parts: fixed and variable costs that are associated with each unit of capacity used. The problem is to assign each item to a bin, considering capacity constraints, so that the total cost of all bins is minimized. Here, the energy required per unit of PM usage is considered as the variable cost. Fixed cost is defined as the energy required for an idle PM with zero utilization. BP is a particular case of BPLUC, where all fixed costs set to 1 and variable costs set to 0.

When we consider multiple dimensions for items and bins, the problem is called VBP and VBPLUC for BP and BPLUC, respectively. For example, all items and bins have two dimensions: volume and weight. However, data center VMs and PMs include multiple dimensions such as CPU, RAM, bandwidth, and storage. This study examines the VMP problem with three dimensions CPU, RAM, and bandwidth, with the aim to minimize power consumption.

To the best of the author's knowledge, there is no existing approximation algorithm available for the energy-efficient Virtual Machine Placement (VMP) problem. In this research paper, we establish a comprehensive and meaningful connection between two fundamental problems, namely Bin Packing (BP) and Bin Packing with Linear Usage Cost (BPLUC), as well as between Vector Bin Packing (VBP) and Vector Bin Packing with Linear Usage Cost (VBPLUC). This correspondence is relevant and applicable to a novel family of VBP problems. The problem of VMP shares similarities with VBPLUC, as both involve considering the cost of a Physical Machine (PM) in terms of the power it consumes. The power consumption is determined by the utilization of the PM, which, in turn, depends on the number of Virtual Machines (VMs) allocated to it and the resource requirements of each VM.

The main contributions of this paper are:

1. We prove that any approximation algorithm for BP with an approximation ratio α is an approximation algorithm for BPLUC with an approximation ratio based on α for homogeneous hosts.
2. We prove that any approximation algorithm for VBP with an approximation ratio β is an approximation algorithm for VBPLUC with an approximation ratio based on β for homogeneous hosts.
3. We apply the proposed algorithm for VBP presented in [6] for VMP with power minimization objective in heterogeneous data centers.

The remaining sections of this paper are organized as follows: Sect. 2 provides an overview of the related work for VMP. In Sect. 3, we first present the system architecture, then define and formulate the BPLUC and VBPLUC problems considering BP and VBP problems. Section 4 deals with the first and second contributions. Section 5 reports the experiments on real datasets and compares the results with other algorithms. Finally, conclusions and future work are presented in Sect. 6.

2 Related work

This section provides a brief overview of related work available on the VMP topic. There is extensive literature on how to solve this problem with different goals, such as power minimization, network traffic minimization, economic revenue maximization, performance maximization, and resource utilization maximization.

There are several attempts to solve VMP with the goal of power minimization. This section describes some of them. Deterministic methods [7, 8], heuristic methods [9–15], meta-heuristic methods [16–20] and other methods [21–25] have been suggested to solve VMP.

2.1 Deterministic methods

Mann et al. [8] solved the VMP issue when the virtual and physical machine processors are multicore. The authors presented some greedy algorithms based on PABFD [12] and a model based on Constraint Programming (CP). The objective function is based on the sum of weighted minimization of the number of active physical machines, reduction in the number of migrations, and minimization of SLA violations. In the proposed method, there are two stages based on CP. First, the method examines the search space for solutions of assigning VMs to PMs. In the second stage, the best mapping of virtual machine cores to physical machine cores is made for the allocation found. However, this model offers the best solution for the small instances compared to the greedy algorithms. For larger samples, for example, those containing more than 350 virtual machines, the method could not to find a solution within the given time.

Wie et al. in [7] attempt to simultaneously reduce active and idle physical machines' power and physical machines' activation time. The authors express the activation time of physical machines with a threshold constraint. Two mathematical models are presented with two objective functions, minimizing energy and minimizing physical machine activation time. The combination of solutions in each model dictates the final placement of virtual machines.

Deterministic methods are based on mathematical models and algorithms that provide a guaranteed optimal solution, given enough time and resources.

However, the computational complexity of deterministic methods often makes them infeasible for large-scale VM placement problems.

2.2 Heuristic methods

Ajmera et al. used a new criterion for power-based VMP [13]. In this study, the authors addressed two problems, the initial placement of VMs and finding a suitable target PM during migration. Choosing the right PM to place a VM on is first based on performance metrics(how much power is required for a given workload for each type of PM). The PM with minimum utilization to power ratio is selected when the VM is placed on it.

Another method to solve this problem is described by [10] called GRVMP. The authors proposed a greedy algorithm that randomly assigns PMS to VMs with the aim of power optimization. They introduced a new factor called resource wastage. There, PMs with minimal waste of resources are better suited for hosting VMs. The results show that this method is suitable for large data centers and does not work well for data centers with a small number of VMs and PMs.

Knowing that VMP is an NP-hard combinatorial problem, it is impossible to solve it optimally for a large number of VMs and PMs. Data centers typically consist of thousands to millions of VMs and PMs, and the best solution provided by a thorough search can be expensive. Therefore, a compromise must be made between the quality of the solution and the computational costs of a real cloud management system. Heuristic algorithms based on the bin packing problem are extensively employed in order to effectively minimize the number of physical machines(PMs) utilized and enhance energy efficiency. Beloglazov et al. [12] proposed a PABFD algorithm based on the Best Fit(BF) heuristic algorithm. The algorithm chooses the PM with the lowest power consumption for VM placement.

The paper [9] delves into the challenges encountered by cloud service providers when it comes to efficiently managing multiple cloud data centers. The primary objectives revolve around meeting the escalating demands of applications while simultaneously striving to minimize energy consumption. The paper proposes an energy-efficient method called Resource Allocation based on Request Prediction (RARP) in multiple cloud data centers. The RARP method anticipates application request volume and allocates VMs and PMs based on the minimum remaining resources available to minimize energy consumption. The proposed method is evaluated through extensive experiments, and the results show significant improvements in request detection accuracy and energy consumption compared to other algorithms.

In another study, Jagiti et al. [14] proposed an FF-based algorithm for VMP in the multidimensional case. In this study, the VMs are sorted based on the requirements for each dimension, and then the rank of their requests is aggregated for each resource. This value is used to sort the VMs in the FF algorithm.

In paper [15], two energy-efficient VM placement algorithms based on bin packing heuristics were proposed, namely Energy Efficient VM Placement (EEVMP) and Modified Energy Efficient VM Placement (MEEVMP). These

algorithms aim to reduce the number of idle hosts in the data center by optimizing the placement of VMs, thereby achieving a more energy-efficient resource utilization. Experimental results showed that EEVMP and MEEVMP can reduce energy consumption compared to the default VM placement algorithm PABFD. The study highlights the importance of efficient VM placement in achieving energy efficiency in data centers.

In [11]'s study, VMP defined four thresholds to distinguish little loaded, lightly loaded, normally loaded, medium loaded, and heavily loaded servers. The authors used these metrics to detect the appropriate PMs for VMs. Additionally, during the VM allocation process, they suggested considering two factors: power consumption and SLA violations. This method showed acceptable results compared to similar methods. Generally, Heuristic methods are based on practical experience and common sense and are used to find good-quality solutions that may not be optimal. Heuristic methods are often faster than deterministic methods, but the quality of the obtained solutions may vary depending on the specific problem instance.

2.3 Meta-heuristic methods

The paper [19] proposes an ant colony system (ACS) algorithm for energy-efficient dynamic virtual machine (VM) placement in data centers. The proposed algorithm uses ant-like agents to explore the search space and find an optimal solution to the VM placement problem, with the objective to minimize energy consumption while satisfying the resource demands of VMs and meeting the service-level agreements (SLAs) of cloud users. The algorithm also includes a dynamic migration strategy to deal with the changing workload demands of the data center. The authors report that the proposed algorithm outperforms existing VM placement algorithms in terms of energy consumption and SLA violation rate. The proposed algorithm also provides good scalability and robustness to changes in workload demand.

