Journal of Industrial and Management Optimization

https://www.aimspress.com/journal/jimo

JIMO, 22(8): 3741–3760.
DOI:10.3934/jimo.2026134
Received: 13 July 2024
Revised: 12 March 2025
Accepted: 05 June 2026
Published: 16 July 2026

Research article

Multi-objective adaptive large neighborhood search for virtual machine placement problem

Dindar ÖZ [1] **and Tolga Bugra ALTUNTAS**[1,2,*]

[1] Department of Software Engineering, Yaşar University, Turkey

[2] Department of Computer Science, OsloMet - Oslo Metropolitan University, Norway

* **Correspondence:** Email: tolga.b.altuntas@oslomet.no.

Abstract: Cloud computing enables the delivery of various services over the Internet. Cloud Service Providers manage these services using Virtual Machines, which emulate physical machines to provide the necessary computing resources. Efficiently managing and allocating these resources is essential for achieving optimal performance and cost-effectiveness. Yet, the increasing complexity and size of cloud environments pose issues for Virtual Machine Placement (VMP), a problem acknowledged as NP-hard because of its complicated and exponentially growing solution space. This paper addresses the multi-objective VMP problem, focusing on energy management and resource utilization. We adapted the Adaptive Large Neighborhood Search (ALSN) algorithm for VMP by using five problem-specific destroy and repair operators to generate new solutions. These operators are selected adaptively by changing their probabilities according to their performance on runtime. Also, a weight value is utilized to improve the spread of solutions on the Pareto set generated by ALNS. To the best of our knowledge, ALNS has never been used in multi-objective VMP. Extensive experiments are conducted on problem instances using both legacy and modern physical machine configurations. The results show that the ALNS algorithm outperforms state-of-the-art algorithms such as the strength Pareto evolutionary algorithm II (SPEA-II), the non-dominated sorting genetic algorithm II (NSGA-II), and the epsilon dominance NSGA-II (ϵ-NSGAII), achieving up to 4.2 higher hypervolume in large-scale scenarios and maintaining solution diversity where others fail. The experimental results demonstrate the robustness and adaptability of the ALNS algorithm in varying hardware configurations and large-scale instances.

Keywords: virtual machine placement problem; multiobjective optimization problem; adaptive large neighborhood algorithm; cloud computing; multiobjective algorithms
Mathematics Subject Classification: Primary: 90C59, 90C27; Secondary: 90C29

1. Introduction

Cloud computing has established itself as the primary method of delivering services through the provision of virtualized resources on demand. Cloud Service Providers (CSPs) oversee the provision of computing resources to meet the Service-Level Agreement (SLA), which entails metrics such as security and quality that CSPs must achieve. Users access infrastructure, platforms, and software applications as a service, paying only for what they use from any location and at any time. At the center of these services are virtual machines (VMs), which simulate physical machines by enabling the necessary computing resources in a virtual environment. VMs provide CSPs with a powerful mechanism to efficiently manage and allocate resources dynamically, ensuring optimal performance and cost-efficiency. However, the increasing complexity and scale of cloud environments require sophisticated strategies for the placement of virtual machines (VMP). Figure 1 shows a generic representation of VMP.

