






Article

A Multi-Objective Approach for Optimizing Virtual Machine Placement Using ILP and Tabu Search

Mohamed Koubàa ^{1,*}, Rym Regaieg ^{1,†}, Abdullah S. Karar ^{2,†}, Muhammad Nadeem ^{2,†} and Faouzi Bahloul ^{3,†}

¹ SYSCOM Laboratory, National Engineering School of Tunis (ENIT), University of Tunis El Manar, Tunis 1002, Tunisia; rym.regaieg@enit.utm.tn

² College of Engineering and Technology, American University of the Middle East, Egaila 15453, Kuwait; abdullah.karar@aum.edu.kw (A.S.K.); muhammad.nadeem@aum.edu.kw (M.N.)

³ SERCOM Laboratory, Tunisia Polytechnic School (EPT), University of Carthage, Tunis 2078, Tunisia; faouzi.bahloul@enit.utm.tn

* Correspondence: mohamed.koubaa@enit.utm.tn

† These authors contributed equally to this work.

Abstract: Efficient Virtual Machine (VM) placement is a critical challenge in optimizing resource utilization in cloud data centers. This paper explores both exact and approximate methods to address this problem. We begin by presenting an exact solution based on a Multi-Objective Integer Linear Programming (MOILP) model, which provides an optimal VM Placement (VMP) strategy. Given the NP-completeness of the MOILP model when handling large-scale problems, we then propose an approximate solution using a Tabu Search (TS) algorithm. The TS algorithm is designed as a practical alternative for addressing these complex scenarios. A key innovation of our approach is the simultaneous optimization of three performance metrics: the number of accepted VMs, resource wastage, and power consumption. To the best of our knowledge, this is the first application of a TS algorithm in the context of VMP. Furthermore, these three performance metrics are jointly optimized to ensure operational efficiency (OPEF) and minimal operational expenditure (OPEX). We rigorously evaluate the performance of the TS algorithm through extensive simulation scenarios and compare its results with those of the MOILP model, enabling us to assess the quality of the approximate solution relative to the optimal one. Additionally, we benchmark our approach against existing methods in the literature to emphasize its advantages. Our findings demonstrate that the TS algorithm strikes an effective balance between efficiency and practicality, making it a robust solution for VMP in cloud environments. The TS algorithm outperforms the other algorithms considered in the simulations, achieving a gain of 2% to 32% in OPEF, with a worst-case increase of up to 6% in OPEX.

Keywords: cloud computing; virtualization; virtual machines; placement problem; multi-objective optimization; integer linear programming; tabu search; ant colony optimization; genetic algorithm; first fit; first fit decreasing; worst fit



Citation: Koubàa, M.; Regaieg, R.; Karar, A.S.; Nadeem, M.; Bahloul, F. A Multi-Objective Approach for Optimizing Virtual Machine Placement Using ILP and Tabu Search. *Telecom* **2024**, *5*, 1309–1331. <https://doi.org/10.3390/telecom5040065>

Received: 30 October 2024

Revised: 5 December 2024

Accepted: 9 December 2024

Published: 16 December 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Cloud computing has transformed the IT landscape, providing scalable, flexible, and cost-efficient solutions for handling large volumes of data and applications across industries such as healthcare, education, and e-commerce [1,2]. A key enabler of this transformation is virtualization, which facilitates the creation of Virtual Machines (VMs) to optimize resource utilization in data centers (DCs) [3]. However, achieving this optimization hinges on addressing the Virtual Machine Placement Problem (VMPP).

The VMPP involves allocating VMs with specific resource requirements (e.g., CPU, memory, and storage) to Physical Machines (PMs) in a way that maximizes efficiency while meeting application demands. Effective placement enhances resource utilization, system performance, and compliance with operational constraints. Conversely, poor placement

can result in wasted resources, reduced performance, and violations of Service Level Agreements (SLAs), underscoring the importance of robust solutions.

The complexity of the VMPP stems from multiple interrelated factors [4]. Resource heterogeneity in modern cloud environments complicates the matching of VM requirements to available PMs. The sheer scale of DCs introduces a combinatorial challenge as the number of potential VM–PM allocations grows exponentially with infrastructure size. Additionally, balancing conflicting objectives—such as minimizing energy consumption, reducing costs, ensuring Quality of Service (QoS), and adhering to SLAs—requires strategic prioritization. Dynamic workloads and fluctuating resource demands further amplify these challenges, necessitating adaptable and real-time solutions.

These factors significantly impact cloud operational efficiency (OPEF) and sustainability [5]. High energy consumption increases costs and contributes to carbon emissions, making energy-efficient strategies vital for reducing environmental impact. Effective resource utilization is critical to avoiding under-utilization, which wastes resources, and over-utilization, which causes performance bottlenecks and degrades QoS. SLA compliance is also essential to avoid penalties and maintain client satisfaction, requiring careful management of latency and workload distribution. Finally, scalability and adaptability are paramount for accommodating infrastructure growth and dynamic workloads, ensuring reliable performance under changing conditions.

Traditional methods, such as greedy algorithms or single-objective optimization techniques, often fall short in addressing the multifaceted nature of the VMPP. These methods typically focus on optimizing a single objective (e.g., minimizing energy consumption or maximizing resource utilization), neglecting the interdependencies and trade-offs that exist between competing objectives in real-world cloud environments. For instance, while reducing energy consumption is crucial for sustainability and cost savings, it may inadvertently lead to performance degradation if not balanced with latency or QoS requirements. Moreover, traditional approaches struggle to handle the dynamic and heterogeneous nature of modern cloud applications, such as cloud-native applications and serverless computing, which demand highly adaptive and resource-efficient placements. These challenges underscore the limitations of existing methods and highlight the need for more sophisticated, multi-objective strategies.

In this context, this study introduces and compares two innovative methods that tackle the VMPP by considering multiple conflicting objectives simultaneously:

- The first approach is a Multi-Objective Integer Linear Programming (MOILP) model that optimizes three key objectives: maximizing the number of hosted VMs, minimizing resource wastage, and reducing power consumption. This model integrates objectives that are crucial for cloud data center operators—increasing revenue, enhancing client satisfaction, and reducing operational expenditures (OPEX). By considering these objectives together, the MOILP approach offers a novel framework that seeks to strike a balance between OPEF and cost-effectiveness.
- The second method explored is a Tabu Search (TS) algorithm, which offers a meta-heuristic solution to the same multi-objective problem. Unlike traditional heuristics, the TS algorithm is designed to optimize multiple objectives simultaneously, providing a practical solution to the computational challenges associated with the MOILP model. The TS algorithm excels in navigating large-scale problem spaces, offering high-quality solutions within reasonable computation times, making it particularly suitable for real-time cloud management scenarios.

The scientific novelty of this work lies in the development and comparison of two distinct yet complementary methods—a precise optimization model and a meta-heuristic approach—both tailored to solve the complex, multi-objective nature of the VMPP. This dual approach contributes significantly to the current literature by addressing the shortcomings of traditional methods and offering practical, scalable solutions for optimizing VM Placement (VMP) in modern cloud data centers. The findings from this study provide

a solid foundation for further research and hold great potential for improving OPEF in cloud computing infrastructures.

The structure of this paper is as follows: Section 2 provides a comprehensive description of the problem addressed in this study and a review of the current state of the art, highlighting relevant research and methodologies. Section 3 presents the proposed solutions, offering in-depth insights into their development and implementation. Section 4 discusses the simulation results, evaluating the performance of the proposed approaches. Finally, Section 5 summarizes the main findings and suggests potential avenues for future research.

2. Problem Description and Related Work

2.1. Problem Overview

The VMPP, also called the VM mapping problem, involves assigning a set of VMs to a set of PMs while satisfying certain constraints and optimizing an objective function as previously discussed. Specifically, given a set of PMs with defined resource capacities (CPU, memory, and storage) and a set of VMs with resource requirements, the goal is to allocate physical resources such that VM requirements are met, constraints are satisfied, and the objective function is optimized. For example, we consider three PMs with specified capacities for CPU cores, memory, and storage (Table 1), and five VMs with specific resource requirements (Table 2). The task is to allocate the VMs across the PMs to maximize the number of hosted VMs, ensure a one-to-one VM-PM assignment, and avoid resource overcommitment on any PM.

Table 1. PM resource characteristics.

PM	CPU (Cores)	RAM (GB)	Storage (GB)
PM1	8	32	500
PM2	16	64	1000
PM3	4	16	250

Table 2. VM resource requirements.

VM	CPU (Cores)	RAM (GB)	Storage (GB)
VM1	4	16	100
VM2	2	8	50
VM3	6	24	200
VM4	4	16	100
VM5	2	8	50

As shown in Figure 1, multiple solutions exist for the VMPP, each with its own set of advantages and trade-offs. In the first solution, depicted in Figure 1a, VM1 and VM2 are hosted on PM1, utilizing six CPU cores, 24 GB of RAM, and 150 GB of storage. VM3 and VM5 are placed on PM2, consuming eight CPU cores, 32 GB of RAM, and 250 GB of storage, while VM4 is placed on PM3, fully utilizing its resources. Although this configuration hosts all VMs, it results in higher energy consumption since all three PMs are active.

Alternatively, the second solution, shown in Figure 1b, optimizes resource utilization and minimizes energy consumption. In this solution, VM1, VM2, and VM5 are allocated to PM1, fully utilizing its resources (eight CPU cores, 32 GB of RAM, and 200 GB of storage). VM3 and VM4 are placed on PM2, consuming 10 CPU cores, 40 GB of RAM, and 300 GB of storage. Notably, PM3 remains unused, reducing energy consumption by activating only PM1 and PM2. This configuration maximizes the number of accepted VMs while minimizing resource waste and energy expenditure, demonstrating the effectiveness of strategic VM placement in optimizing resource utilization and cost-efficiency in cloud environments.