Paper [18] explores the use of a genetic algorithm (GA) for energy-efficient virtual machine (VM) placement in data centers. While GA is known for providing high-quality solutions, its fitness function is computationally demanding, limiting its use in large-scale systems or specific scenarios where fast VM placement is required. This paper proposes a data structure to reduce the complexity of the fitness computation from quadratic to linear and an alternative fitness function to reduce the number of instructions, resulting in an $11x$ acceleration of GA computation for energy-efficient VM placement in large-scale data centers. The study highlights the importance of VM placement in improving energy efficiency in data centers and proposes a novel approach to overcome the computational limitations of GA.

The article [20] discusses the importance of efficient virtual machine placement (VMP) to maximize the utilization of physical machines (PMs) in data centers and reduce energy consumption. The authors propose a Metaheuristic Virtual Machine Placement Framework toward the Power Efficiency of Sustainable Cloud Environment (MV-PESC) approach that uses an Extended Flower Pollination Optimization algorithm to improve VMP efficiency. The study evaluates the proposed approach using actual workload traces and compares it with state-of-the-art solutions. The

results show significant reductions in power consumption, active PMs, and execution time.

In [17], authors used resource reservation for VMP. Many service providers enable resource reservations for their customers to enable efficient cloud resource management and lower costs. The objective function of this research is based on instruction-energy and the goal is to effectively reduce energy consumption, and increase utilization of reserved resources. This study applied an evolutionary algorithm to obtain the best mapping of virtual machines to physical machines such that energy consumption is minimized.

The authors of [16] studied the VMP with the aim of minimizing energy, taking into account the non-deterministic requirements of virtual machines. Instead of using the deterministic values for resource requirements, they presented a random placement in which variations in resource requirements are represented as random variables. In this work, the VMP problem is formulated as a random optimization model considering non-deterministic resource requirements. The authors used a meta-heuristic algorithm to search VMP solution objects to minimize energy consumption in the data center.

Generally, metaheuristics methods for VM placement problems can suffer from local optimality problems. These methods rely on searching a large search space to find the optimal solution but can get stuck in suboptimal solutions that are locally optimal but not globally optimal. This can lead to sub-optimal solutions. In addition, metaheuristic methods can be computationally intensive and time-consuming, making them impractical for large-scale VM placement in large-scale data centers.

2.4 Other methods

In another study [22], the authors proposed a resource-aware algorithm for VMP. The first goal of the proposed algorithm is energy minimization in cloud IaaS, which is achieved by minimizing the number of active physical machines. This is implemented using a new method called Resource Usage Factor(RUF). RUF efficiently uses of physical machine resources by placing virtual machines on appropriate physical machines. The secondary goal is to minimize resource usage imbalances among active physical machines. This is achieved using a new resource usage model. This model can detect imbalanced resource utilization.

The inefficient use of resources can lead to low system utilization and more physical server usage, which increases power consumption. To address this issue, the paper [25] presents an energy-efficient topology-aware VM placement scheme in cloud DCs, formulated as a multi-objective optimization problem with a focus on minimizing power consumption and resource wastage. The proposed solution uses an advanced multi-objective discrete version of the JAYA (MOD-JAYA) algorithm to solve the combinatorial problem of VMP. The simulation results demonstrate the effectiveness of the proposed algorithm in solving the VMP problem compared to other existing schemes in terms of prominent assessment metrics.

The VMP issue has been investigated by [21] to reduce energy consumption. The authors proposed using game theory to solve the problem that has a successful

performance for the dynamic case of VMP(when requests come online). Another advantage of the proposed algorithm is that all optimal solutions are produced using this procedure. The proposed method has intelligent computational properties. It uses the initial solution and guarantees that there is a list of virtual machine migrations according to the initial solution that can lead to the solution of the problem. Algorithmic predictions are made using evolutionary game theory. The analysis of the obtained outcomes demonstrates that the proposed approach exhibits the capability to attain the optimal solution within the dynamic virtual machine placement (VMP) context, effectively optimizing energy consumption.

In game theory and multi-agent systems, the focus is on modeling the interactions and decisions of multiple agents. However, in the VM placement problem, the agents (i.e., the VMs) may have conflicting objectives, making coordination challenging. Moreover, the optimal placement decisions may depend on the placement decisions of other VMs, making it difficult to find a globally optimal solution.

Using machine learning techniques is one of the most effective methods in VMP. The authors of [23] tried to place virtual machines based on workload and required bandwidth of virtual machine. They applied the feedforward neural network, a helpful tool for time series forecasting. The proposed algorithm called PACPA shows acceptable results compared to similar algorithms.

In the work [24], Wang et al. tackled the issue of VM allocation and migration costs through the utilization of a distributed multi-agent (MA) based approach. The proposed MA first dispatches a cooperative agent to each PM to assist the PM in managing VM resources. An auction-based VM allocation mechanism is then applied to these agents to determine the allocation of VMs to PMs. PMs negotiate with each other to perform migration when it makes sense from a power-saving perspective. The proposed algorithm can be used in static and dynamic environments, and the results showed a significant reduction in energy consumption compared to other methods.

Applying machine learning techniques to the virtual machine (VM) placement problem can be highly advantageous for tackling intricate issues. However, certain challenges need to be addressed in order to ensure effective utilization of these techniques. These challenges encompass factors such as data availability and quality, which can impact the accuracy and reliability of the models. Additionally, the ability of machine learning models to generalize well across diverse VM placement scenarios is another area of concern. Furthermore, the complexity of the VM placement problem itself poses a challenge, requiring sophisticated approaches to handle its intricacies effectively.

Although BPLUC is commonly used for transportation problems, Cambazard et al. [26] has used this problem to minimize data center energy. The authors considered a bin packing problem where there are linear costs associated with using bins to model energy consumption. They also examined lower bounds based on linear programming and extended the global bin packing constraints to include cost information. They focused on ways to reduce energy costs by addressing the CPU requirements of client applications, IT equipment, and virtualization techniques. The problem is defined in heterogeneous data centers.

A comprehensive examination of the existing literature indicates the absence of an approximation algorithm specifically designed for solving the virtual machine placement (VMP) problem. Previous research usually looked only at heuristics, meta-heuristics, and a few exact techniques for solving the problem. Approximation methods are used to obtain a good-quality solution that is close to the optimal solution but may not be guaranteed to be optimal. Approximation methods are often faster than deterministic methods and can be used to solve large-scale VM placement problems that are infeasible to solve exactly. However, the quality of the solution obtained may depend on the specific approximation algorithm used and the input parameters. This work is conducted in response to the need for an approximation technique to know how significant the difference between the found solution and the best solution is.

3 System architecture and problem definition

This section is focused on the system architecture and problem definition of our work.