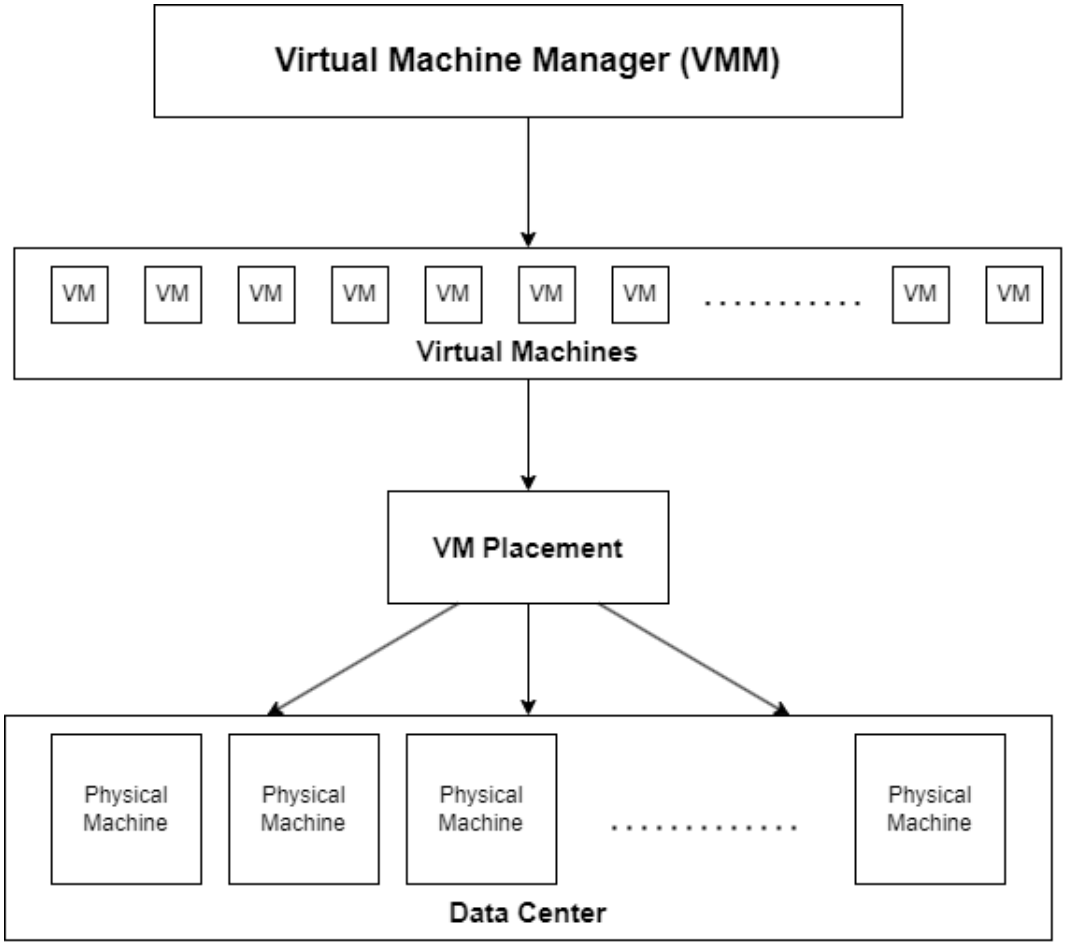


Figure 1. VM Placement in Cloud Computing.

The rapid expansion of cloud programming highlighted the need for efficient solutions for VMP, which is vital for CSPs due to its direct impact on the performance and energy consumption of data centers, ultimately affecting both operational costs and customer satisfaction.

VMP involves mapping a given set of virtual machines (VMs) into a limited number of physical machines (PMs) while maintaining several conflicting objectives, such as optimizing resource utilization, energy management, minimizing communication costs, and delays. Due to its complex nature and exponential growth of solution space, VMP has been acknowledged as an NP-hard problem [1, 2]. To address these challenges, heuristic algorithms like First Fit Decreasing [3], Best Fit [4], or meta-heuristic algorithms such as Ant Colony Optimization [5], Particle Swarm Optimization [6], and Genetic Algorithm [7] are often used.

The VMP problem can be divided into static and dynamic settings. In static VMP, the workload of all VMs is known in advance and does not change. In real systems, it is often utilized in the initial provisioning phase, when VMs are first deployed. On the contrary, dynamic VMP includes reallocation or migration of VMs due to changing requirements of VMs, failures, and load imbalances in PMs. This paper focuses on the static VMP setting, aiming to determine an initial placement of VMs onto PMs while optimizing for energy efficiency and resource utilization.

In order to stay competitive and minimize the cost, CSPs combine two or more objectives. Trying to tackle these objectives alone might result in a suboptimal solution that reduces overall system performance. For example, aggressively trying to minimize energy consumption can lead to SLA violations due to insufficient resource allocation to the users or loss in performance. Contrariwise, maximizing the performance could cause unnecessary energy consumption and increased costs. Therefore, much of the research focuses on solving Multi-Objective VMP.

Despite considerable advancements in heuristic and metaheuristic strategies for Virtual Machine Placement (VMP), present studies largely depend on population-based methods, such as Genetic Algorithms [7], Ant Colony Optimization [8], and Particle Swarm Optimization [6]. These strategies, while effective, frequently face challenges related to scalability, convergence, or the preservation of solution variety as the problem size expands. Additionally, adaptive single-solution based search strategies, like the Adaptive Large Neighborhood Search (ALNS) framework, have not been extensively investigated for VMP problems, especially in a multi-objective environment.

This paper tackles energy management and resource utilization objectives to solve the multi-objective VMP problem. The proposed solution utilizes an extended Adaptive Large Neighborhood Search (ALNS) used by Erdogdu et al. [9], and the algorithm has been successfully applied to other single and multi-objective problems [10–12]. ALNS focuses on a strategy that destroys and repairs parts of a single solution. This strategy is particularly effective at escaping local optima and balancing exploration and exploitation. The adaptive selection of operators based on their real-time performance further enhances it as a strong candidate for handling the large, NP-hard solution space of VMP problems. Five destroy and repair operators are integrated to use ALNS for the VMP problem. To the best of our knowledge, ALNS has never been applied in VMP to create a Pareto set of solutions. The proposed solution is compared with non-dominated sorting genetic algorithm II (NSGA-II), strength Pareto evolutionary algorithm II (SPEA-II), and epsilon dominance NSGA-II (ϵ-NSGAII).