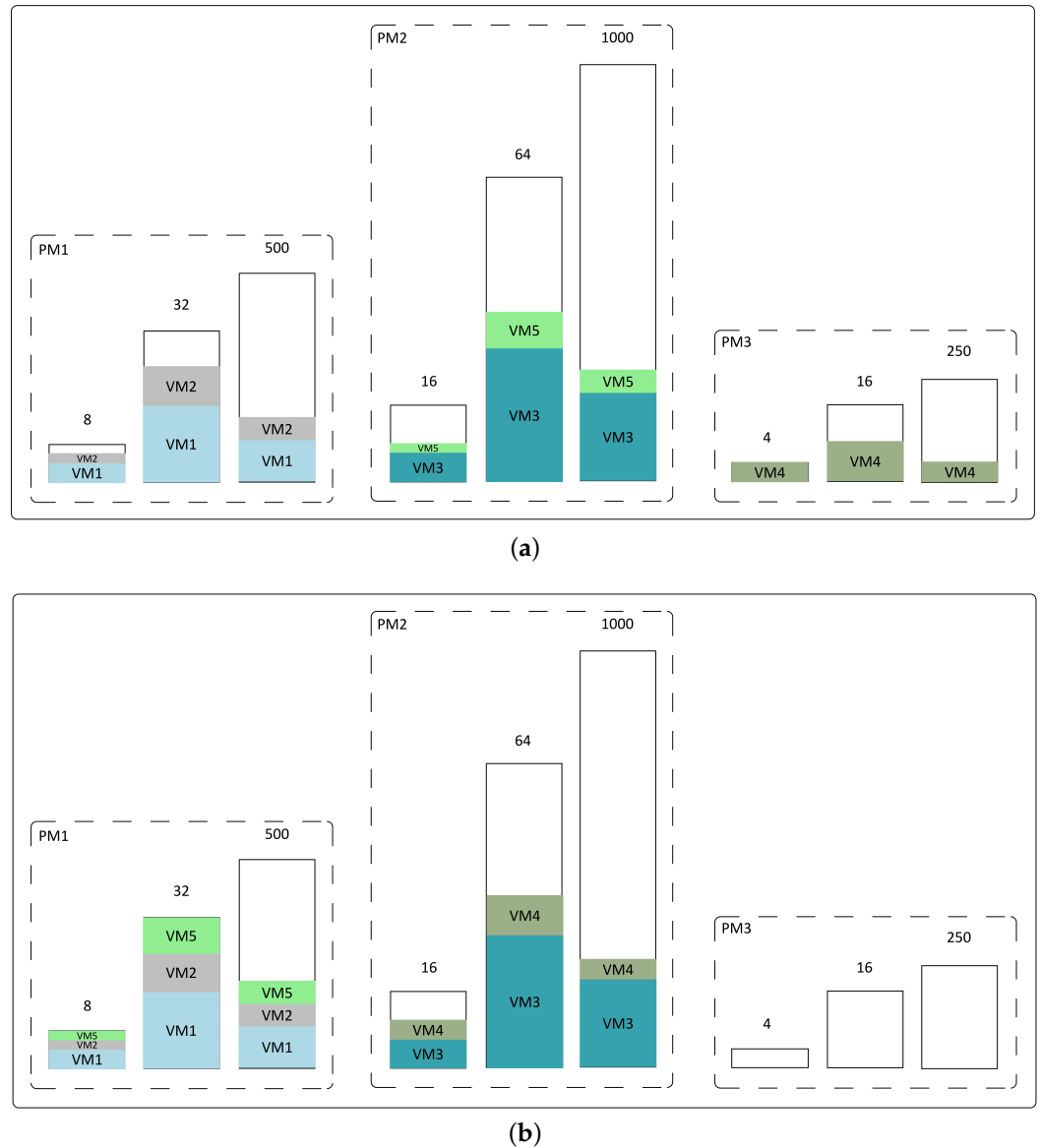


Figure 1. Comparative analysis of VMP solutions: solution 1 vs. solution 2. (a) VMP solution 1; (b) VMP solution 2.

This example highlights the variability in VMPP solutions and emphasizes the need to optimize multiple criteria simultaneously. A comprehensive approach that sequentially optimizes several objective functions is essential for achieving more efficient and effective VMP solutions.

2.2. Objectives and Challenges

The VMP is guided by several objectives, focusing on maximizing resource utilization across CPU, memory, and storage [6–11]. Minimizing energy consumption is another key objective, achievable through intelligent server management that reduces the number of active servers or by migrating VMs to energy-efficient servers [12–15]. Load balancing is also important to prevent resource bottlenecks and improve system reliability [16–19]. Additionally, meeting QoS requirements, as outlined in SLAs, ensures that performance levels are met [20–22]. Other factors include scalability, which ensures that the system can adapt to changing demands, and minimizing migration costs when relocating VMs [23–27]. These objectives are interrelated, requiring a careful balance to achieve optimal performance, cost efficiency, and sustainability in cloud environments.

2.3. Multi-Objective VMP Optimization

Multi-objective optimization (MOO) offers a framework for solving the VMPP by evaluating competing objectives simultaneously. Techniques such as genetic algorithms (GAs), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) have been applied to VMPP, each with strengths and limitations.

GAs are commonly used for multi-objective VMP, evolving solution populations over generations [28–32]. However, GAs may suffer from slow convergence, especially in large solution spaces, and face challenges with maintaining population diversity, leading to premature convergence. PSO, inspired by social behaviors in nature, has been used to minimize energy consumption and optimize resource utilization [33,34]. However, PSO can become trapped in local optima and is sensitive to parameter tuning. ACO, which simulates ant foraging behavior, has also been applied to VMPP, using pheromone trails to guide efficient placements [35–37]. However, ACO may struggle with balancing exploration and exploitation, leading to significant computational overhead in large environments. Hybrid approaches combining GAs with local search methods or PSO with heuristics have also been explored to overcome these challenges [30,38,39]. While these hybrid methods can improve solution quality, they may introduce additional complexity in implementation and tuning, and may struggle with real-time adaptation to dynamic environments.

3. Proposed Methods

This section presents the proposed methods for solving the VMPP, focusing on their contributions to addressing the limitations of existing approaches.

We introduce a novel solution by combining two advanced methods: a Multi-Objective Integer Linear Programming (MOILP) model and a Multi-Objective Tabu Search (MOTS) meta-heuristic. To the best of our knowledge, MOTS is applied for the first time to address the VMPP.

The key innovation of this study lies in addressing the limitations of MOILP in large-scale cloud environments. While MOILP provides a precise mathematical framework for optimizing conflicting objectives, its computational cost and inefficiency grow exponentially with the size of the problem, especially with thousands of VMs and PMs. As the complexity of MOILP increases, it struggles with long solution times and high resource demands, making it unsuitable for real-time or dynamic cloud scenarios.

To address these challenges, we employ MOTS, a TS-based optimization technique that is highly effective in solving complex problems. MOTS improves local search processes by utilizing a memory-based approach, which helps it avoid getting trapped in local optima. The method explores the solution space methodically, permitting moves that may initially worsen the objective, and uses a tabu list—a short-term memory mechanism—to prevent revisiting recently explored solutions. This approach ensures greater diversity and directs the search toward high-quality solutions [40,41]. MOTS improves computational efficiency, accelerates convergence, and enhances scalability, making it particularly suited for dynamic environments with larger solution spaces.

The combination of MOILP and MOTS provides a unique balance of theoretical rigor and practical scalability, making it an ideal solution for large-scale cloud environments.

The study simultaneously optimizes three critical objectives to improve VMP:

- Maximizing hosted VMs: This objective aims to optimize PM utilization, increasing cloud profitability by hosting more clients.
- Minimizing resource wastage: Efficient allocation of CPU, memory, and storage prevents over-provisioning and enhances server performance. Unbalanced residual resources can limit future VM placements. Resource wastage is calculated as the sum of normalized residuals for each dimension, with greater discrepancies leading to more waste [42]. For instance, a server with unused RAM but insufficient CPU cannot accommodate more VMs. Figure 2 shows the VMP for three VMs over a PM. The total CPU and RAM capacities are represented among the x-axis and y-axis respectively. Placing three VMs on the host reduces its resource capacity across the two dimensions,

with each small rectangle representing the resources allocated to a VM. The dark gray area indicates the remaining residual resources available for future allocation. In the example of Figure 2a, the host has substantial unused RAM capacity but limited CPU, which prevents it from accommodating any additional VMs due to CPU shortage. Figure 2b shows a VMP scenario where CPU and memory utilization reaches 90%, indicating that all dimensions are balanced, and the PM is being fully utilized.

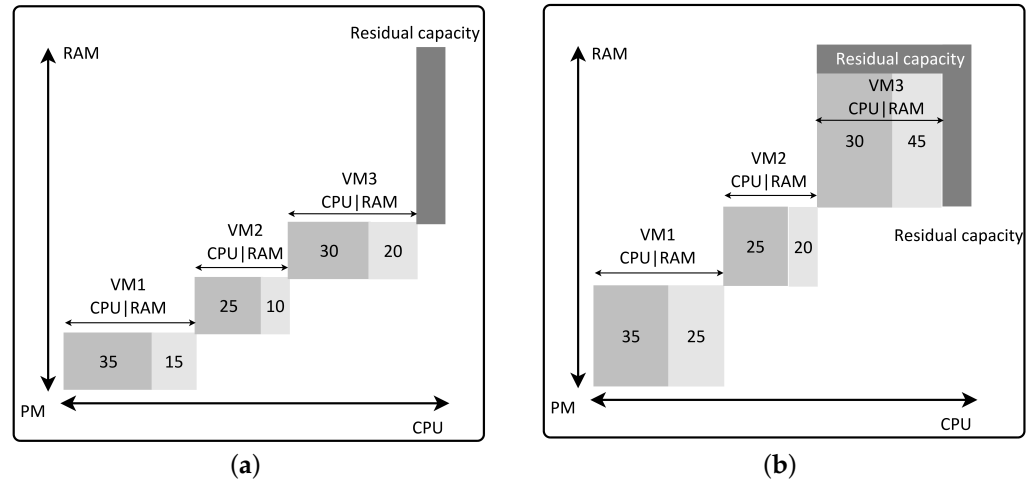


Figure 2. An example of a VMP of three VMs over one PM. (a) A first example of a VMP of three VMs over one PM; (b) A second example of a VMP of three VMs over one PM.

- **Minimizing energy consumption:** Reducing power usage lowers operational costs and environmental impact. Energy consumption primarily depends on CPU usage, calculated using the formula [35]

$$\mathcal{E}_j = (\mathcal{E}_j^1 - \mathcal{E}_j^0) \times U_j + \mathcal{E}_j^0 \tag{1}$$

where U_j is the normalized CPU usage on PM P_j ($U_j \in [0, 1]$), and \mathcal{E}_j^1 and \mathcal{E}_j^0 are the power values when the PM is busy or idle, respectively. We assume that CPU consumes the most energy compared to other resources like memory and storage, though the model can be extended to include other parameters.

Balancing these objectives enables effective trade-offs between performance, cost, and energy efficiency in cloud environments.

3.1. Mathematical Formulation of the Multi-Objective Integer Linear Programming Model

In the realm of optimization, MOILP models play a crucial role in addressing complex problems where multiple, often conflicting, objectives need to be simultaneously optimized. This section delves into the detailed description of an MOILP model designed to tackle the VMPP.

3.1.1. Notations

We use the following notations and formatting conventions.

Index Notations

The indices i and j refer to a VM request and a PM, respectively.

Parameters

The parameters used in the model capture the core characteristics of the VMs and PMs in the DC.

- $V = \{v_1, v_2, \dots, v_N\}$ represents the set of VM requests received by the DC. Each VM v_i is defined by

$$v_i = \{c_i, r_i, s_i\}$$

where

- c_i represents the CPU requirements of VM v_i ;
- r_i represents the RAM requirements of VM v_i ;
- s_i represents the storage requirements of VM v_i .
- $P = \{P_1, P_2, \dots, P_M\}$ represents the set of available PMs in the DC. Each PM P_j is characterized by

$$P_j = \{C_j, R_j, S_j, \mathcal{E}_j^0, \mathcal{E}_j^1\}$$

where

- C_j represents the initial CPU capacity of P_j ;
- R_j represents the initial memory capacity of P_j ;
- S_j represents the initial storage capacity of P_j ;
- \mathcal{E}_j^0 represents the average power consumed by P_j when the PM is idle.;
- \mathcal{E}_j^1 represents the average power consumed by P_j when the PM is busy.
- $N = |V|$ and $M = |P|$ are the number of VM requests and the number of PMs available in the DC, respectively.