3.1 System architecture

The system architecture of our work is illustrated in Fig. 1. The diagram on the right-hand side of the figure illustrates the existence of three distinct layers leading to two fundamental mapping phases. First, user applications are mapped to VMs and

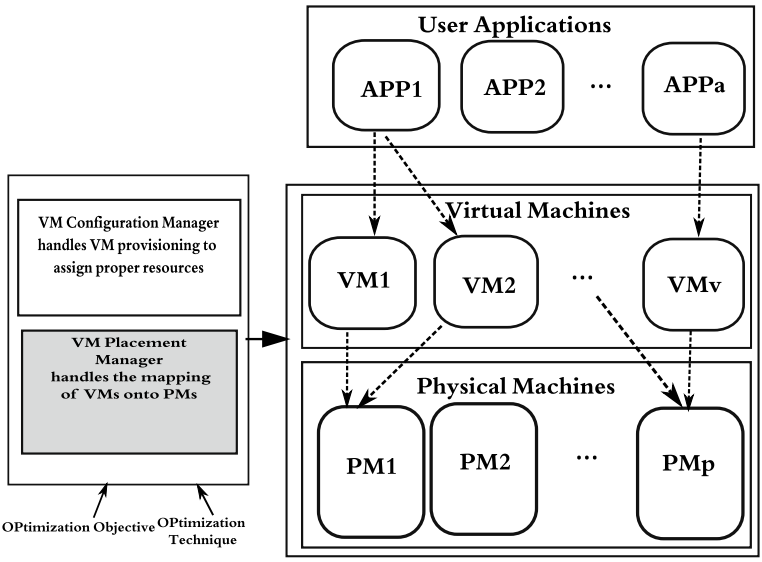

Fig. 1 System architecture of cloud computing. [27]

VMs are mapped to PMs. The two mapping phases are handled by two main entities, the VM Configuration Manager and the VM Placement Manager, respectively. VM configuration addresses issues related to VM deployment in terms of both the number and size of VMs (individual characteristics) and is not the focus of this study. In contrast, we assume the VM configuration has already been completed and focus solely on the next phase, VM placement. It is intended to consider different optimization goals and apply different optimization techniques to assign VMs to PMs. Our algorithm focuses on this part.

3.2 Problem definition

Here is a problem similar to the classical problem of bin packing. Table 1 presents the primary symbols used in this paper and provides an explanation of each of them.

BPLUC is a variant of the BP problem that defines the cost of a used bin differently. In BPLUC, we are given a set of n items, $V = \{v_1, \ldots, v_n\}$ with integer sizes and an unlimited supply of identical bins. A bin has a capacity S, a non-negative fixed cost, f, and a non-negative cost, c, for each unit of used capacity. Let B be the set of available bins, $B = \{B_1, B_2, \ldots\}$. A bin is used when at least one item is assigned to it. The cost of a used bin, B_j, $j \in B$, is a linear function $f + cl_j$, where l_j is the total size of the items in bin j. The problem is to assign each item to a bin under capacity constraints so that the sum of the costs of all bins is minimized. This problem is known as the Bin Packing with Linear Usage Cost problem (BPLUC). BP is a particular case of BPLUC with all f set to 1 and all c set to 0.

3.3 BPLUC formulation

The BPLUC can be defined using the following linear model. The objective function is to minimize the sum of the costs of all bins. l_j is the total loads of allocated items to the bin j. The variable x_{ij} is 1 if item i is assigned to PM j; otherwise it is 0. The decision variable y_j is 1 if bin j has at least one item and therefore is used; otherwise it is 0.

$$\begin{aligned} &\text{minimize} && \sum_{j \in \mathcal{B}} y_j (f + cl_j) \\ &\text{subject to} && l_j = \frac{\sum_{i \in \mathcal{V}} x_{ij} v_i}{S} * 100 \forall j \in \mathcal{B} \end{aligned} \tag{1}$$

$$\sum_{j \in \mathcal{B}} x_{ij} = 1 \, \forall i \in \mathcal{V} \tag{2}$$

$$y_j \geq x_{ij} \, \forall i \in \mathcal{V}, j \in \mathcal{B} \tag{3}$$

Table 1 Main Notations and Description

	Symbol	Description
Set	V	Set of items, where $\|V\| = n$
	B	Set of Bins, where $\|B\| = m$
	I	Set of VMs requests
	Q	Set of large VMs
	D	Set of small VMs
	M	Set of PMs configurations
Index	i	Index of item/VM, $i \in V$
	j	Index of Bin/ PM, $j \in B$
	t	Index of resource dimension, $t \in d$
Input parameters	S	Bin capacity
	W	Total size of all items in all bins
	k	Number of non-empty bins
	f	Non-negative fixed cost for a used bin
	c	Non-negative cost for each unit of a used bin
	W_t	Total size of all items in dimension t in all bins
	v_i	Item/VM size
	l_j	Total size of items in bin j
	s_t	Capacity of a bin in dimension t
	$l_{j,t}$	Total amount of space used in dimension t in bin j
	$v_{i,t}$	Requirement of item/ VM i in dimension t
	c_t	The cost of a unit space used in dimension t
	B_j^{max}	Power consumption of B_j when it is full utilization
	B_j^{min}	Power consumption of B_j in idle mode
	u_j^{CPU}	Normalized CPU utilization of B_j
	R_j^W	Resource wastage of B_j
	R_j^t	Remained normalized resource of B_j in dimension t
	u_j^t	Normalized resource utilization of B_j along dimension t
	d	Number of dimensions
	β	The parameter for classifying VMs to large and small classes
	p_i	i^{th} large VM
	m	Number of PMs in optimal allocation
	m_j	Number of PMs for the assignment of type j
	h	Number of classes
	r_L	Number of VM types in Q
	r	Number of classes
	b_j^t	Occupied capacity of B_j in dimension t
Variables	$x_{i,j}$	The value is 1 if item/VM i is allocated on bin/PM j otherwise the value is 0
	y_j	The value is 1 if bin/PM j has at least one item/VM otherwise the value is 0
	P^{tot}	Total power consumption of all PMs
	$\mathbb{R}^{tot}$	Total resource wastage of all PMs

$$\sum_{i \in \mathcal{V}} x_{ij} v_i \leq S y_j \, \forall j \in \mathcal{B} \tag{4}$$

$$x_{ij} \in \{0, 1\} \, \forall i \in \mathcal{V}, j \in \mathcal{B} \tag{5}$$

$$y_j \in \{0, 1\} \, \forall j \in \mathcal{B} \tag{6}$$

Constraint 1 defines the total load of bin j, where it depends on the items assigned to it. Constraint 2 expresses that all item requests must be assigned. Constraint 3 shows that a bin must have at least one item to be activated, $y_i = 1$. Constraint 4 is the capacity constraint. Constraints 5 and 6 determine the bound of the variables.

3.4 VBPLUC formulation

In the linear usage cost vector bin packing, bins and items have multiple dimensions, for instance, weight and volume. The goal is to allocate multidimensional items to storage bins in a way that does not violate capacity constraints in any dimension and minimizes costs. The cost is similar to the BPLUC problem.

Given a set of identical bins called $B = \{B_1, B_2, \dots\}$ where each bin, B_j, has similar characteristics as bins have in the BPLUC. A bin, B_j, has a capacity for each dimension and defined as, $S = \{s_1, \dots, s_d\}$, where $s_t, t \in d$ denotes the bin capacity in dimension t. $l_{j,t}$ is the load of dimension t in bin j. An item, $i \in V$, has multiple requirements and defined as a d dimensional vector $v_i = \{v_{i,1}, \dots, v_{i,d}\}$. The cost of a unit space used in dimension t is defined by c_t. The goal is to efficiently allocate items to bins to minimize the total cost of all bins.

The objective function of VBPLUC is defined as follows. The constraints are the same as the linear model presented in the previous section, except constraints 1 and 4 that are modified according to constraints 7 and 8.

$$\begin{aligned} \text{minimize} \quad & \sum_{j \in \mathcal{B}} y_j (f + \sum_{t=1}^{d} s_t l_{j,t}) \\ & l_{j,t} = \frac{\sum_{i \in \mathcal{V}} s_t}{x_{ij} v_{i,t}} * 100 \qquad \forall j \in \mathcal{B}, t \in [d] \end{aligned} \tag{7}$$

$$\sum_{i \in \mathcal{V}} x_{ij} * v_{i,t} \leq s_t \quad \forall j \in \mathcal{B}, t \in [d] \tag{8}$$

Where in equation 7, $v_{i,t}$ and s_t are the item sizes and capacity of bin B_j in dimension t, respectively. Equation 8 shows the capacity constraint for all dimensions.