The remaining sections of the article are organized as follows: Section 2 presents the current state of the research on this topic. Then, in Section 3, the details of the proposed solution are listed. Following that, Section 4 demonstrates the experimental work. Finally, this study's summary and future work constitute Section 5.

2. Related Work

The VMP problem has been an active research topic [13] with varying objectives ranging from energy management, resource utilization, network traffic, and multi-objective. Azizi et al. [14] proposed a Greedy Randomized Algorithm for large-scale data centres with multi-dimensional resources. They apply the greedy solution to reduce the number of required PMs and energy consumption. Wang et al. [15] also designed a two-phase greedy algorithm that focuses on the power efficiency of PMs to decrease active PM count and energy consumption. Lu et al. [7] introduced an improved genetic algorithm (GA), which innovates the initial solution generation step of GA. Their algorithm displayed significant improvements in energy efficiency. Moorthy et al. [5] use ant colony optimization to dynamically allocate resources to meet VM demands. They proposed a system to prioritize resource utilization and minimize migrations of VMs.

Real-life scenarios of VMP require optimizing multiple conflicting objectives. Gao et al. [8] proposed a multi-objective ant colony algorithm to tackle resource wastage and energy consumption objectives simultaneously. Their experiments show the algorithm can compete with existing multi-objective algorithms and demonstrate scalability for real-life cases. Another VMP algorithm designed by Braiki et al. [6] is based on particle swarm optimization that aims to minimize energy consumption and resource wastage. Their fitness calculation gave equal weight to objective fitness functions and reduced it to a single objective. Another work by Braiki et al. [16] introduced a fuzzy algebra that combines energy consumption and resource utilization. They use a fuzzy-based multi-objective implementation of best fit-decreasing. Their method shows improvements in both objectives. Recent research has increasingly explored learning-based techniques for solving the VMP problem. Qin et al. [1] proposed a multi-objective reinforcement learning approach that finds a Pareto set by minimizing both resource wastage and energy consumption. They employed a Chebyshev scalarization function to simplify weight selection across multiple objectives. Another recent research that focuses on reinforcement learning is done by Bhatt et al. [17] aims to optimize resource and energy management. They applied a multi-objective reinforcement learning strategy that uses an enhanced VM selection strategy along with two resource management heuristics to order VMs.

ALNS framework was first introduced by Ropke et al. [12, 18] to solve the vehicle routing problem. Instead of using one large neighborhood, they applied various destroy and repair operators to a given solution. These operators are selected dynamically based on their past performance. Recent works of ALNS also focus on solving multi-objective problems. Rifai et al. [19] employed ALNS to solve distributed reentrant permutation flow shop scheduling. They simultaneously satisfied three objectives: minimizing makespan, total cost, and average tardiness. Liao et al. [20] developed ALNS in order to tackle the Green Meal Delivery Routing Problem. They proposed a multi-objective scheduling model to maximize customer satisfaction and rider balance utilization and minimize carbon footprint. Erdogdu et al. [9] proposed a hybrid-ALNS algorithm that utilizes two new local search algorithms. They studied a multi-objective green vehicle routing problem with two objectives: minimizing the total distance and fuel consumption of all vehicle routes.

In this work, we propose a solution based on the ALNS algorithm. Demir et al. [10] put forward an extended ALNS algorithm to solve the pollution routing problem. The algorithm includes a simulated annealing (SA) mechanism to accept worse solutions with a probability. This mechanism allows us to

conduct extensive exploration of search space. Five destroy operators are utilized to access a diversity of solutions. After that, one of the five repair operators is selected to generate a new solution. Repair algorithms include greedy methods such as Best Fit [21], First Fit Decreasing [22], Greedy Randomized Algorithm [14], and random repair. These operators are explained in Section 3.2.1 and 3.2.2

The following section will explain the formal definition of the VMP problems that we focus on in this study and provide detailed information about the expanded ALNS algorithm.

3. Methodology

3.1. Problem Formulation

As mentioned in the introduction, CSPs must stay competitive by solving VMP problems that combine two or more objectives. Therefore, this paper tackles the VMP problem with two main objectives: energy management and resource utilization with the SLA constraints.

Let $V = \{v_1, v_2, \ldots, v_n\}$ be the set consisting of n virtual machines (VMs).