Variables

To define the decision-making aspect of the model, we introduce the following binary variables:

- The binary variable ϕ_{ij} , equal to 1 if VM v_i is hosted by PM P_j and 0 otherwise.
- The binary variable κ_j , equal to 1 if there is at least one VM hosted by PM P_j and 0 otherwise.

Table 3 presents an overview of the notations employed in the MOILP model, serving as a convenient reference for each parameter and its role in the optimization framework.

Table 3. Notations used in the MOIP model.

Symbol	Description
V	Set of VM requests.
N	Number of VM requests.
P	Set of available PMs in the DC.
M	Number of PMs.
c_i	CPU requirements of VM v_i .
r_i	RAM requirements of VM v_i .
s_i	Storage requirements of VM v_i .
C_j	CPU capacity of PM P_j .
R_j	RAM capacity of PM P_j .
S_j	Storage capacity of PM P_j .
\mathcal{E}_j^0	Power consumption of PM P_j when idle.
\mathcal{E}_j^1	Power consumption of PM P_j when busy.

Table 3. Cont.

Symbol	Description
ϕ_{ij}	Binary decision variable, where $\phi_{ij} = 1$ if VM v_i is hosted by PM P_j , and $\phi_{ij} = 0$ otherwise.
κ_j	Binary decision variable, where $\kappa_j = 1$ if at least one VM is hosted by PM P_j , $\kappa_j = 0$ otherwise.

3.1.2. Mathematical Formulation

The mathematical formulation of the MOILP model is structured in three sequential stages. Each stage focuses on one key objective, ensuring that prior objectives are preserved:

- Objective 1 (O1): Maximize the number of accepted VMs.
- Objective 2 (O2): Minimize resource wastage in the DC.
- Objective 3 (O3): Minimize power consumption in the DC.

The stages are optimized in a lexicographical order, prioritizing O1 over O2, and O2 over O3. This ensures that the results of one stage are not compromised by subsequent stages. Figure 3 visually represents this three-stage optimization approach.

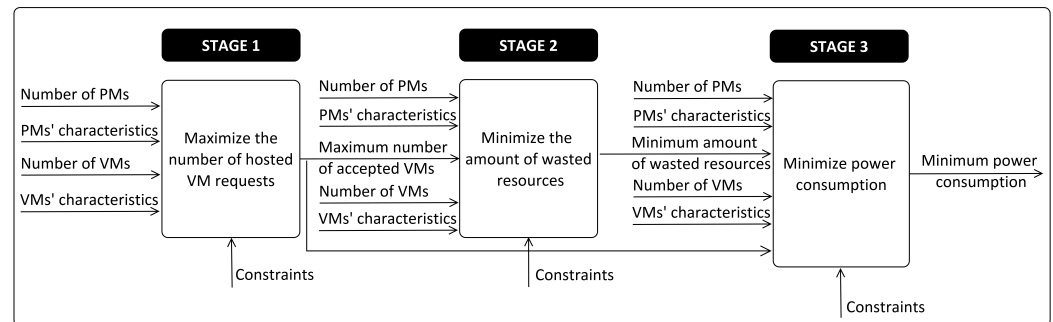


Figure 3. Graphical representation of the three-stage MOILP solution.

Stage 1: Maximizing the Number of Accepted VMs—O1

This stage focuses on maximizing the total number of hosted VM requests, denoted as ϕ_{max} , by determining the optimal placement of VMs across PMs.

Objective:

$$\text{Maximize } \phi_{max} = \sum_{i=1}^N \sum_{j=1}^M \phi_{ij} \quad (2)$$

Subject to the following constraints:

- Single host constraint: Each VM request v_i is assigned to, at most, one PM:

$$\sum_{j=1}^M \phi_{ij} \leq 1, \quad \forall 1 \leq i \leq N \quad (3)$$

- CPU capacity constraint: The total CPU usage of the VMs assigned to PM P_j must not exceed the PM's available CPU capacity C_j :

$$\sum_{i=1}^N c_i \phi_{ij} \leq C_j, \quad \forall 1 \leq j \leq M \quad (4)$$

- RAM capacity constraint: The total RAM usage of the VMs assigned to PM P_j must not exceed the PM's available RAM capacity R_j :

$$\sum_{i=1}^N r_i \phi_{ij} \leq R_j, \quad \forall 1 \leq j \leq M \quad (5)$$

- Storage capacity constraint: The storage usage of the VMs assigned to PM P_j must not exceed the PM's available storage capacity S_j :

$$\sum_{i=1}^N s_i \phi_{ij} \leq S_j, \quad \forall 1 \leq j \leq M \quad (6)$$

Stage 2: Minimizing Resource Wastage—O2

Once ϕ_{max} , the maximum number of hosted VMs, is determined, the next objective is to minimize resource wastage, denoted as W_{min} . This stage ensures that resources are used efficiently across all PMs while maintaining the number of hosted VMs unchanged from Stage 1.

Objective:

$$\text{Minimize } W_{min} = \sum_{j=1}^M \left(3 - \left(\sum_{i=1}^N \left(\frac{c_i \phi_{ij}}{C_j} + \frac{r_i \phi_{ij}}{R_j} + \frac{s_i \phi_{ij}}{S_j} \right) \right) \right) - 3 \left(M - \sum_{j=1}^M \kappa_j \right) \quad (7)$$

Subject to the following constraints:

- VM hosting preservation constraint: The total number of hosted VMs must be at least equal to ϕ_{max} , computed in Stage 1:

$$\phi_{max} \leq \sum_{i=1}^N \sum_{j=1}^M \phi_{ij} \quad (8)$$

- VM assignment linking constraint: A VM can only be assigned to a PM if that PM is active ($\kappa_j = 1$):

$$\phi_{ij} \leq \kappa_j, \quad \forall 1 \leq i \leq N, \forall 1 \leq j \leq M \quad (9)$$

- PM usage indicator constraint: A PM is considered active ($\kappa_j = 1$) only if it hosts at least one VM:

$$\kappa_j \leq \sum_{i=1}^N \phi_{ij}, \quad \forall 1 \leq j \leq M \quad (10)$$

and (3)–(6).

Stage 3: Minimizing Power Consumption—O3

The final stage minimizes the total power consumption, denoted as \mathcal{E}_{min} . This includes the static power consumption of active PMs and the dynamic power usage associated with hosted VMs.

Objective:

$$\text{Minimize } \mathcal{E}_{min} = \sum_{j=1}^M \left((\mathcal{E}_j^1 - \mathcal{E}_j^0) \sum_{i=1}^N \left(\frac{c_i \phi_{ij}}{C_j} \right) + \mathcal{E}_j^0 \kappa_j \right) \quad (11)$$

Subject to the following constraints:

- Resource wastage limitation constraint: Ensure that total resource wastage does not exceed the minimum value, W_{min} , determined in Stage 2:

$$\sum_{j=1}^M \left(3 - \left(\sum_{i=1}^N \left(\frac{c_i \phi_{ij}}{C_j} + \frac{r_i \phi_{ij}}{R_j} + \frac{s_i \phi_{ij}}{S_j} \right) \right) \right) - 3 \left(M - \sum_{j=1}^M \kappa_j \right) \leq W_{min} \quad (12)$$

and (3)–(6) and (8)–(10).

Computational Complexity Analysis

In Stage 1, the number of variables is $N \times M$. The number of constraints is $N + 3M$, giving a computational complexity of $O(N \times M)$. In Stage 2, the same number of variables $N \times M$ is used, with an additional M binary variables (κ_j), and the number of constraints remains $N + 3M$, with the complexity still $O(N \times M)$. In Stage 3, the model retains the $N \times M$ variables and the additional M binary variables (κ_j), with the total number of constraints remaining $N + 3M$, and the complexity is, again, $O(N \times M)$. Overall, the entire model involves $N \times M$ variables plus M additional binary variables across all steps. The total number of constraints across all steps is $N + 3M$. Thus, the model's computational complexity across all steps is $O(N \times M)$.

3.2. Multi-Objective Tabu Search

TS is a meta-heuristic optimization method specifically designed to address combinatorial problems. The algorithm systematically explores the solution space until it satisfies a predetermined stopping condition, such as reaching a specified number of iterations or achieving a target cost value. The process starts with an initial solution, which may be produced using another approach (e.g., randomly when no better alternative is available). This initial solution serves as the algorithm's starting point.

At each iteration, TS generates a neighborhood of potential solutions by applying small changes to the current solution. The best candidate, evaluated using a predefined cost function, is then selected as the new current solution. To avoid revisiting solutions that have been explored, the algorithm employs a tabu list—a short-term memory structure that temporarily stores recently visited solutions. These solutions are excluded from selection for a specified number of iterations, helping the algorithm break free from cycles and local optima.

Although TS does not guarantee convergence to a global optimum, it is widely regarded as a powerful and practical technique capable of finding high-quality solutions in many applications. For further details on TS, refer to [40,41,43].

To design a TS algorithm, three problem-specific components must be defined: an initial solution, a cost function to evaluate the solutions produced by the algorithm, and a perturbation procedure to create new solutions from the current one.

3.2.1. Initial Solution

The initial solution is created by randomly shuffling the VMs and assigning them to PMs in a random order. If a VM cannot be placed on a PM due to insufficient capacity, the VM is rejected.

3.2.2. Perturbation Procedure

To generate a new solution from the current one, we define the following two perturbation procedures:

- Swap perturbation: given a VM v_i , where $v_i \in V$, the following steps are conducted:
 - Generate a random number k , uniformly distributed in the interval $[1..N]$.
 - Swap VM v_i with VM v_k , ensuring that v_k is a hosted VM. If v_k is not hosted, repeat the process by generating a new k until a valid swap can be performed.
- Migration perturbation
 - Let P_p , where $P_p \in P$, be the PM currently hosting VM v_i .
 - Generate a random number q , uniformly distributed in the interval $[1..M]$, ensuring that $q \neq p$.
 - Attempt to move VM v_i from P_p to P_q . If the migration is not feasible due to resource constraints, fall back to the swap perturbation.

The overall perturbation procedure is defined as follows:

- Generate a random number i , uniformly distributed in the interval $[1..N]$, to select a VM v_i . The two perturbations are executed, depending on whether VM v_i is hosted or rejected.
 - If v_i is a rejected VM, perform a swap perturbation.
 - If v_i is a hosted VM, execute a migration perturbation.

3.2.3. Cost Function

The cost function is defined based on the objective to be optimized. The TS algorithm is executed three times, each targeting a different objective. Similar to the MOILP approach, the first execution of the TS aims to maximize the number of VMs hosted in the data center. Once this maximum is achieved, the algorithm is run again with the goal of minimizing resource wastage, while ensuring that the previously obtained maximum number of hosted VMs is maintained. Finally, the TS is executed for a third time, focusing on minimizing energy consumption, while keeping both the maximum number of hosted VMs and the reduced resource wastage from earlier iterations.

3.2.4. Diversification Strategy

The diversification strategy is intended to assist the algorithm in exploring new regions of the solution space, particularly when it becomes trapped in a local optimum or shows no improvement after several iterations. A solution with a higher cost than the previously best-known solution is promoted to the current solution. The tabu list is cleared, the stagnation iteration count is reset to zero, and the diversification application counter is incremented. This prevents the algorithm from overusing diversification and helps maintain focus on intensifying around promising solutions.