3.5 BPLUC and power efficient VMP

Looking at the VMP problem from a power minimization perspective, each PM consumes different power for each utilization level. SPECPOWER [28] provides the first industry-standard benchmark for characterizing the power and performance of computer servers. Table 2 comes from SPECPOWER and shows two servers with different power and utilization characteristics. The G4 and G5 cores have CPU frequencies of 1860 and 2660 MIPS, respectively, and both models have 4096 MB of memory.

According to Table 2, we can define a linear relationship between power and utilization for each server. For example, the diagram Fig. 2 shows this relationship for server types in Table 2.

The linear equation approximating the relationship between power and utilization is presented on each line as $y = ax + b$, where y represents power and x represents the total utilization value (i.e., the overall usage of allocated virtual machines) for each server. Considering the cost of a bin in BPLUC, we have the relationship $f + cl_j, j \in B$. For a PM, we can assume that the cost is the power according to the total utilization defined by the equation $y = ax + b$. Here we can use the BPLUC cost definition such that x equals l_j, and a and b are equivalent to f and c, respectively. Therefore, BPLUC can be used to solve power-efficient VMP since their cost definitions are similar.

Table 2 Power consumption according to hosts utilization

Server	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP ProLiant G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP ProLiant G5	93.7	97	101	105	110	116	121	125	129	133	135

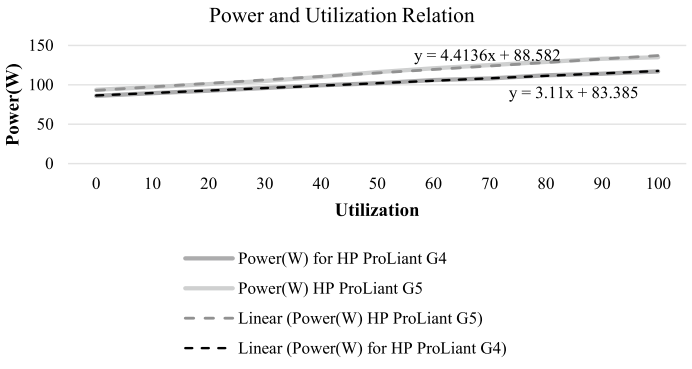

Fig. 2 Linear relation between power and utilization for 2 server types 2660 and 1860

4 Proposed method

In this section, we first prove that any approximation algorithm for BP (resp. VBP) can also be used as an approximation algorithm for BPLUC (resp. VBPLUC). Concerning these results and the relation between VBPLUC and VM placement problem, we then modify and implement an elaborate approximation algorithm for VBP by Bansal et al. [6] and compare it to other well-known and state-of-the-art VM placement methods in Section 5.

4.1 Approximation algorithms For BPLUC and VBPLUC

Theorem 1 *Any approximation algorithm for BP with approximation ratio α is an approximation algorithm for BPLUC with approximation ratio α.*

Proof We prove the theorem by setting up a one-to-one correspondence between any solution of BP to a solution for BPLUC with a linearly related cost. By this correspondence, the optimum solution for BP has a corresponding optimum solution for BPLUC, and the approximation solution for BP is also an approximation solution for BPLUC.

First, consider a solution $I = (B_1, B_2, \cdots, B_m)$ for BP, which satisfies the capacity constraints for each bin. In this solution, each B_j is the set of items placed in bin j. The cost of such a solution equals to kf, where k is the number of non-empty bins and f is the cost of using a bin. If we use the same allocation for BPLUC, the cost is equal to

$$\sum_{\substack{j \in B \\ j \text{ is not empty}}} (f + cl_j),$$

which can be written as

$$\begin{aligned}
&= \sum_{\substack{j \in B \\ j \text{ is not empty}}} f + \sum_{\substack{j \in B \\ j \text{ is not empty}}} cl_j \\
&= kf + \sum_{\substack{j \in B \\ j \text{ is not empty}}} cl_j \\
&= kf + c \sum_{\substack{j \in B \\ j \text{ is not empty}}} l_j \\
&= kf + c \sum_{i \in \mathcal{V}} v_i = kf + cW
\end{aligned}$$

In the last equation, W is the total size of all the items.

The above result shows that the cost of a solution for BPLUC is a constant value (the cost of a unit capacity of a bin multiplied by the total size of the items) greater than the cost of the same solution for BP. Furthermore, the optimum solution for BP and BPLUC are the same,

Another result from the above equations, which is related to any approximation algorithm for BP, is as follows. If an algorithm guarantee to find a solution within $\alpha \cdot OPT$, where OPT is the cost of the optimum solution, the cost of that solution for BPLUC will be at most $\alpha \cdot OPT + cW$. Since the optimum solution for BP remains optimum for BPLUC, the cost of the optimum solution for BPLUC is $OPT + cW$. Finally, the approximation ratio of the given algorithm for BPLUC is bounded by $\frac{\alpha \cdot OPT + cW}{OPT + cW} < \alpha$. □

Theorem 2 *Any approximation algorithm for VBP with approximation ratio α is an approximation algorithm for VBPLUC with approximation ratio α.*

Proof We can follow the same approach as the previous theorem and prove the theorem. Consider an approximation algorithm with an approximation ratio α for VBP. This means the cost of the solution of the algorithm is at most $\alpha \cdot OPT$, where OPT is the cost of the optimum solution. The cost of the corresponding optimum solution of the VBPLUC problem is $OPT + \sum_{t=1}^{d} c_t \cdot W_t$. In this equation, c_t is the cost of unit space used in dimension t, and W_t is the total amount of space used in dimension t for all bins.

We can bound the cost of the solution of the approximation algorithm for VBPLUC from above as follows. Let $I = (B_1, B_2, \cdots, B_m)$ denote the solution. The total cost is equal to

$$\sum_{\substack{j \in B \\ j \text{ is not empty}}} (f + \sum_{t=1}^{d} c_t \cdot l_{j,t}).$$

Here $l_{j,t}$ is the total amount of space used in dimension t in bin j. The total cost can be written as

$$= \sum_{\substack{j \in B \\ j \text{ is not empty}}} f + \sum_{\substack{j \in B \\ j \text{ is not empty}}} \sum_{t=1}^{d} c_t \cdot l_{j,t}$$

$$= kf + \sum_{\substack{j \in B \\ j \text{ is not empty}}} \sum_{t=1}^{d} c_t \cdot l_{j,t}$$

$$= kf + \sum_{t=1}^{d} \sum_{\substack{j \in B \\ j \text{ is not empty}}} c_t \cdot l_{j,t}$$

$$= kf + \sum_{t=1}^{d} (c_t \cdot \sum_{\substack{j \in B \\ j \text{ is not empty}}} l_{j,t})$$

$$= kf + \sum_{t=1}^{d} (c_t \cdot W_t)$$

Since the total cost of the approximate solution, kf is at most $\alpha \cdot OPT$, the total cost of the corresponding solution is at most $\alpha \cdot OPT + \sum_{t=1}^{d}(c_t \cdot W_t)$. Finally, the approximation ratio of the given algorithm for VBPLUC is bounded by $\frac{\alpha \cdot OPT + \sum_{t=1}^{d}(c_t \cdot W_t)}{OPT + \sum_{t=1}^{d}(c_t \cdot W_t)} < \alpha$. □

4.2 Mapping a VBP algorithm to VMP

Bin packing is an NP-hard problem, and there are many approximation algorithms for it. We use an algorithm packing items into at most $(1 + 2\epsilon)m + 1$ bins with ϵ resource augmentation in $(d - 1)$ dimensions, where m is the number of bins in optimal packing [6]. The authors studied the d-dimensional vector bin packing problem, and their algorithm is based on resource augmentation and rounding items. This algorithm has been modified to work with cloud domains and VMP for two objectives. For the first goal, the cost is assumed to be the minimum number of PMs used for VM allocation. The second objective considers the cost as the PM's power consumption.