A VM comprises two types of resources that CSP must fulfill: CPU and Memory. These requirements are represented as a vector of tuples for each v_i

$$v_i = (v_i^{cpu}, v_i^{mem}), \quad i \in \{1, 2, \ldots, n\}. \tag{3.1}$$

The vector of P represents the m physical machines (PMs) as follows:

$$P = \{p_1, p_2, \ldots, p_m\}. \tag{3.2}$$

A PM consists of two different resource capacities: CPU and Memory. A vector of tuple p_i denotes these capacities as follows:

$$p_i = (p_i^{cpu}, p_i^{mem}), \quad i \in \{1, 2, \ldots, m\}. \tag{3.3}$$

A solution for VMP can be represented as an $n \times m$ matrix μ,

$$\mu = \begin{bmatrix} \mu_{11} & \mu_{12} & \cdots & \mu_{1m} \\ \mu_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \mu_{n1} & \cdots & \cdots & \mu_{nm} \end{bmatrix}, \tag{3.4}$$

where $\mu_{ij} \in \{0, 1\}$ and $\mu_{ij} = 1$ denotes the assignment of i^{th} VM on the j^{th} PM. Note that a VM can only be placed on a single PM, that is,

$$\sum_{j=1}^{m} \mu_{ij} = 1, \quad \forall\, i \in \{1, 2, \ldots, n\}. \tag{3.5}$$

3.1.1. Energy Management

Cloud computing demands a substantial amount of electrical energy to function. Therefore, it's vital to implement Green Cloud Computing [23] solutions in order to reduce operational costs and minimize the environmental impact. Beloglazov et al. [24] introduced a power consumption function that can represent the energy consumption of j^{th} PM as follows:

$$P(U_j^{cpu}) = \begin{cases} k, & P_{\max} + (1-k)\, P_{\max}\, U_j^{cpu}, \quad \text{if } \sum_{i=1}^{n} \mu_{ij} \geq 1, \\ 0, & \text{otherwise,} \end{cases} \tag{3.6}$$

$$U_j^{cpu} = \sum_{i=1}^{n} \frac{v_i^{\text{cpu}} \times \mu_{ij}}{p_j^{\text{cpu}}}, \tag{3.7}$$

where $P_{\max}$ represents the maximum power consumed by a fully utilized PM, k is the fraction of the power consumed by the idle PM, and U_j^{cpu} is the CPU utilization of the jth PM which is calculated as in Equation 3.7. Equation 3.6 ensures that if a PM has no VMs allocated to it, it is considered powered off and does not contribute to the calculation of total power consumption.

3.1.2. Resource Utilization

Effective resource utilization is pivotal for data centers to reduce operational costs and increase overall efficiency. The aim is to effectively allocate all demands to maximize the resource utilization of PMs. This paper focuses on a static VMP problem [13] where all resource demands are known upfront and the objective is to determine the initial provisioning of virtual machines. The bin packing likewise handles a similar resource utilization problem, Falkernauer et al. [25] proposed a function to calculate the efficient utilization of a bin's capacity. This objective does not aim to balance the load across all PMs. Instead, it tries to pack VMs onto a subset of PMs efficiently. By maximizing the utilization of PMs, this approach reduces idle capacity and allows the remaining PMs to stay powered off. The resource utilization objective can be expressed as follows:

$$U^{\text{cpu}} = \frac{\sum_{j=1}^{m} \left(\sum_{i=1}^{n} \frac{v_i^{\text{cpu}} \times \mu_{ij}}{p_j^{\text{cpu}}} \right)^2}{m}, \tag{3.8}$$

$$U^{\text{mem}} = \frac{\sum_{j=1}^{m} \left(\sum_{i=1}^{n} \frac{v_i^{\text{mem}} \times \mu_{ij}}{p_j^{\text{mem}}} \right)^2}{m}. \tag{3.9}$$

Hence, two objectives for VMP can be formulated as follows:

$$Minimize \sum_{j=1}^{m} P(U_j^{cpu}), \tag{3.10}$$

$$Maximize\ U = \frac{U^{\text{mem}} + U^{\text{cpu}}}{2}, \tag{3.11}$$

$$s.t. \sum_{i=1}^{n} \mu_{ij} v_i^{\text{cpu}} < p_j^{\text{cpu}}, \quad \forall j \in \{1, 2, \ldots, m\}, \tag{3.12}$$

$$\sum_{i=1}^{n} \mu_{ij} v_i^{\text{mem}} < p_j^{\text{mem}}, \quad \forall j \in \{1, 2, \ldots, m\}. \tag{3.13}$$

Equations 3.12 and 3.13 ensure that there will be no overflow of PM resource capacity. This guarantees that VMs receive sufficient resource allocation, preventing SLA violations.