4. Simulation Setup and Results

In this section, we present the simulation results to assess the performance of the proposed approaches in tackling the multi-objective VMPP. We compare the effectiveness of the MOILP model and the MOTS algorithm with other methods from the literature, described as follows:

- Multi-Objective Ant Colony Optimization (MOACO): MOACO is a multi-objective optimization method aimed at finding a Pareto-optimal solution set while minimizing energy consumption, resource wastage, and communication energy costs between network elements in the DC [35].
- Multi-Objective Genetic Algorithm (MOGA): MOGA is a multi-objective optimization technique based on genetic algorithms and Bernoulli simulation, designed to minimize both the number of PMs used and resource wastage simultaneously [44].
- First Fit (FF) algorithm: The FF algorithm assigns each incoming VM to the first available PM with sufficient resources to accommodate it.
- First Fit Decreasing (FFD) algorithm: FFD is an extension of the FF algorithm that first sorts the VMs in descending order of their resource requirements (e.g., from largest to smallest CPU demands) before placing them on available PMs.
- Worst Fit (WF) algorithm: The WF algorithm aims to distribute VMs across the available PMs by placing each VM on the PM with the most remaining resources.

4.1. Simulation Parameters

For the benchmarks, we conducted the assessment tests using a homogenous DC with 15 PMs ($M = 15$), each equipped with 48 cores, 128 GB of memory, and 1.5 TB of storage. Each PM consumed 1376 watts when idle (no CPU use) ($\mathcal{E}^0 = 1376$) and 1872 watts when fully utilized (100% CPU use) ($\mathcal{E}^1 = 1872$), according to the manufacturer's specifications.

VM requests were generated according to a uniform random distribution, with four types of VMs: Small (S), Medium (M), Large (L), and XLarge (XL). Their respective characteristics are given in Table 4. Problem instances were solved for different values of N , the number of VM requests. For each value of N , 50 test scenarios were randomly generated,

and the performance metrics were computed for each scenario. Each plot represents the average performance metric across these 50 test scenarios. Figure 4 illustrates the distribution of generated VM sizes for various values of N , with the proportion of each VM type (S, M, L, and XL) remaining relatively balanced as N increases.

The MOILP model was implemented in OPL and solved using CPLEX. The MOTS algorithm, along with the comparative algorithms, were coded in C. All experiments were conducted on a system equipped with an Intel Core i7 (2.6 GHz) processor and 16 GB of RAM.

Table 4. VM types and resource requirements.

VM Type	CPU (Cores)	RAM (GB)	Storage (GB)
S	3	4	50
M	4	8	100
L	5	12	150
XL	6	24	250

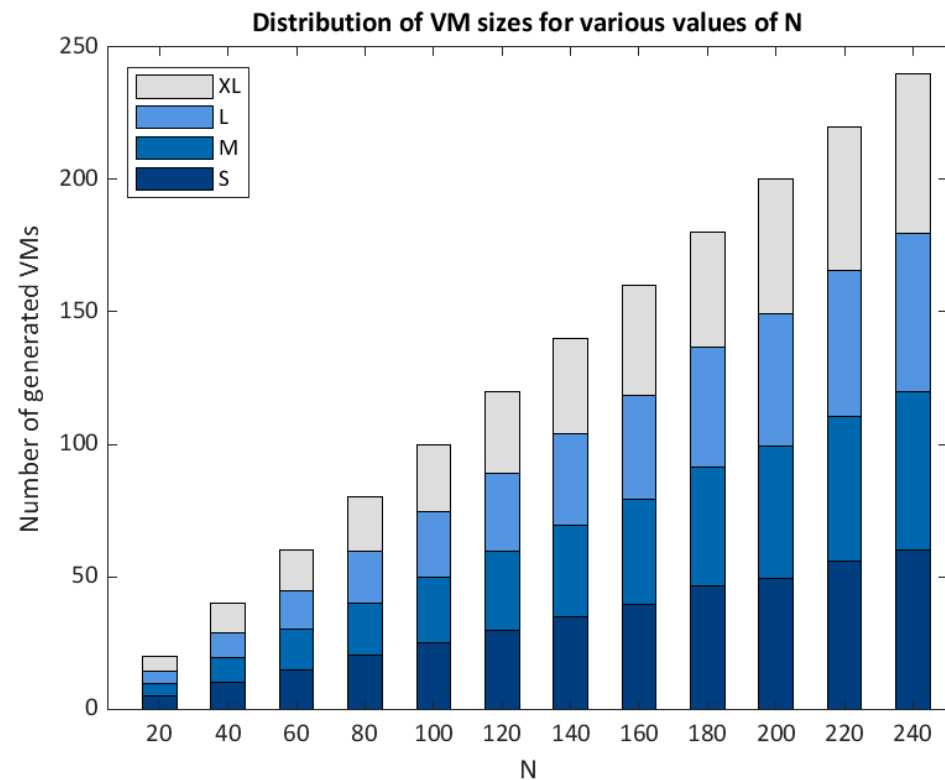


Figure 4. Distribution of VM sizes for various values of N .

4.2. Performance Comparison

Figures 5–7 provide a comparative analysis of the various VM hosting algorithms considered in this study, based on three key performance metrics. The figures respectively illustrate the percentage of VMs hosted, residual resource wastage, and total power consumption, offering insights into the algorithms' efficiency and scalability.

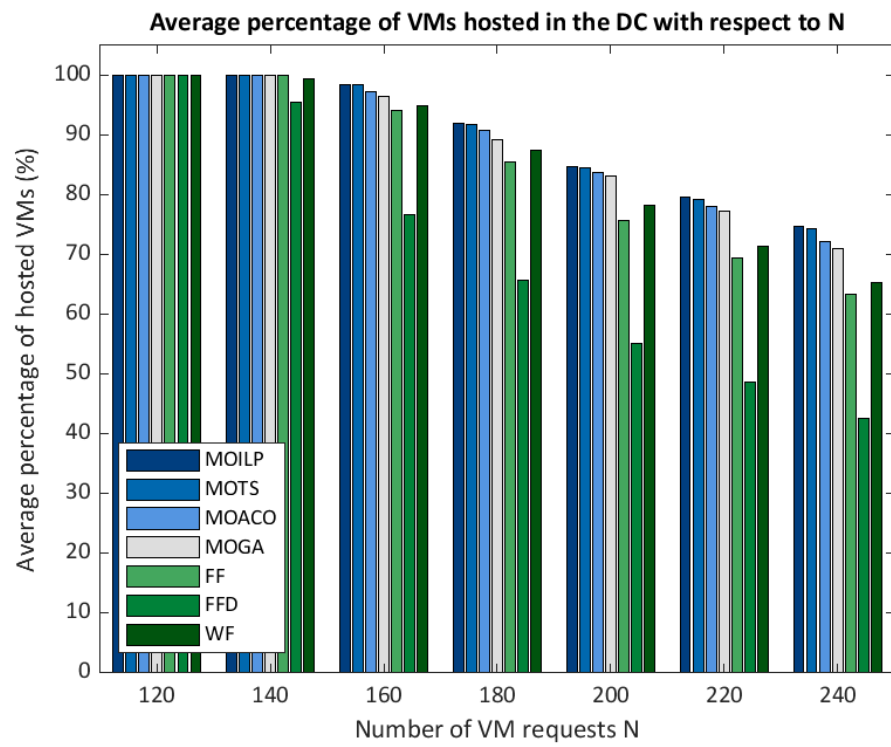


Figure 5. Average percentage of hosted VMs.

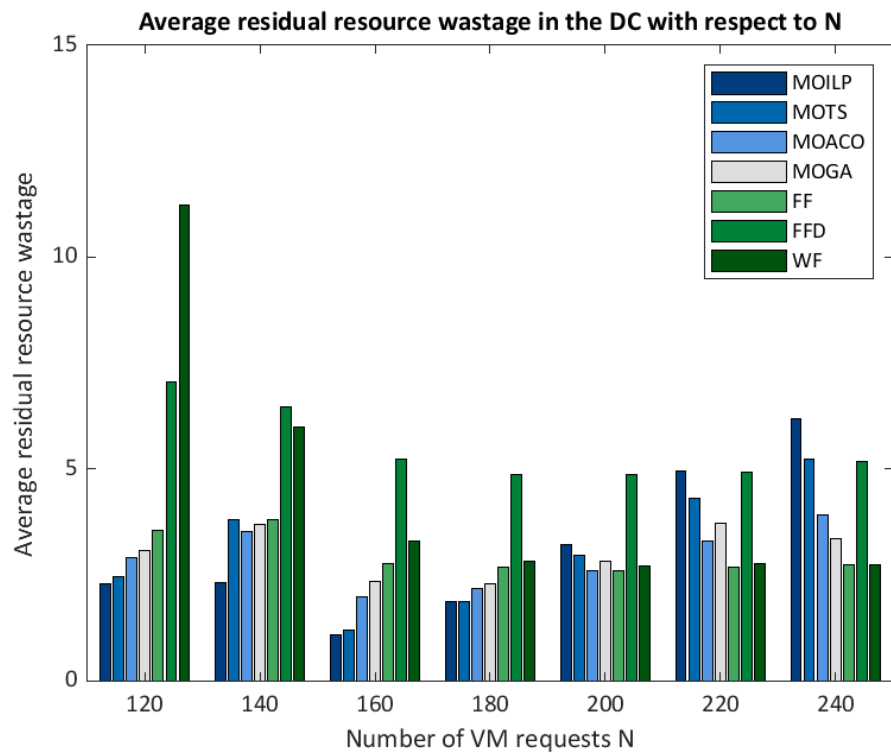


Figure 6. Average residual resource wastage.

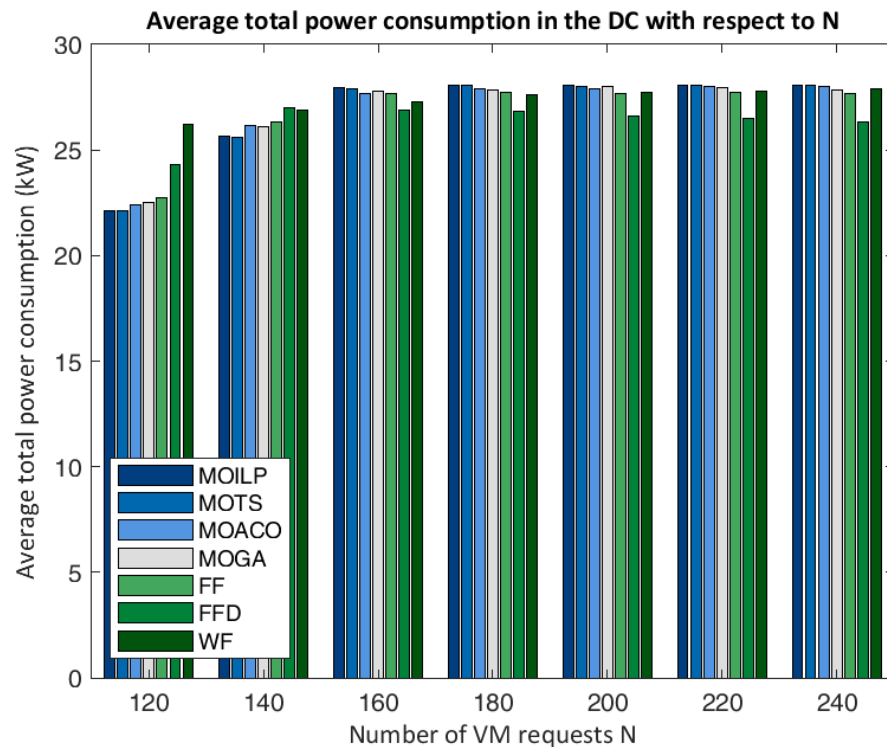


Figure 7. Average total power consumption.