4.2.1 VBP algorithm

Algorithm 1, called VBP, is inspired from [6]. Both PMs and VMs contain multiple resources(dimensions), such as CPU, RAM, and bandwidth. The goal is to assign VMs to the minimum number of PMs. In the VBP algorithm, the first step is to estimate the optimal number of PMs (referred to as m) required to allocate VMs. Based on this

estimation, the VBP algorithm then deals with the solution of the mapping problem. The allocation process in VBP takes place in two different phases. The algorithm first divides VMs into two parts, large VMs, and small VMs, based on the β parameter where β is a real number. If a VM size is greater than β in at least one dimension, it is placed in the large group; otherwise it is placed in the small group. Large VMs are rounded to fall into a fixed number of classes. Rounded large VMs are packed with dynamic programming, and small VMs are packed with linear programming.

Algorithm 1 VBP Algorithm for d-dimensional VM Allocation.

Input VMs I
Output Minimum Number of PMs used to host VMs
1: Guess $m = OPT(I)$
2: **A.Rounding:**
3: Create rounding Q of large VMs of I
4: **B. Large VMs Allocation.**
5: B1. Guess PM configuration of $m + \lceil \frac{\epsilon m}{2} \rceil$ PMs of size $(1 + \epsilon, ..., 1 + \epsilon, 1)$
6: B2. Allocate VMs in Q into configuration M, or return to Guessing phase(phase 1)
7: **C. Small VMs Allocation.**
8: Allocate small VMs using assignment LP, or return to Guessing phase(phase 1).
9: Replace VMs size in Q by original ones(the VMs size before rounding.)

Let I be a set of VM requests. Each request is a vector (*CPU*, *RAM*, *BW*) that defines a VM request in each dimension. The algorithm takes ϵ and estimates the optimal value m. It either assigns VMs into at most $(1 + 2\epsilon)m + 1$ PMs or indicates that the estimation is wrong. The resource augmentation applies to the $(d - 1)$ dimension. These dimensions are called augmentable, and the other dimension is called non-augmentable. The last dimension is assumed to be non-augmentable, and the other $(d - 1)$ dimensions increments with ϵ. The value for d is assumed to be $d = 3$ in our problem.

4.2.2 Large VMs allocation

Rounding is done differently for augmentable and non-augmentable dimensions. Augmentable dimensions are rounded to multiples of α. where $\alpha = \frac{\epsilon^2}{2d^2}$ and the $d - th$ dimensions are rounded based on the linear grouping.

- Rounding of augmentable dimensions: Each large VM p_i is replaced with a VM $\hat{q}_i$ as follows:

$$\hat{q}_i^t = \begin{cases} \lceil \frac{p_i^t}{\alpha} \rceil & \text{if } t \in \{1, \ldots, (d-1)\}. \\ p_i^t, & t = d. \end{cases} \tag{9}$$

The original instance I is classified into classes $\{W^u | u \in \{1, \ldots, \lceil \frac{1}{\alpha} \rceil\}^{d-1}\}$ where $W^u = \{p_i | \hat{q}_i^{\,t} = u^t.\alpha, \forall\, t \in [d-1]\}$, creating $r_A = (\lceil \frac{1}{\alpha} \rceil)^{d-1}$ classes [6].

- Rounding of non-augmentable dimension: The last dimension is rounded with linear grouping for each W^u separately. This splits each W^u into $a = \lceil \frac{1}{\lambda} \rceil$ groups, where $\lambda = \frac{\epsilon\beta}{2d}$. We reduce the number of groups and increase the number of items in each group. In our method, the λ parameter is multiplied by 1000. This change made it possible to allocate a large number of VMs.

  VMs from W^u are sorted in non-ascending order based on the last dimension. Let $(p_1, \ldots, p_{h_u})$ be the sorted VMs where $h_u = |W^u|$. For each $e = \{1, \ldots, a-1\}$ class $W^{u,e}$ is defined having $b = \lceil \lambda h_u \rceil$ VMs as follows: $W^{u,e} = \{p_{(e-1)b+1}, \ldots, p_{eb}\}$. The last group $W^{u,a} = \{p_{(a-1)b+1}, \ldots, p_{h_u}\}$ can contain less than b elements. The first element in each group is the largest VM in that group and is named *round vector*. The final rounded instance Q is obtained by replacing each vector $p_i \in W^{u,e}$ with q_i, where

$$q_i^t = \hat{q}_i^{\,t}, for\, t \in [d-1],$$
$$q_i^d = max\{p^d | p \in W^{u,e}\}.$$

  So the dth dimension is rounded up to the dth dimension of the group's *round vector*, and other coordinates are rounded to multiples of α.

  The result of rounding large VMs is a vector $\{n_1, n_2, \ldots, n_h\}$ containing the number of elements in h different classes, where $V = \sum_{i=1}^{h} n_i$. This vector is the dynamic programming input.
- Assigning large VMs: Since Q has a fixed number of VM types r_L, there is only $r \leq (\frac{d}{\beta})^{r_L}$ possible configurations of a single PM. $M = (m_1, \ldots, m_r)$ is called a PM configuration, where m_j indicates the number of PMs for the assigning of type j.

  The result of dynamic programming for large VMs is the minimum number of PMs needed to allocate large VMs.

4.2.3 Small VMs allocation

Linear programming is used to allocate small VMs. PMs used by large VMs may have space for smaller VMs. Linear programming assigns small VMs to these PMs. If some VMs cannot be placed in the previously used PMs by large VMs, they will be placed in a new PM by the Next Fit heuristic algorithm.

The linear programming formulation is as follows. Let denote D to be the set of all small VMs in I and define $b_j^t = \sum_{q \in Q_i} q^t$ for each PM, B_j, is the occupied capacity of B_j in dimension t. $t \in [d]$ indicates each dimension.

$$\begin{aligned} &\text{maximize} && \sum_{j=1}^{m} b_j^l \,\forall l \in [d] \\ &\text{subject to} && \sum_{j=1}^{m} x_{ij} = 1, \forall p_i \in S \end{aligned} \tag{1}$$

$$\sum_{i=1}^{|S|} x_{ij} p_i^l \leq (1+\epsilon) - b_j^l,\ \forall j \in [m], l \in [d-1] \tag{2}$$

$$\sum_{i=1}^{|S|} x_{ij} p_i^l \leq 1 - b_j^l,\ \forall j \in [m], l \in [d] \tag{3}$$

$$x_{ij} \geq 0,\ \forall i, j \tag{4}$$

The objective function states that the capacity of the PMs should be filled with as many small VMs as possible. Constraint (1) denotes that every small VM has to be assigned to one PM. The expressions (2) and (3) are capacity constraints for all $d-1$ and d dimensions, respectively. The resulting integer elements of LP are assigned directly to the PMs. Additional PMs are used for other items.

4.2.4 PAVBP algotirhm

Algorithm 2, called PAVBP, shows an algorithm inspired by the algorithm 1 with more details. All phases are similar to the algorithm 1, but the goal is energy efficiency instead of the minimum number of PMs. In other words, the algorithm chooses PMs based on power consumption. PMs with lower power consumption have higher priority for hosting VMs.