3.2. Extended Adaptive Large Neighborhood Search

ALNS is a well-known and state-of-the-art metaheuristic algorithm that is utilized to handle numerous problems. The ALNS used in this paper was derived from Erdogdu et al. [9] and adapted to VMP problems. Destroy and Repair operators specified in their work are purposefully selected for the Vehicle Routing Problem. Therefore, five new destroy and repair operators are chosen to use ALNS. The following section briefly overviews these operators.

3.2.1. Destroy Operators

The destroy operators consist of various methods to relax the current solution. According to the destroy operator, either one or more VMs are unassigned to allow repair operators to generate new solutions. If there is an empty PM, it is closed to reduce energy consumption. Five destroy operators used in our implementation:

1. **Random VM Destroy**: Randomly selects an allocated VM and releases the resources to the allocated PM. This allows us to gain diversity for the search.

2. **Random PM Destroy**: Randomly selects a PM and releases all VMs allocated to it. Therefore, it allows for a perturbation in the solution.

3. **CPU-based Destroy**: The operator finds the PM with the worst CPU utilization and removes all allocated VMs. Hence, it gives the solution a chance to allocate VMs better.

4. **Memory-based Destroy**: Similar to the previous operator, searches for PM with the worst memory utilization and removes all allocated VMs. This operator allows us to improve our multi-dimensional resource utilization objective further.
5. **Highest Energy Destroy**: Removes the allocated VMs in the PM with the highest energy consumption.

3.2.2. Repair Operators

After a destruction operator is applied to the current solution, a repair operator must be applied to make it a feasible solution without violating SLA restrictions. Repair operators search for an unallocated VM and allocate it to a PM based on the selected operator. This process is repeated until all VMs are assigned to a PM that satisfies their resource constraints. In case none of the available PMs satisfy the VM resource constraints, a new PM is created, and the VM is allocated to the new PM.

1. **Random Repair**: The operator tries to allocate a VM into a randomly selected PM that can satisfy its constraints.

2. **Greedy Repair**: Inspired from Azizi et al. [14] greedy algorithm, this operator allocates VM to the most power-efficient PM with sufficient resources.

3. **First Fit Decreasing Repair**: PMs are sorted in descending order based on their total remaining resources. Each unallocated VM is then assigned to a PM if it can meet the resource constraints.

4. **Resource Based Best Fit Repair**: This operator searches for the PM with the highest resource utilization after allocating the new VM.

5. **Energy Based Best Fit Repair**: The operator assigns the VM to a PM that results in the lowest energy consumption.

3.2.3. Initialization

A greedy solution generation algorithm is used to generate the initial solution. At first, a randomly selected PM is generated, and VMs are allocated in order to the PM until one VM resource constraint cannot be satisfied. Then, a new randomly selected PM is generated to assign the VM. This is repeated until all VMs are allocated.

3.2.4. Adaptive Large Neighborhood Search

The pseudo-code for the extended ALNS algorithm proposed in this work is given in Algorithm 1. ALNS starts with generating an initial solution using the greedy solution generation algorithm. This current solution is inserted into an archive where nondominated solutions are kept. A loop is utilized to change the weight values of w_1 and w_2 according to following equations:

$$w_1 = w, \tag{3.14}$$

$$w_2 = N_1 - w. \tag{3.15}$$

These values are later used in $Cost$ to calculate a weighted sum of the two objectives as follows:

$$Cost(s) = w_1 \times P(u) + w_2 \times U. \tag{3.16}$$

Until the terminal condition is satisfied, destroy and repair operators are selected according to their probabilities and applied to *sNew*. The archive is updated with this newly generated solution. Equation 3.16 is applied with normalized objective values to calculate a weighted sum to compare *sNew* and *sCurrent*. If the cost of the new solution is greater, it will replace the current solution. In case the new solution has the best cost, it also replaces the best solution. Even when the new solution is worse than the current solution, it could still replace the current solution with the SA acceptance criteria given in Algorithm 1.

The overall time complexity of the proposed ALNS algorithm for the VMP problem is $O(N_1Imn)$, where N_1 is the number of independent runs with different objective function weights, n is the number of VMs, m is the number of PMs, and I is the number of iterations performed until the terminal condition is satisfied. Since a time-based termination condition is employed, the value of I is not predetermined and depends on the time allocated to each problem instance. For example, if T is the total time reserved for the algorithm, I becomes $\frac{T}{N_1mn}$.