Figure 5 shows the percentage of VMs successfully hosted as VM requests increase. All algorithms host all VMs when N is below 140, except FFD and WF, which start rejecting requests at $N = 140$. As N grows, rejections increase, with MOILP and MOTS consistently outperforming others. Under heavy loads, MOTS nearly matches MOILP, with only a 0.25% difference. Compared to MOACO and MOGA, MOTS achieves average gains of 2% and 3%, respectively, and performs significantly better than FF, FFD, and WF, hosting 11%, 32%, and 9% more VMs. FFD shows the poorest performance, especially at high N .

Figure 6 highlights average residual resource wastage as N increases. MOILP and MOTS demonstrate superior resource efficiency, especially at lower request levels, by hosting all VMs with minimal wastage. While MOACO and MOGA exhibit higher wastage due to less effective allocation, FF, FFD, and WF perform moderately but host fewer VMs. As N grows, MOILP and MOTS experience slightly increased wastage, reflecting their higher hosting rates. Overall, MOILP and MOTS are the most effective in reducing unused resources.

Figure 7 depicts total power consumption across algorithms as N rises. MOILP and MOTS consume slightly more power due to their higher VM acceptance rates and associated CPU usage. Despite this, their energy efficiency stems from better resource utilization. In contrast, FF, FFD, and WF exhibit inefficient resource use, resulting in comparable or higher power consumption relative to MOILP and MOTS. MOTS consumes 1% more energy than MOACO, MOGA, and FF on average, with differences rising to 2% and 6% compared to WF and FFD, respectively, as the latter two accept fewer VMs.

To summarize, MOILP and MOTS excel in resource utilization and energy efficiency due to their optimization techniques. MOILP leverages integer linear programming to achieve precise, globally optimized allocations, particularly effective under high demand. MOTS, with its TS heuristic, delivers near-optimal solutions quickly, making it ideal for large-scale scenarios. Both algorithms minimize resource wastage and prevent over-provisioning, leading to lower energy consumption. In contrast, MOACO and MOGA, while hosting fewer VMs than MOILP and MOTS, are less energy-efficient due to their heuristic nature, which can result in suboptimal allocations under heavy loads.

In Figures 8–11, we present the average percentage of accepted VMs of types S, M, L, and XL, respectively, as a function of N , for the various algorithms evaluated. The MOILP model and the MOTS algorithm achieve the highest acceptance rates for VMs of types S, M, and L, with nearly 100% acceptance for types S and M, even under a heavy load. However, as the number of VM requests increases, the acceptance rates for type L drop to 92% for MOILP and 82.5% for MOTS. Both MOILP and MOTS outperform the other algorithms in hosting VMs of types S, M, and L. In contrast, Figure 11 shows that MOILP and MOTS have the lowest acceptance rates for XL VMs. Specifically, the acceptance rate for XL VMs drops to 7.5% for MOILP and 15.5% for MOTS when N reaches 240. This difference is due to the fact that MOILP and MOTS prioritize maximizing the total number of hosted VMs over the specific VM types, while the other algorithms aim for a more balanced allocation of all VM types.

In Figure 12, the number of accepted VMs of different types across the available PMs ($M = 15$) is presented for a single problem instance with $N = 200$. The results for FF, FFD, and WF are excluded due to their inefficiency. The x-axis corresponds to the individual PMs (1 to 15), and each group of bars represents the performance of one algorithm (e.g., MOILP, MOTS, MOACO, or MOGA), with the bars segmented by VM type (S, M, L, and XL), as shown in the legend. The height of each bar indicates the number of VMs of a specific type hosted on the respective PM.

These results corroborate earlier findings on the resource utilization efficiency of the MOILP model and the MOTS algorithm. Both demonstrate the highest number of accepted VMs, with MOACO and MOGA following closely. The bar segments also reveal how each algorithm distributes the VM types: some favor larger VMs, while others show a more balanced allocation across types.

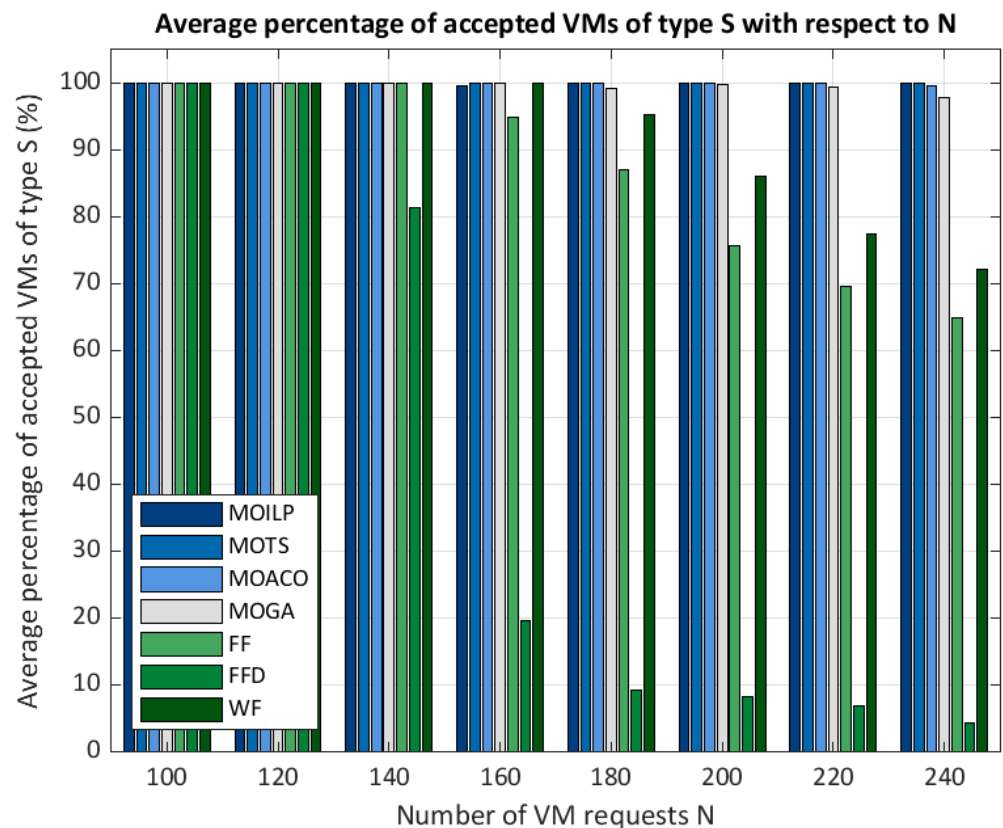


Figure 8. Average percentage of hosted VMs of type S.

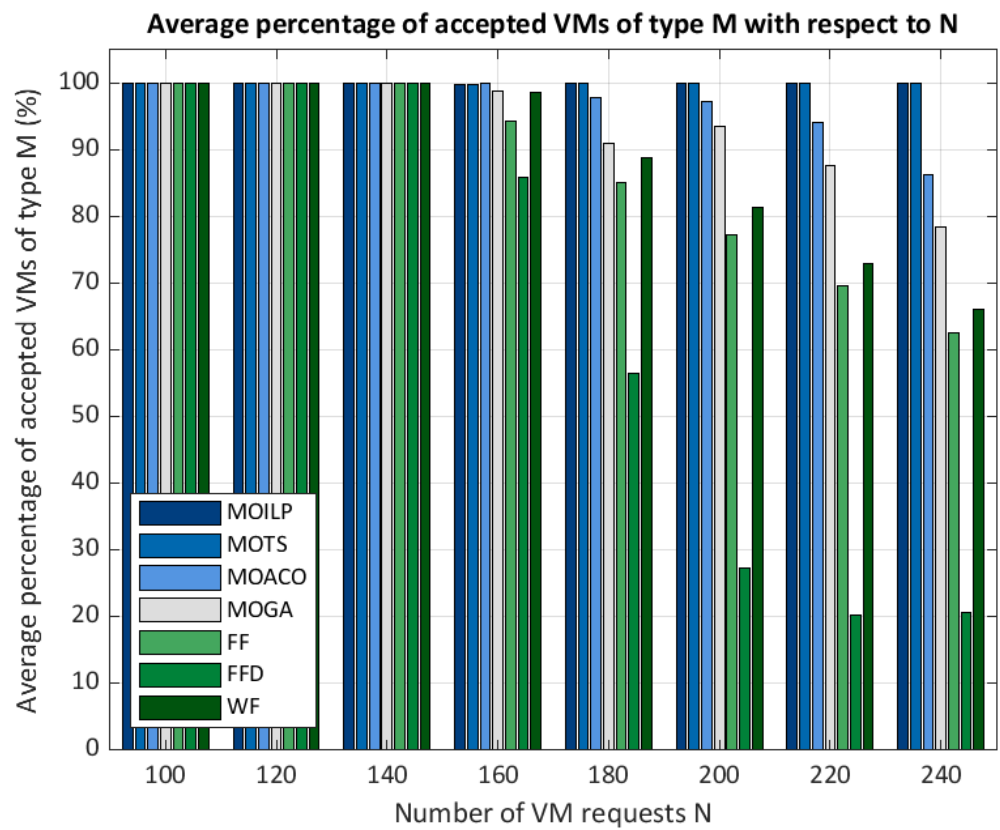


Figure 9. Average percentage of hosted VMs of type M.

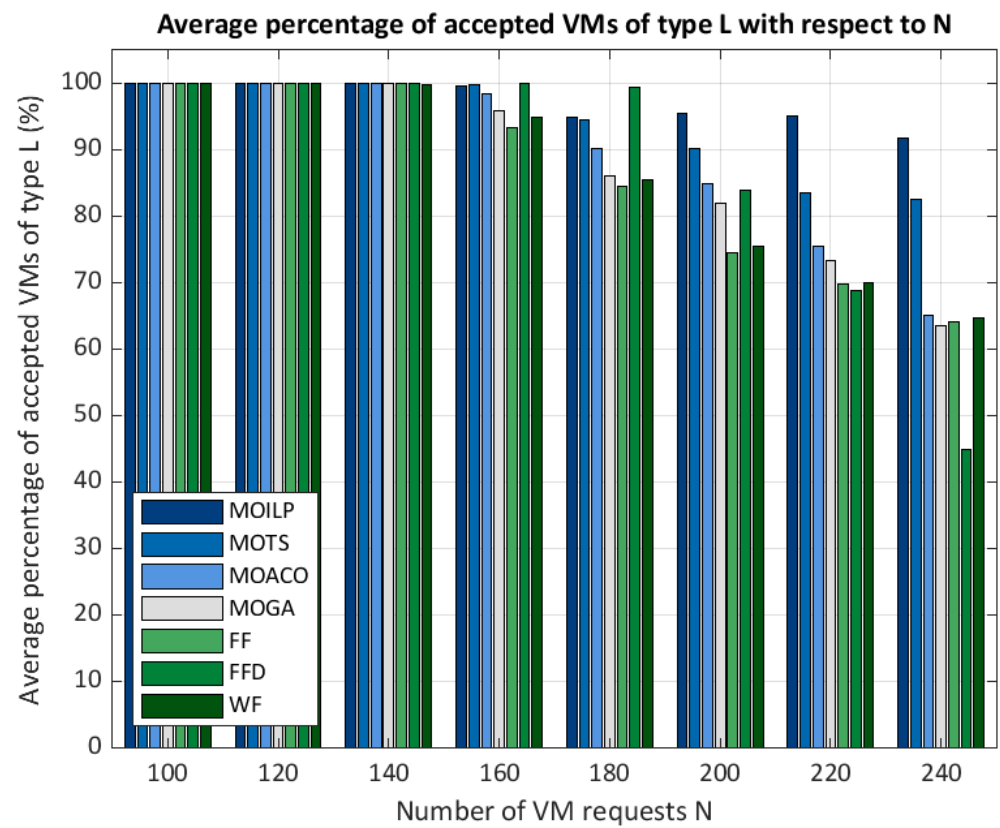


Figure 10. Average percentage of hosted VMs of type L.