In the algorithm, OPT_{power} means the best power to be achieved by assigning VMs to PMs. This value is equivalence with the minimum required power, min_{power}.

Algorithm 2 Algorithm for d-dimensional Power Aware VMP.

Input VMs I
Output Minimum Power for Assigning VMs

A.Rounding:
  Classify VMs to large and small VMs based on β, gives Q and D respectively
  Rounding and grouping large VMs, Q, gives $V = (n_1, n_2, \ldots, n_h)$
B. Assigning large VMs.
  B1. Determine the set of all h-tuples of $V = (n_1, n_2, \ldots, n_h)$ that can be assigned into a single PM which has the minimum power consumption, $(q_1, \ldots, q_h)$ where $OPT_{power}(q_1, \ldots, q_h) = \min_{power}$ and $0 \leq q_i \leq n_i$ for all i
B2. Calculate the recurrence $OPT_{Power}(n1_1, \ldots, n_h) = \min_{power} + \min_{power} OPT(n_1 - q_1, \ldots, n_h - q_h)$ for all $q \in Q$
C. Assigning small VMs.
  Assign small VMs using linear programming or the Next Fit heuristic
Replace VMs size in Q by original ones(VMs size before rounding)

5 Performance evaluation

This section presents the results of the study. Experiments are implemented using the CloudSim 3.0.3 simulator [29] and CPLEX library. The proposed approach was compared with several heuristic methods regarding power consumption, resource wastage, and difference ratio. All the simulation results are executed on a system equipped with a 3.10 GHz Intel Core i5 CPU and 4 GB RAM.

5.1 Experimental setup

We simulated a data center consisting of 1400 heterogeneous PMs and varying numbers of VMs 1052, 512, 256, and 128. Half of the PMs are HP ProLiant ML110 G4 (referred to as G4), and the other half are HP ProLiant ML110 G5 (referred to as G5). The characteristics of the PMs are illustrated in Table 2. This simulation assumes that both servers contain one core.

To evaluate our algorithm on different instances, we use three categories of Virtual Machines (VMs): Based VMs, Big VMs, and Small VMs. It must be mentioned that these categories have nothing to do with the categories in the proposed algorithm in which virtual machines are divided into two classes, large and small. They are just three instances of different virtual machine categories for evaluating our algorithm.

Each of these categories is the input of our algorithm in different executions. For example, for Big VMs, we take it as VMs set and divide it into two classes large VMs and small VMs. This process is also done for the other two categories, Based VMs and Small VMs. The size of instances in each VM category is presented on Table 3. We modeled the VM types according to the Amazon EC2 Instance types(referred to as ”Based VMs”), as shown in Table 3. Furthermore, the proposed method is evaluated on two other VM classes called Big VMs and Small VMs.

To evaluate our approach, we tested real-world workloads, Control PlanetLab workload [30] and Bitbrain’s workload [31], and WK100, a synthetic workload where all values are always 100% during execution. Workloads are dynamically assigned to VMs at runtime. In Cloudsim, once the workload is zero, the VM is permanently removed from the PM, even if the VM experiences a nonzero workload over the next few clocks. We modified this part so that after checking the workload of the deleted VM and finding it nonzero, the VM will be placed on an appropriate PM according to the proposed algorithm.

Table 3 VMs instances configurations

VMs Category	Size of VMs
Based VMs	2500, 2000, 1000, 500
Big VMs	2500, 2100, 1500, 800
Small VMs	2200, 1800, 800, 400

5.2 Evaluation results

Some concepts have been changed to explain the VBP issues in the cloud domain. We consider PMs and VMs instead of bins and items, respectively. We modified the VBP algorithm initially proposed in [6] to make it applicable to the cloud domain. Algorithm 1 shows this modified algorithm. We derived another approach called PAVBP from the algorithm that aims to select the PMs that consume the least power. Algorithm 2 shows the algorithm. Analyzes include comparisons of the results of this study with the following published work:

- PABFD: PABFD is proposed by Beloglozov[32] and performs the allocation of virtual machines with a BFD algorithm where the appropriate PM is selected based on their power consumption.
- GRVMP: GRVMP[10] selects VMs using a greedy randomization technique to allocate on a PM with the minimum resource wastage. The GRVMP results reported in this study are an average of 10 runs.
- AFED-EF: AFED-EF[11] assigns VMs based on a parameter called energy efficiency. Power-efficient PMs have higher priority when deploying VMs.

Three evaluation metrics are reported in the results: total power consumption, total resource consumption, and difference ratio.

- Total power consumption: This metric is obtained from the following equation:

$$P^{\text{tot}} = \sum_{j=1}^{m} B_j^{\text{power}} = \sum_{j=1}^{m} y_j * (B_j^{min} + (B_j^{\max} - B_j^{\min})) * u_j^{\text{cpu}}) \tag{5}$$

  where B_j^{min} is the power consumption of B_j in idle mode, B_j^{max} is the power consumption of B_j when it is full utilization and u_j^{cpu} is the normalized CPU utilization of B_j.
- Total resource wastage: This metric intends to maximize the resource utilization of PMs and establishes a load balancing within the resources of a PM [10]. To obtain this metric, two parameters are needed, R_j^t and u_j^t, which they can get from the following equations:

  This metric is obtained using the following equation:

$$\mathbb{R}^{\text{tot}} = \sum_{j=1}^{m} R_j^w = \frac{\sum_{t=1}^{d} |R_j^t - \min(R_j^t)| + \epsilon}{\sum_{t=1}^{d} u_j^t}$$

  where R_j^t is the remaining normalized resource of B_j in dimension t, $min(R_j^t)$ is the minimum remaining resource that is normalized within all dimensions of PM B_j and u_j^t is the normalized resource utilization of B_j along the t-th dimension [10]. Similarly, ϵ is a small positive real number, and the value is considered to be 0.0001.

- Difference Ratio: To observe the performance difference within methods, we report the following metric computing for each workload.

$$\text{Difference Ratio} = \frac{Power - minPower}{Power} * 100 \tag{6}$$

where *minPower* refers to the power with the minimum value among the methods, and Power is the power consumption of the method.

The total power consumption and total resource wastage for WK100 workload are illustrated in Fig. 3. When the workload consistently remains at 100%, our methods produce more accurate power consumption results. We try this workload because VMs never switch off, and this leads to evaluating our methods when the demands are in their maximum state; VMs request 100% of their demands. We test different scenarios with different VM categories and the number of VMs to test the algorithms.

The experimental results obtained with VBP and PAVBP are slightly lower than the comparison algorithm GRVBP in both total power consumption and total resource wastage and outperform AFED-EF and PABFD in all scenarios. This superiority can be explained by using a combination of the exact techniques of

Fig. 3 Total power consumption and resource wastage results for WK100 workload

dynamic programming and linear programming that form the core of the algorithm for finding the optimal mapping.

In the big VM category of the WK100 workload, the difference in total power consumption and resource wastage between VBP and PAVBP is noticeable. However, these values are the same for the ”based” and ”small VMs” category. VBP generally recommends minimizing the number of PMs and ignoring power consumption when allocating VMs. In contrast, PAVBP tries to choose the PM with the lowest power consumption to allocate VMs. PMs may have insufficient capacity after just hosting a VM from the ”big VMs” category (PMs can host at most one VM).