Each destroy and repair operator is given a score (s_{opr}) according to their past performance. For every new solution that is accepted as the best solution, their score increases by σ_1. If the new solution is better than the current solution but worse than the best solution, the operator's score is incremented by σ_2. In case the new solution is inferior to the current solution but accepted via SA acceptance criteria, the operator's score is increased by σ_3. After N_2 iteration without improvement on the best solution, probabilities of all operators are updated according to the operator's score (s_{opr}) and number of times used (c_{opr}) as follows:

$$P_{opr}^{t+1} = P_{opr}^{t} \times (1 - r) + r \times \left(\frac{s_{opr}}{c_{opr}}\right). \tag{3.17}$$

All control parameters of the extended ALNS algorithm, together with the values used in our experiments, are summarized in Table 1.

Table 1. Parameters Used in ALNS.

Symbol	Description	Value
N_1	Number of objective function weight loops	10
N_2	Number of iterations for operator scores, and operator probabilities	200
$\sigma_1, \sigma_2, \sigma_3$	Operator scores	5,0,1
r	Roulette wheel operator in adjusting operator probabilities	0.1
h	The cooling rate of SA	0.999

Algorithm 1 Extended ALNS

archive: Nondominated Solutions
sBest, *sCurrent*, *sNew*: Solution
sCurrent ← Initialize Solution
archive.Update(*sCurrent*)
sBest ← *sCurrent*
for $w \leftarrow 0$ to N_1 **do**
  Initialize Operator Probabilities
  sCurrent ← Initialize Solution
  archive.Update(*sCurrent*)
  counter ← 0
  $T \leftarrow 100$
  repeat
    sNew ← *sCurrent*
    Destroy(*sNew*)
    Repair(*sNew*)
    Evaluate(*sNew*)
    archive.Update(*sNew*)
    counter ← *counter* + 1
    if Cost(*sNew*) < Cost(*sCurrent*) **then**
      sCurrent ← *sNew*
      if Cost(*sNew*) < Cost(*sBest*) **then**
        sBest ← *sNew*
        counter ← 0
      end if
    else
      if Random() $< e^{\frac{Cost(sNew)-Cost(sCurrent)}{T}}$ **then**
        sCurrent ← *sNew*
      end if
    end if
  until Terminal Condition Satisfied
  if *counter* = N_2 **then**
    counter ← 0
    Update Operator Probabilities
  end if
  $T \leftarrow T \times h$
end for
return archive

4. Experimental Work

In this section, we present the experiments done to compare the performance of ALNS with other state-of-the-art algorithms. All algorithms are coded in Java programming language to achieve a fair evaluation. All algorithms were executed with a time-based termination condition, where the total allowed runtime was determined by the product of the number of virtual machines and 1000 milliseconds. This configuration guarantees an equal comparison among different algorithms proportional to the instance size. The parameter k in the energy management model Equation 3.6 is fixed at 0.6, following the recommendation of Braiki et al. [6]. All instances are performed on the same platform with an Intel Core i5-11400H CPU @ 2.70 GHz and 16 GB of RAM.

To compare ALNS, we have selected three multi-objective algorithms: SPEA-II [26], NSGA-II [27], and ϵ-NSGAII [28]. All algorithms use the same initialization method mentioned in section 3.2.3. For each problem instance, algorithms are terminated after the time-based terminal condition, and all nondominated solutions are returned as a result. ALNS repeats a terminal condition ten times with different w_1 and w_2 weights. Therefore, for the sake of fairness, ALNS is given $\frac{n}{N_1}$ seconds for each terminal condition. For all three algorithms selected for comparison, one-point crossover with a probability of 0.7, point mutation with a probability of 0.1, and a population size of 50 are applied. In ϵ-NSGAII, n is given as the value of ϵ. To ensure a fair comparison, selected destroy and repair operators were also incorporated into SPEA-II, NSGA-II, and ϵ-NSGA-II to enhance offspring refinement during the evolutionary process. Specifically, the Random VM Destroy, Resource-Based Best Fit Repair, and Energy-Based Best Fit Repair operators explained in Section 3.2 were integrated as local search mechanisms.

To extensively evaluate the performance of the proposed method across different hardware generations, experiments are divided into two groups based on the physical machine configurations used: legacy servers and modern servers. A single set of predefined VM configurations given in Table 2 is used across all experiments. Problem instances with 50, 100, 300, and 500 VMs are created by randomly selecting VMs configurations table until the required instance size is reached. PMs are randomly selected from the corresponding configuration group (legacy or modern) and are powered on dynamically during the placement process as needed to host the selected VMs.

Table 2. VM Configurations.