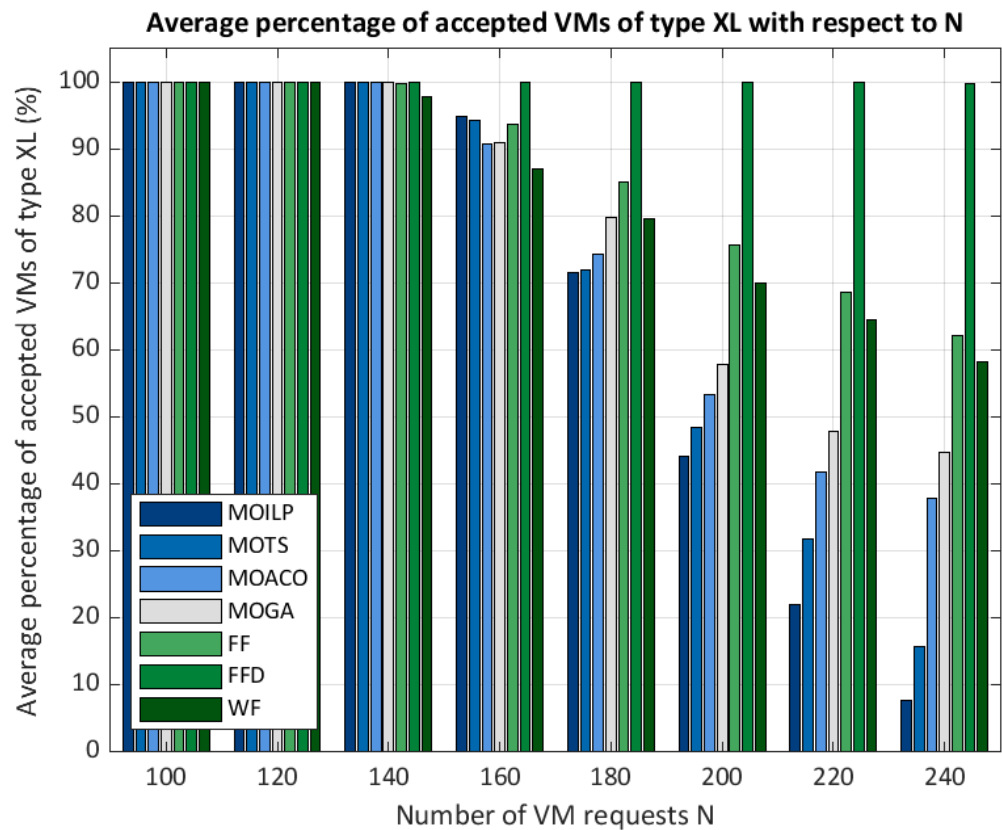


Figure 11. Average percentage of hosted VMs of type XL.

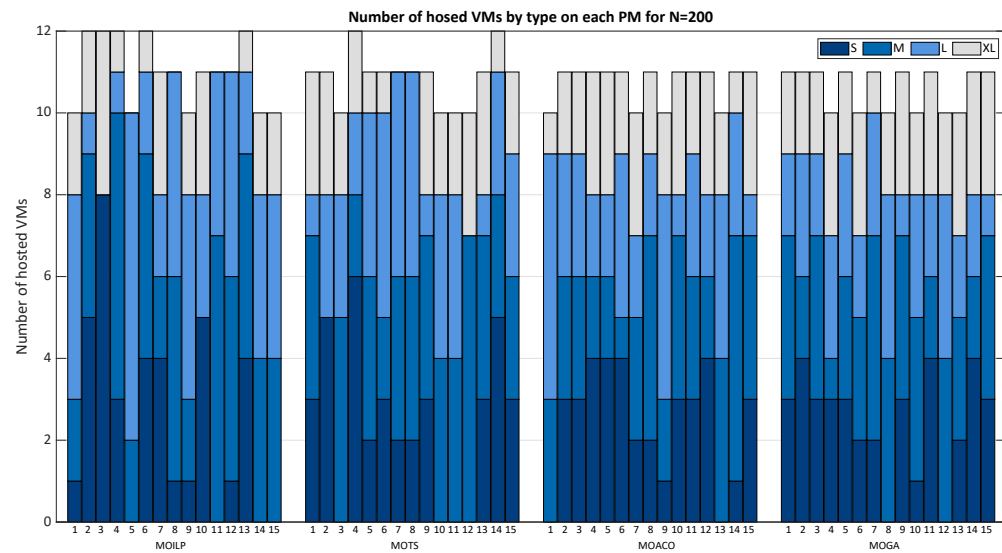


Figure 12. Distribution of hosted VMs by type across PMs for N = 200.

Figures 13–15 compare the average percentage of CPU, RAM, and storage usage across active PMs as N , the number of VM requests, increases. Each plot includes data for MOILP, MOTS, MOACO, and MOGA. The average percentage of resource usage is calculated by dividing the total resources used by the total number of active PMs. MOILP consistently outperforms the other algorithms in terms of maximizing resource utilization (CPU, memory, and storage) across active PMs. MOTS performs well, slightly behind MOILP, while MOACO and MOGA show more variability, especially at higher VM loads, due to a higher number of active PMs.

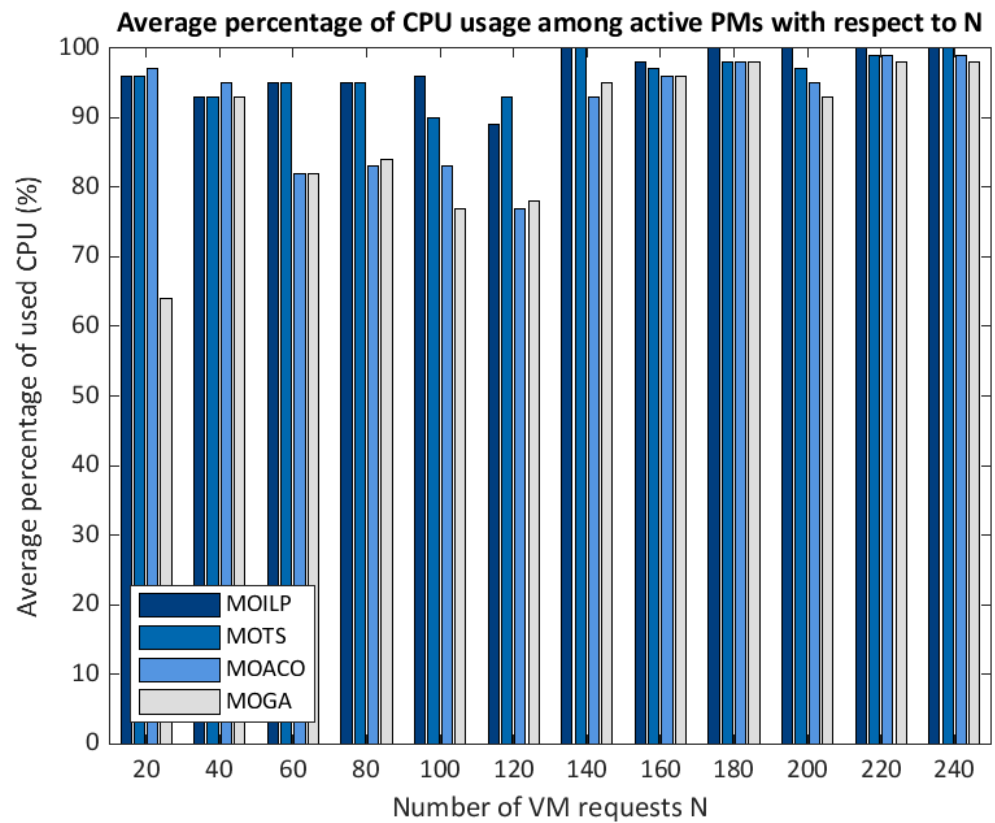


Figure 13. Average percentage of CPU usage among active PMs.

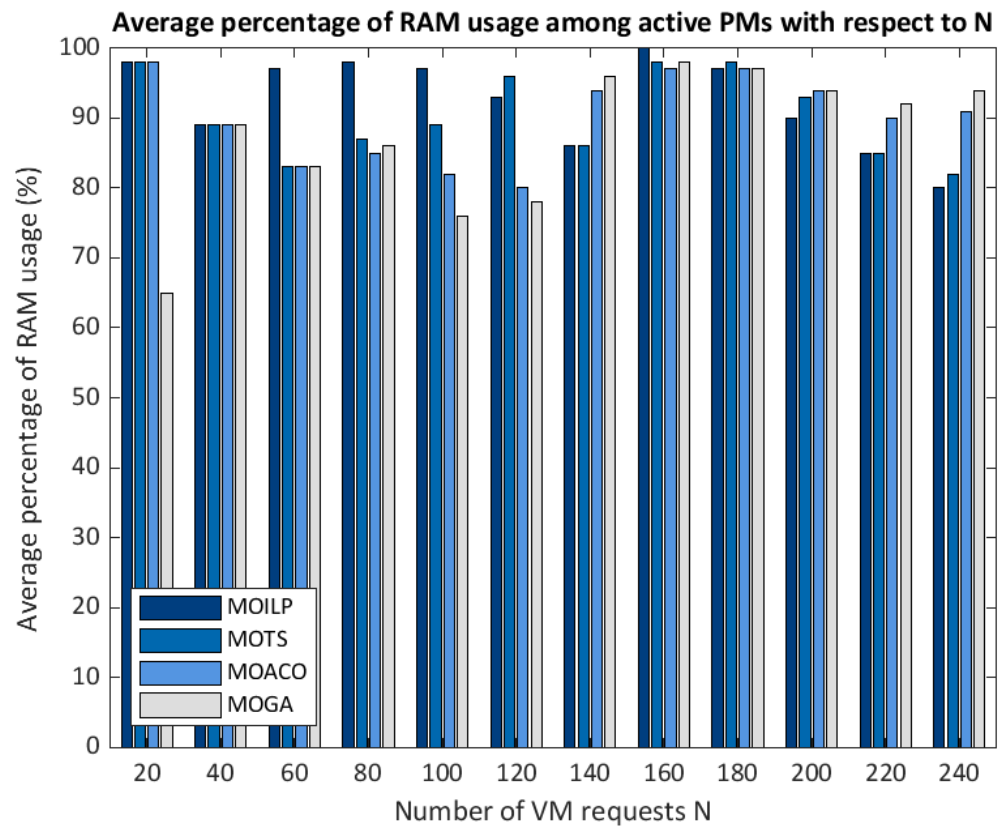


Figure 14. Average percentage of RAM usage among active PMs.