The algorithm packs large and small items separately. The first stage is devoted to packing large items. In the second phase, packing small items, there may be insufficient capacity in already active PMs. Therefore, the algorithm selects new PMs to host the remaining VMs. This action increases the number of active PMs. Considering this scenario, VBP will choose a PM from both types of PM listed in Table 2. It makes no difference, as both types of PM can be hosted on, at most, one ”big” VM. However, PAVBP considers total power consumption when allocating VMs. At first glance, PAVBP seems to pick the first PM type in Table 2. This is due to its lower power consumption compared to the second type. However, this algorithm has a general vision and chooses the second type of PM. This type consumes more power than the first type but can allocate more VMs. This PM type can host multiple VMs. Finally, at the end of the algorithm, fewer active PMs reduce overall power consumption. As a result, PAVBP has the best performance for the ”big VMs” category.

VBP and PAVBP behave similarly on the other two VM categories: the ”based” and ”small” VMs. VM size plays a vital role in these algorithms. After the first phase of the algorithm (packing of large items), the PMs may still have enough capacity for small items. Therefore, there is no need to use a new PM. In VBP, every PM attempts to host multiple VMs with a few active PMs, so the second PM type is chosen based on a larger capacity. This is also true for PAVBP, as we discussed earlier for the ”big VMs” category.

Among the algorithms compared, the GRVBP algorithm yielded sub-optimal outcomes, exhibiting values that closely approached those achieved by our proposed method. However, more difference between our methods and the GRVBP algorithm is observed in the ”small VMs” category. The reason for this difference is that we try to allocate as many VMs as possible to already activated PMs and activate as few PMs as possible. Therefore, this procedure can allocate smaller VMs better. GRVBP gives the second-best results among the compared algorithms, but the randomness of the algorithm prevents it from being deterministic, and this problem causes different outputs from run to run.

PAVBP demonstrates the lowest total resource consumption value for large virtual machines (VMs) in comparison with the other evaluated algorithms. This observation suggests that PAVBP effectively utilizes the active PMs capacity to its maximum potential. This value is higher for the VBP algorithm. This is due to the inefficient use of active PMs capacity. These algorithms demonstrate similar power consumption behavior for the ”base” and ”small” VM categories, leading to an equivalent total resource consumption.

For ”small” VMs, VBP and PAVBP show the lowest total resource consumption among the compared algorithms. This is because these algorithms allocate smaller VMs better than comparable algorithms and use PM capacity efficiently. The total power consumption and total resource consumption of two real-world workloads, PlanetLab and Bitbrains are illustrated in Figs. 4 and 5, respectively. These figures show that VBP and PAVBP have the lowest total power consumption and are partially differ from the second-best algorithm, GRVBP. This fact is due to the fluid nature of workloads, which do not always request 100% of their demands. Similarly, the VBP and PAVBP algorithms' total resource wastage values come first or second. Finally, the difference ratio for each workload is specified in Tables 4, 5, and 6, respectively. Either VBP or PBVBP has the lowest value for all workloads.

5.2.1 Time complexity

The time complexity of AFED-EF is $O(mn)$, m is the number of servers, and n is the number of VMs within a data center. The time complexity of the GRVMP algorithm is equivalent to $O(nmk + mlogm)$. k is a constant value, where $k << n$. m and n refer to the number of servers and VMs within a data center, respectively. The time complexity of the PABFD algorithm is equivalent to $O(nm)$. Here, m and n have similar definitions with GRVMP and AFED-EF algorithms. The time

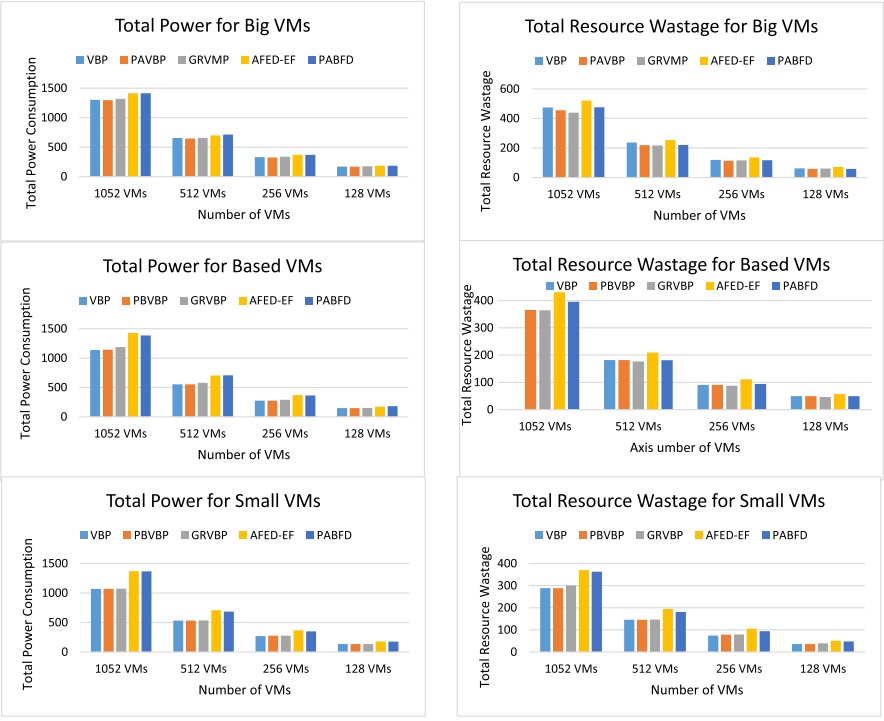

Fig. 4 Total power consumption and resource wastage results for control planetlab workload

Fig. 5 Total power consumption and resource wastage results for bitbrains workload

Table 4 Difference ratio for WK100 workload

	Big VMs				Based VMs				Small VMs			
VMs #/ Algorithm	1052	512	256	128	1052	512	256	128	1052	512	256	128
VBP	3%	5%	4%	5%	0%	0%	0%	0%	0%	0%	0%	0%
PAVBP	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
GRVMP	1%	1%	2%	1%	0%	1%	1%	2%	3%	4%	5%	6%
AFED-EF	17%	17%	17%	17%	28%	28%	28%	28%	40%	40%	39%	40%
PABFD	18%	19%	19%	19%	29%	30%	29%	30%	40%	40%	40%	40%

complexity of the proposed algorithm is $O((\frac{dn}{\beta})^{r_L}.m)$, where m is the number of PMs in optimal packing, d is the number of resource's dimensions (in our problem $d = 3$, CPU, RAM and Bandwidth) and *beta* used to classify items into large and small. Readers can refer to Table 1 for detailed information about notations.

Table 5 Difference ratio for control planetlab workload

VMs #/Algorithm	Big VMs				Based VMs				Small VMs			
	1052	512	256	128	1052	512	256	128	1052	512	256	128
VBP	1%	2%	1%	0%	0%	0%	0%	0%	0%	0%	0%	0%
PAVBP	0%	0%	0%	0%	1%	0%	0%	0%	0%	0%	2%	0%
GRVMP	2%	2%	3%	3%	4%	4%	5%	2%	0%	1%	2%	1%
AFED-EF	8%	8%	12%	8%	20%	21%	25%	13%	22%	25%	26%	25%
PABFD	8%	9%	12%	8%	18%	22%	24%	19%	22%	22%	23%	24%

Table 6 Difference ratio for bitbrains workload

VMs #/Algorithm	Big VMs				Based VMs				Small VMs			
	1052	512	256	128	1052	512	256	128	1052	512	256	128
VBP	0%	0%	1%	0%	0%	0%	0%	0%	0%	0%	0%	0%
PAVBP	1%	1%	0%	7%	0%	0%	0%	0%	0%	0%	2%	0%
GRVMP	3%	3%	0%	9%	1%	3%	5%	4%	2%	0%	4%	3%
AFED-EF	6%	7%	0%	2%	11%	17%	16%	9%	17%	7%	15%	16%
PABFD	8%	7%	5%	5%	12%	16%	19%	10%	15%	6%	14%	10%

6 Conclusions and future work

In this paper, we addressed the problem of power-efficient VM placement and the bin-packing problem, focusing on BPLUC (Bin Packing with Linear Usage Cost), a variant of bin packing that considers fixed and variable costs. We showed that an algorithm for BP with an approximation ratio of α can be used as an approximation algorithm for BPLUC with an approximation ratio based on α for homogeneous hosts. We also extended this result to VBP(Vector Bin Packing) and VBPLUC(Vector Bin Packing with Linear Usage Cost), which deal with items and bins with multiple dimensions.