VM type	CPU	Mem
Micro	500 MIPS	500 MB
Small	1000 MIPS	1000 MB
Medium	1500 MIPS	2000 MB
Large	2000 MIPS	3000 MB

4.1. Legacy Server Experiments

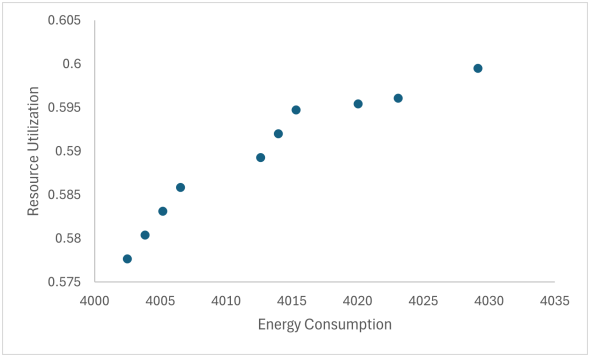
This section presents experimental results found by using HP ProLiant G4 and G5 physical machine configurations, representing older-generation servers with lower resource capacities. G4 and G5 have previously been used in other research [6]. Two types of PM configuration used in legacy problem

instances are presented in Table 3.

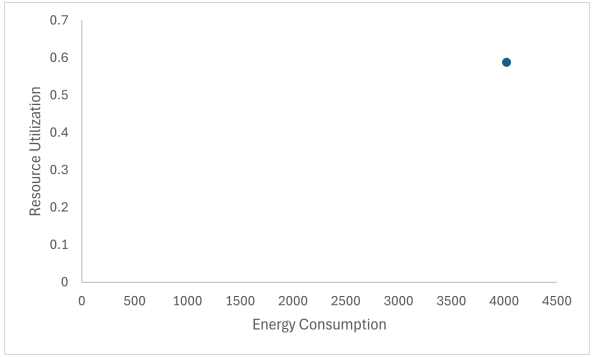
Table 3. Legacy PM Specifications.

PM type	CPU	Mem	P_{max}
HP ProLiant G4	3720 MIPS	4GB	117
HP ProLiant G5	5320 MIPS	4GB	135

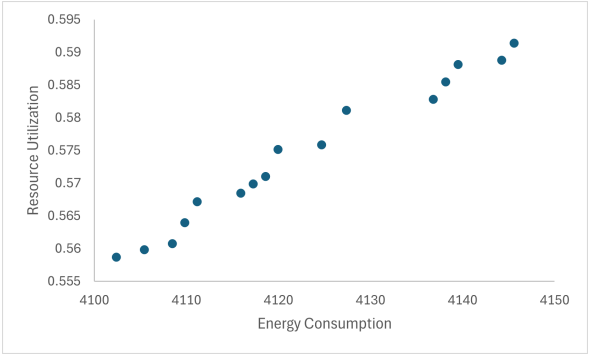
Figures 2, 3, 4, and 5 display the Pareto set found by the four different algorithms on four different sized problem instances. Among all algorithms, NSGA-II found the fewest number of solutions in every problem instance. Moreover, with increasing VM count in problem instances, other algorithms also decreased the number of nondominated solutions found, down to only a few nondominated solutions in the figure 5 500 VM problem instance. In contrast, the ALNS algorithm demonstrates an increasing number of nondominated solutions as the VM count grows in problem instances. ALNS consistently achieves lower power consumption and higher resource utilization, particularly in larger problem instances. These results indicate that, in terms of scalability, ALNS is more effective at maintaining solution diversity and Pareto set quality as the problem size increases.