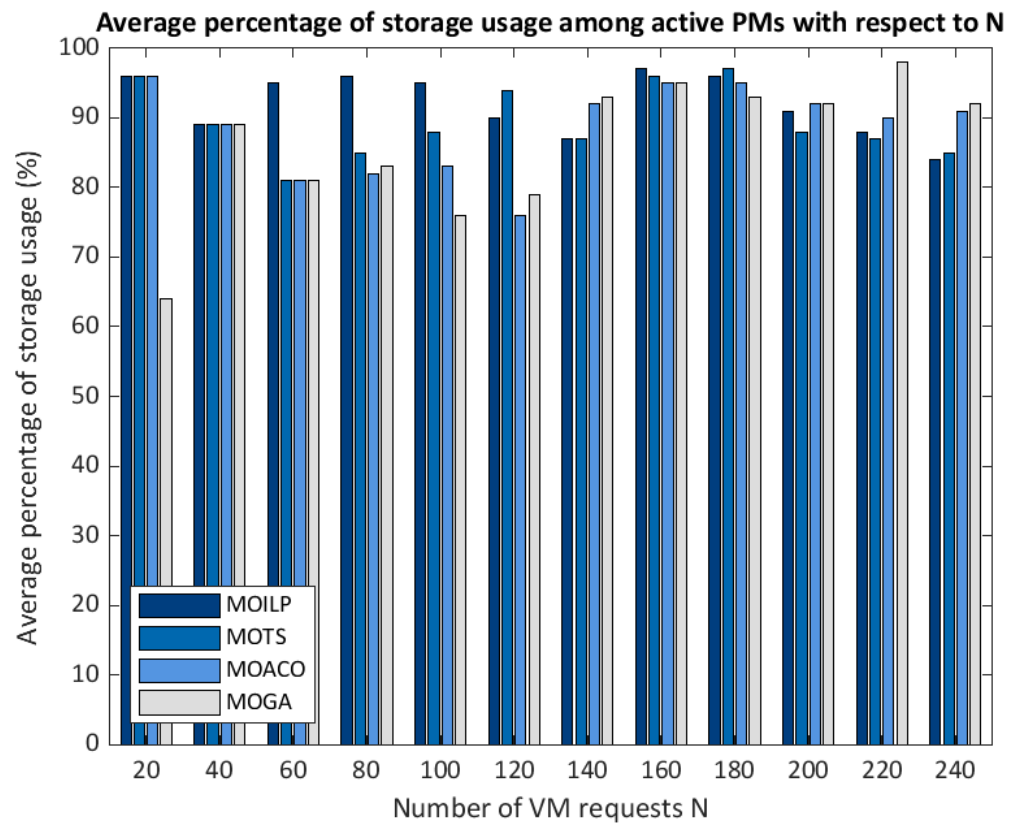


Figure 15. Average percentage of storage usage among active PMs.

In Figure 16, the average execution time (in seconds) for different numbers of VM requests is presented. MOILP exhibits a sharp increase in execution time as N grows, due to its high computational complexity, particularly for larger problem sizes. For $N = 240$, the execution time exceeds 2500 s, making it the slowest algorithm when handling large numbers of VM requests. In contrast, MOTS, MOACO, and MOGA maintain much lower execution times, staying under 500 s, even for $N = 240$ —five times faster than MOILP. MOTS strikes the best balance between efficiency and execution time when compared to MOACO and MOGA.

4.3. Discussion

The experimental results highlight the strengths and weaknesses of the MOILP and MOTS algorithms compared to other VM hosting algorithms. Both MOILP and MOTS consistently outperform others in VM hosting efficiency, resource utilization, and power consumption, with some trade-offs in execution time:

- VM hosting efficiency: MOILP and MOTS host significantly more VMs, with MOTS hosting 32% more VMs than FFD and 11% more than FF at high loads. MOTS is nearly as efficient as MOILP, with only a 0.25% difference in hosted VMs. MOACO and MOGA perform better than other algorithms but still fall short of MOILP and MOTS.
- Residual resource wastage: MOILP and MOTS exhibit the lowest resource wastage, even as VM demand increases, while FF, FFD, and WF show the highest wastage due to their limited hosting capacity.
- Power consumption: MOILP and MOTS consume slightly more power than MOACO and MOGA due to higher VM acceptance rates. For example, MOTS consumes up to 6% more power than FFD but only 1–2% more than MOACO and MOGA, reflecting better resource utilization despite the increased energy use.
- Execution time: MOILP is five times slower than other VMP solutions, while MOTS offers the best balance between efficiency and execution time.

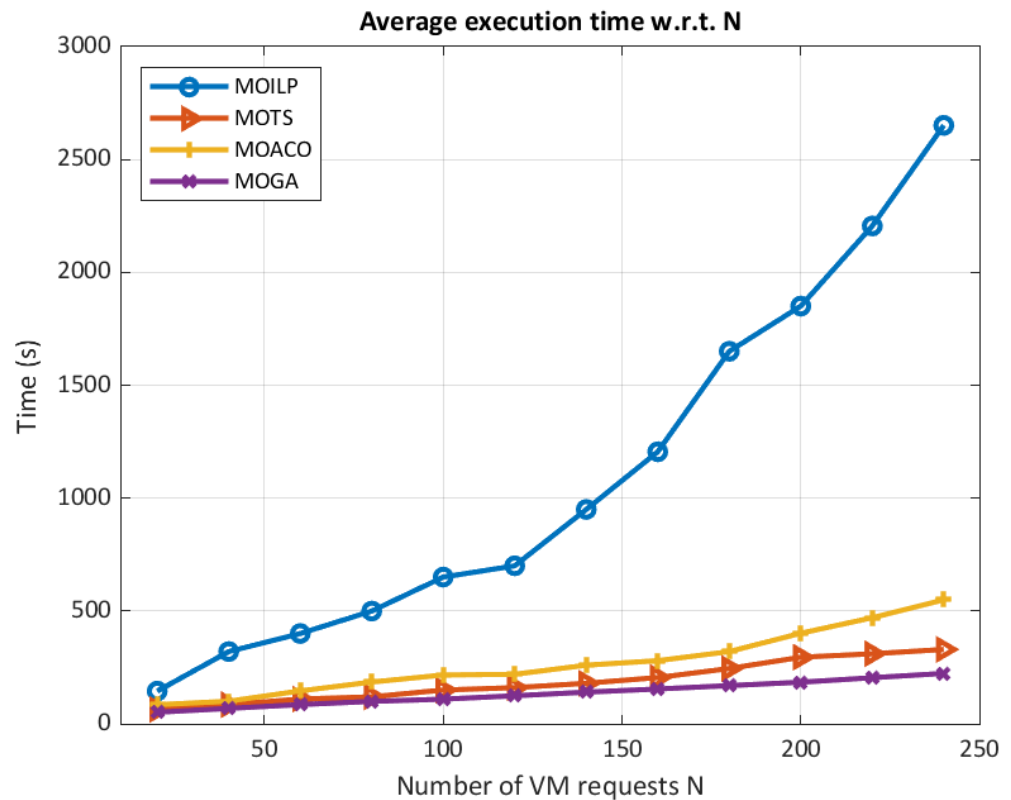


Figure 16. Average CPU execution time for various values of N .

In summary, MOILP and MOTS are the top-performing algorithms, excelling in hosting efficiency, resource utilization, and power consumption. While MOILP is time-consuming, MOTS strikes the best balance between performance and computational cost. Both algorithms, though slightly more power-hungry, offer superior VM hosting capacity and resource optimization, making them ideal for environments where these factors are prioritized.

5. Conclusions and Future Work

In this paper, we proposed exact and approximate methods to address the VM placement problem in cloud data centers, focusing on an MOILP model and a TS algorithm. The model offers optimal solutions, while the TS algorithm provides near-optimal solutions in a more computationally efficient manner. Both methods address three critical objectives: maximizing the number of accepted VMs, minimizing resource wastage, and reducing energy consumption. Simulation results showed that the TS algorithm achieved results close to the optimal ones provided by the MOILP model, offering a practical trade-off between accuracy and computational efficiency. The TS algorithm consistently outperformed traditional methods in the literature, making it a robust approach for real-time VMP in large-scale cloud environments.

Future work will expand this research to the optimization of VM and container placement in cloud-native environments. Containers, unlike VMs, are more lightweight and share the host operating system, which presents unique challenges and opportunities in resource allocation. Key factors such as inter-container communication, resource isolation, and fault tolerance need to be addressed for effective placement strategies in containerized environments. To adapt the TS algorithm for this new context, modifications would involve incorporating container-specific constraints, such as affinity and anti-affinity rules, which define which containers should be placed together or kept apart for optimal performance and fault isolation. Additionally, the model would need to account for dynamic scaling

requirements to handle the elasticity of containerized workloads in real time. Further, load balancing and network management for containerized applications must be integrated into the placement strategy to ensure efficient resource utilization and minimize communication latency between containers. By exploring these challenges and developing hybrid placement strategies, this work aims to enhance resource efficiency, operational agility, and performance in modern, cloud-native infrastructures, thus bridging the gap between traditional VM environments and emerging container-based systems.

Author Contributions: Conceptualization, M.K. and R.R.; methodology, M.K., R.R., A.S.K., M.N. and F.B.; software, M.K. and R.R.; validation, M.K., A.S.K., M.N. and F.B.; formal analysis, M.K., A.S.K., M.N. and F.B.; investigation, M.K. and R.R.; resources, M.K., R.R., A.S.K., M.N. and F.B.; data curation, M.K. and R.R.; writing—original draft preparation, M.K.; writing—review and editing, M.K., R.R., A.S.K., N.M. and F.B.; visualization, M.K.; supervision, M.K., A.S.K., M.N. and F.B.; project administration, M.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data supporting the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