To solve the VMP(Virtual Machine Placement) problem in a heterogeneous cloud data center, we modified and implemented the VBP algorithm proposed in [6] to minimize power consumption. In our analysis, we treated virtual machines (VMs) and physical machines (PMs) as multidimensional items, taking into account their respective dimensions of CPU, RAM, and bandwidth. Our experimental results demonstrate that the proposed algorithm outperforms existing methods, especially when there is a significant difference between the sizes of VMs and hosts.

In our future work, we aim to expand the application of the power-based vector bin packing algorithm to online environments. This entails developing a framework that can effectively allocate suitable physical machines (PMs) for newly added virtual machines (VMs) in real-time, as well as during the migration process. We also aim to expand the algorithm to handle multiple resources in each dimension, such as a CPU with four cores and two memory units. This extension will enable

the algorithm to be applied to a broader variety of VM and PM types, making it more useful in cloud environments. Additionally, we will investigate the relationship between approximation methods of BP and BPLUC and their applicability to VMP with heterogeneous hosts.

References

1. Askarizade Haghighi M, Maeen M, Haghparast M (2019) An energy-efficient dynamic resource management approach based on clustering and meta-heuristic algorithms in cloud computing iaas platforms: Energy efficient dynamic cloud resource management. Wirel Pers Commun 104:1367–1391
2. Beloglazov A (2013) Energy-efficient management of virtual machines in data centers for cloud computing. PhD thesis
3. Jennings B, Stadler R (2015) Resource management in clouds: survey and research challenges. J Netw Syst Manage 23(3):567–619
4. Martello S, Toth P (1990) Bin-packing problem. Knapsack problems: algorithms and computer implementations, pp. 221–245
5. Cambazard H, Mehta D, O'Sullivan B, Simonis H (2013) Bin packing with linear usage costs–an application to energy management in data centres. In: International Conference on Principles and Practice of Constraint Programming, Springer, pp. 47–62
6. Bansal N, Eliáš, M, Khan A (2016) Improved approximation for vector bin packing. In: *Proceedings of the twenty-seventh annual ACM-SIAM symposium on discrete algorithms*, pp. 1561–1579. SIAM
7. Wei C, Zhi-Hua H, Wang Y-G (2020) Exact algorithms for energy-efficient virtual machine placement in data centers. Futur Gener Comput Syst 106:77–91
8. Zoltán Ádám Mann (2016) Multicore-aware virtual machine placement in cloud data centers. IEEE Trans Comput 65(11):3357–3369
9. Chen H, Wen Y, Wang Y (2023) An energy-efficient method of resource allocation based on request prediction in multiple cloud data centers. Concurr Comput Pract Exp 35(9):e7636
10. Azizi S, Shojafar M, Abawajy J, Buyya R (2020) Grvmp: a greedy randomized algorithm for virtual machine placement in cloud data centers. IEEE Syst J 15(2):2571–2582
11. Zhou Z, Shojafar M, Alazab M, Abawajy J, Li F (2021) Afed-ef: An energy-efficient VM allocation algorithm for IoT applications in a cloud data center. IEEE Trans Green Commun Netw 5(2):658–669
12. Beloglazov A, Buyya R (2012) Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr Comput Pract Exp 24(13):1397–1420
13. Ajmera K, Tewari TK (2018) Greening the cloud through power-aware virtual machine allocation. In: 11th International Conference on Contemporary Computing (IC3), pp. 1–6. IEEE
14. Jangiti S, Ram ES, Sriram VSS (2019) Aggregated rank in first-fit-decreasing for green cloud computing. In: Cognitive informatics and soft computing, pp. 545–555. Springer
15. Sunil S, Patel S (2023) Energy-efficient virtual machine placement algorithm based on power usage. Computing, pp. 1–25
16. Zhou J, Zhang Y, Sun L, Zhuang S, Tang C, Sun J (2019) Stochastic virtual machine placement for cloud data centers under resource requirement variations. IEEE Access 7:174412–174424
17. Zhang X, Tingming W, Chen M, Wei T, Zhou J, Shiyan H, Buyya R (2019) Energy-aware virtual machine allocation for cloud with resource reservation. J Syst Softw 147:147–161
18. Ding Z, Tian Y-C, Wang Y-G, Zhang W-Z, Zu-Guo Yu (2023) Accelerated computation of the genetic algorithm for energy-efficient virtual machine placement in data centers. Neural Comput Appl 35(7):5421–5436
19. Alharbi F, Tian Y-C, Tang M, Zhang W-Z, Peng C, Fei M (2019) An ant colony system for energy-efficient dynamic virtual machine placement in data centers. Expert Syst Appl 120:228–238
20. Singh AK, Swain SR, Lee CN (2023) A metaheuristic virtual machine placement framework toward power efficiency of sustainable cloud environment. Soft Comput 27(7):3817–3828

21. Xiao Z, Jiang J, Zhu Y, Ming Z, Zhong S, Cai S (2015) A solution of dynamic VMS placement problem for energy consumption optimization based on evolutionary game theory. J Syst Softw 101:260–272
22. Gupta MK, Amgoth T (2018) Resource-aware virtual machine placement algorithm for iaas cloud. J Supercomput 74(1):122–140
23. Shaw R, Howley E, Barrett E (2019) An energy efficient anti-correlated virtual machine placement algorithm using resource usage predictions. Simul Model Pract Theory 93:322–342
24. Wang W, Jiang Y, Weiwei W (2016) Multiagent-based resource allocation for energy minimization in cloud computing systems. IEEE Trans Syst Man Cybern Syst 47(2):205–220
25. Shirvani MH (2023) An energy-efficient topology-aware virtual machine placement in cloud datacenters: a multi-objective discrete Jaya optimization. Sustain Comput Inf Syst 38:100856
26. Cambazard H, Mehta D, O'Sullivan B, Simonis H (2015) Bin packing with linear usage costs. arXiv preprint arXiv:1509.06712
27. Pietri I, Sakellariou R (2016) Mapping virtual machines onto physical machines in cloud computing: a survey. ACM Comput Surv (CSUR) 49(3):1–30
28. SPEC Power characteristics for servers (2008) https://www.spec.org/power/. [Online; Accessed 15 Apr 2020]
29. Buyya R, Calheiros RN, Beloglazov A (2009) Cloudsim: a framework for modeling and simulation of cloud computing infrastructures and services. The cloud computing and distributed systems (CLOUDS) Laboratory.[Online].[Accessed 18 May 2018]
30. Peterson L, Bavier A, Fiuczynski ME, Muir S (2006) Experiences building planetlab. In: *Proceedings of the 7th symposium on operating systems design and implementation*, pp. 351–366
31. Shen S, van Beek V, Iosup A (2015) Statistical characterization of business-critical workloads hosted in cloud datacenters. In: *2015 15th IEEE/ACM international symposium on cluster, cloud and grid computing*, pp. 465–474. IEEE
32. Beloglazov A, Abawajy J, Buyya R (2012) Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener Comput Syst 28(5):755–768