- Gopu, A.; Thirugnanasambandam, K.; R, R.; AlGhamdi, A.S.; Alshamrani, S.S.; Maharajan, K.; Rashid, M. Energy-efficient virtual machine placement in distributed cloud using NSGA-III algorithm. *J. Cloud Comput.* **2023**, *12*, 1–20. [[CrossRef](#)]
- Alourani, A.; Khalid, A.; Tahir, M.; Sardaraz, M. Energy efficient virtual machines placement in cloud datacenters using genetic algorithm and adaptive thresholds. *PLoS ONE* **2024**, *19*, e0296399. [[CrossRef](#)] [[PubMed](#)]
- Dheeraj; Bansal, K. An Analytical Review of VM Allocation and Migration Policies in Cloud Computing. In Proceedings of the 2023 International Conference on Advancement in Computation & Computer Technologies (InCACCT), Gharuan, India, 5–6 May 2023; pp. 704–710. [[CrossRef](#)]
- Saxena, D.; Gupta, I.; Kumar, J.; Singh, A.K.; Wen, X. A Secure and Multiobjective Virtual Machine Placement Framework for Cloud Data Center. *IEEE Syst. J.* **2022**, *16*, 3163–3174. [[CrossRef](#)]
- Chauhan, N.; Kaur, N.; Saini, K.S. Energy Efficient Resource Allocation in Cloud Data Center: A Comparative Analysis. In Proceedings of the 2022 International Conference on Computational Modelling, Simulation and Optimization (ICCMO), Pathum Thani, Thailand, 23–25 December 2022; pp. 201–206. [[CrossRef](#)]
- Natarajan, P.; Panjatharam, V.G.; Rajkumar, T. Comparative Analysis of Techniques for Efficient Resource Utilization in Cloud Environments Through VM Placement Optimization. In Proceedings of the 2023 International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS), Erode, India, 18–20 October 2023; pp. 1015–1018. [[CrossRef](#)]
- Nagadevi, S.; Vidhya; Jeya. Two Dimensional Balanced Resource Utilization Using Vector Heuristics for Multi-Core Aware Virtual Machine Placement Algorithms in a Cloud Environment. In Proceedings of the 2023 2nd International Conference on Edge Computing and Applications (ICECAA), Namakkal, India, 19–21 July 2023; pp. 112–116. [[CrossRef](#)]
- Choudhury, A.; Nath, K.K.; Ghose, M.; Thakran, Y. Memory and CPU utilization-aware Energy-Efficient VM Placement and Consolidation in Cloud Data Centers. In Proceedings of the 2023 IEEE Guwahati Subsection Conference (GCON), Guwahati, India, 23–25 June 2023; pp. 1–6. [[CrossRef](#)]
- Gupta, M.K.; Amgoth, T. Resource-aware algorithm for virtual machine placement in cloud environment. In Proceedings of the 2016 Ninth International Conference on Contemporary Computing (IC3), Noida, India, 11–13 August 2016; pp. 1–6. [[CrossRef](#)]
- Mosa, A.; Sakellariou, R. Dynamic Virtual Machine Placement Considering CPU and Memory Resource Requirements. In Proceedings of the 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), Milan, Italy, 8–13 July 2019; pp. 196–198. [[CrossRef](#)]
- De, U.C.; Satapathy, R.; Patra, S.S. Optimizing Resource Allocation using Proactive Predictive Analytics and ML-Driven Dynamic VM Placement. In Proceedings of the 2023 4th IEEE Global Conference for Advancement in Technology (GCAT), Bangalore, India, 6–8 October 2023; pp. 1–5. [[CrossRef](#)]
- Khoshkholghi, M.A.; Derahman, M.N.; Abdullah, A.; Subramaniam, S.; Othman, M. Energy-Efficient Algorithms for Dynamic Virtual Machine Consolidation in Cloud Data Centers. *IEEE Access* **2017**, *5*, 10709–10722. [[CrossRef](#)]
- Li, L.; Dong, J.; Zuo, D.; Wu, J. SLA-Aware and Energy-Efficient VM Consolidation in Cloud Data Centers Using Robust Linear Regression Prediction Model. *IEEE Access* **2019**, *7*, 9490–9500. [[CrossRef](#)]
- Liu, X.F.; Zhan, Z.H.; Deng, J.D.; Li, Y.; Gu, T.; Zhang, J. An Energy Efficient Ant Colony System for Virtual Machine Placement in Cloud Computing. *IEEE Trans. Evol. Comput.* **2018**, *22*, 113–128. [[CrossRef](#)]
- Vakilinia, S.; Heidarpour, B.; Cheriet, M. Energy Efficient Resource Allocation in Cloud Computing Environments. *IEEE Access* **2016**, *4*, 8544–8557. [[CrossRef](#)]

16. Patel, K.K.; Desai, M.R.; Soni, D.R. Dynamic priority based load balancing technique for VM placement in cloud computing. In Proceedings of the 2017 International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 18–19 July 2017; pp. 78–83. [\[CrossRef\]](#)
17. Chhabra, S.; Singh, A.K. Optimal VM Placement Model for Load Balancing in Cloud Data Centers. In Proceedings of the 2019 7th International Conference on Smart Computing & Communications (ICSCC), Miri, Sarawak, 28–30 June 2019; pp. 1–5. [\[CrossRef\]](#)
18. Zhao, H.; Wang, Q.; Wang, J.; Wan, B.; Li, S. VM Performance Maximization and PM Load Balancing Virtual Machine Placement in Cloud. In Proceedings of the 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), Melbourne, Australia, 11–14 May 2020; pp. 857–864. [\[CrossRef\]](#)
19. Liu, X.; Mashayekhy, L. Joint Load-Balancing and Energy-Aware Virtual Machine Placement for Network-on-Chip Systems. In Proceedings of the 2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC), Zurich, Switzerland, 17–20 December 2018; pp. 124–132. [\[CrossRef\]](#)
20. Wang, S.; Zhou, A.; Hsu, C.H.; Xiao, X.; Yang, F. Provision of Data-Intensive Services Through Energy- and QoS-Aware Virtual Machine Placement in National Cloud Data Centers. *IEEE Trans. Emerg. Top. Comput.* **2016**, *4*, 290–300. [\[CrossRef\]](#)
21. Li, Z.; Yu, X.; Zhao, L. A Strategy Game System for QoS-Efficient Dynamic Virtual Machine Consolidation in Data Centers. *IEEE Access* **2019**, *7*, 104315–104329. [\[CrossRef\]](#)
22. Hadadi, A.; Shameli-Sendi, A. A Quality of Service Aware VM Placement for User Applications in Cloud Data Center. In Proceedings of the 2022 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI), Virtual, 17–19 October 2022; pp. 1–6. [\[CrossRef\]](#)
23. Chavan, V.; Kaveri, P.R. Clustered virtual machines for higher availability of resources with improved scalability in cloud computing. In Proceedings of the 2014 First International Conference on Networks & Soft Computing (ICNSC2014), Guntur, India, 19–20 August 2014; pp. 221–225. [\[CrossRef\]](#)
24. Talbi, J.; Haqiq, A. Adopting a clustering approach toward a scalable IaaS cloud datacenters. In Proceedings of the 2015 International Conference on Cloud Technologies and Applications (CloudTech), Marrakesh, Morocco, 2–4 June 2015; pp. 1–5. [\[CrossRef\]](#)
25. Duong-Ba, T.; Tran, T.; Nguyen, T.; Bose, B. A Dynamic Virtual Machine Placement and Migration Scheme for Data Centers. *IEEE Trans. Serv. Comput.* **2021**, *14*, 329–341. [\[CrossRef\]](#)
26. Li, K.W.; Huang, P.H.; Wen, C.H.P. Reducing network cost of minimal-migration based VM management in cloud datacenters. In Proceedings of the 2016 7th International Conference on the Network of the Future (NOF), Rio de Janeiro, Brazil, 16–18 November 2016; pp. 1–3. [\[CrossRef\]](#)
27. Duong-Ba, T.; Nguyen, T.; Bose, B.; Tran, T. Joint virtual machine placement and migration scheme for datacenters. In Proceedings of the 2014 IEEE Global Communications Conference, Austin, TX, USA, 8–12 December 2014; pp. 2320–2325. [\[CrossRef\]](#)
28. Wang, S.; Gu, H.; Wu, G. A New Approach to Multi-objective Virtual Machine Placement in Virtualized Data Center. In Proceedings of the 2013 IEEE Eighth International Conference on Networking, Architecture and Storage, Xi’an, China, 17–19 July 2013; pp. 331–335. [\[CrossRef\]](#)
29. Jamali, S.; Malektaji, S. Improving grouping genetic algorithm for virtual machine placement in cloud data centers. In Proceedings of the 2014 4th International Conference on Computer and Knowledge Engineering (ICCKE), Mashhad, Iran, 29–30 October 2014; pp. 328–333. [\[CrossRef\]](#)
30. Wei, W.; Wang, K.; Wang, K.; Guo, S.; Gu, H. A Virtual Machine Placement Algorithm Combining NSGA-II and Bin-Packing Heuristic. In Proceedings of the 2019 20th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), Gold Coast, Australia, 5–7 December 2019; pp. 190–195. [\[CrossRef\]](#)
31. Alam, A.B.M.B.; Halabi, T.; Haque, A.; Zulkernine, M. Multi-Objective Interdependent VM Placement Model based on Cloud Reliability Evaluation. In Proceedings of the ICC 2020—2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–7. [\[CrossRef\]](#)
32. Sajadinia, A.; Yari, A. Virtual Machine Placement Strategy Using Clustering and Genetic Algorithm for increasing cloud performance and power saving. In Proceedings of the 2023 28th International Computer Conference, Computer Society of Iran (CSICC), Tehran, Iran, 25–26 January 2023; pp. 1–5. [\[CrossRef\]](#)
33. Braiki, K.; Youssef, H. Multi-Objective Virtual Machine Placement Algorithm Based on Particle Swarm Optimization. In Proceedings of the 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC), Limassol, Cyprus, 25–29 June 2018; pp. 279–284. [\[CrossRef\]](#)
34. De, U.C.; Satpathy, R.; Patra, S.S. Multi-Objective Optimization for Optimal VM Allocation using Tuna Swarm Optimization. In Proceedings of the 2024 3rd International Conference for Innovation in Technology (INOCON), Bangalore, India, 1–3 March 2024; pp. 1–5. [\[CrossRef\]](#)
35. Malekloo, M.; Kara, N. Multi-objective ACO virtual machine placement in cloud computing environments. In Proceedings of the 2014 IEEE Globecom Workshops (GC Wkshps), Austin, TX, USA, 8–12 December 2014; pp. 112–116. [\[CrossRef\]](#)
36. Qin, Y.; Wang, H.; Zhu, F.; Zhai, L. A Multi-Objective Ant Colony System Algorithm for Virtual Machine Placement in Traffic Intense Data Centers. *IEEE Access* **2018**, *6*, 58912–58923. [\[CrossRef\]](#)
37. Wei, W.; Gu, H.; Lu, W.; Zhou, T.; Liu, X. Energy Efficient Virtual Machine Placement with an Improved Ant Colony Optimization Over Data Center Networks. *IEEE Access* **2019**, *7*, 60617–60625. [\[CrossRef\]](#)

38. Suseela, J.; Venugopal, J. A Multi-Objective Hybrid ACO-PSO Optimization Algorithm for Virtual Machine Placement in Cloud Computing. *Int. J. Res. Eng. Technol.* **2014**, *3*, 474–476.
39. Bharathi, P.D.; Prakash, P.; Kiran, M.V.K. Energy efficient strategy for task allocation and VM placement in cloud environment. In Proceedings of the 2017 Innovations in Power and Advanced Computing Technologies (i-PACT), Vellore, India, 21–22 April 2017; pp. 1–6. [[CrossRef](#)]
40. Glover, F.W. Tabu Search—Part I. *ORSA J. Comput.* **1989**, *1*, 190–206. [[CrossRef](#)]
41. Glover, F.W. Tabu Search—Part II. *ORSA J. Comput.* **1990**, *2*, 4–32. [[CrossRef](#)]
42. Xu, J.; Fortes, J.A.B. Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments. In Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing, Hangzhou, China, 18–20 December 2010; pp. 179–188. [[CrossRef](#)]
43. Glover, F.; Laguna, M. *Tabu Search*; Kluwer Academic Publishers: Boston, MA, USA, 1997.
44. Riahi, M.; Krichen, S. A multi-objective decision support framework for virtual machine placement in cloud data centers: A real case study. *J. Supercomput.* **2018**, *74*, 2984–3015. [[CrossRef](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.