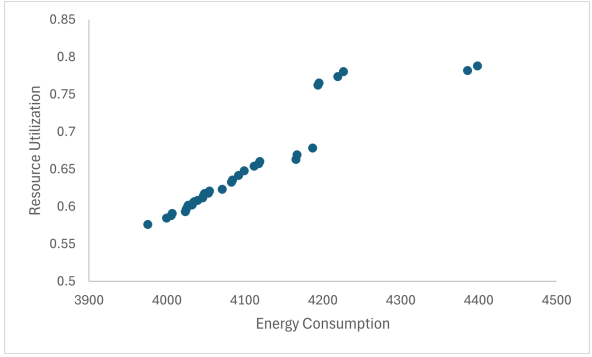
(a) SPEA-II



(b) NSGA-II

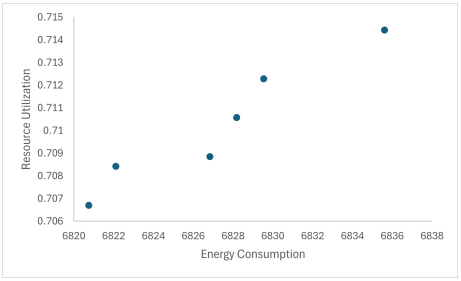


(c) ϵ-NSGAII

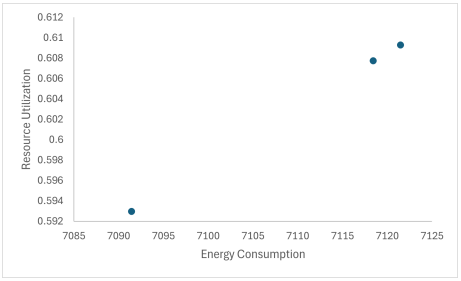


(d) ALNS

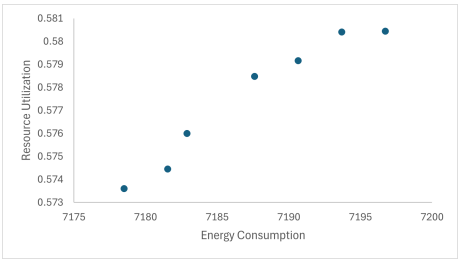
Figure 2. 50 VM problem instance.



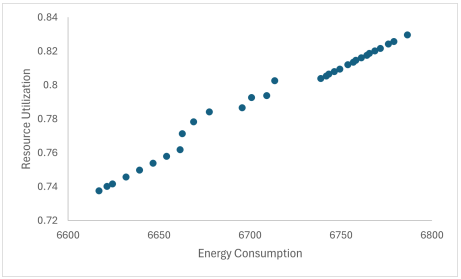
(a) SPEA-II



(b) NSGA-II

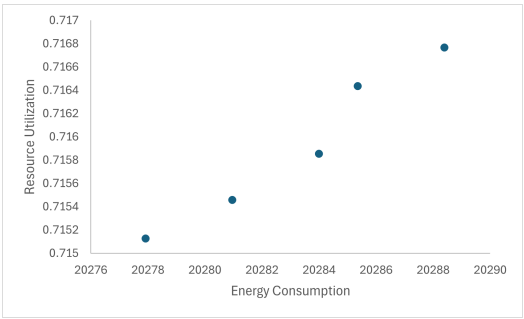


(c) ϵ-NSGAII

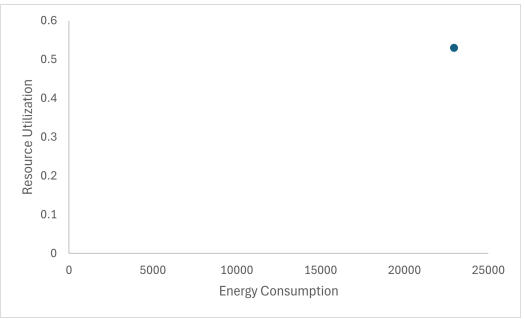


(d) ALNS

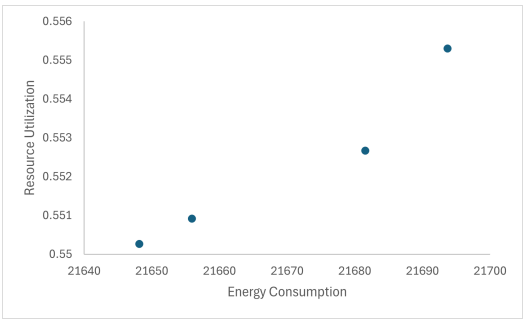
Figure 3. 100 VM problem instance



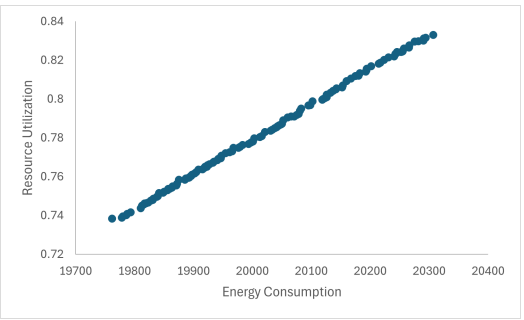
(a) SPEA-II



(b) NSGA-II



(c) ϵ-NSGAII



(d) ALNS

Figure 4. 300 VM problem instance.

(a) SPEA-II

(b) NSGA-II

(c) ϵ-NSGAII

(d) ALNS

Figure 5. 500 VM problem instance.

Table 4. Hypervolume and Number of Solutions on Legacy Configuration.

	NSGA-II		ϵ-NSGAII		SPEA-II		ALNS	
Problem Instance	HV	NoS	HV	NoS	HV	NoS	HV	NoS
50 VM	11.22	1	8.96	16	15.92	10	60.4	34
100 VM	3.31	3	0.06	7	52.88	6	141.08	31
300 VM	0	1	33.02	4	499.08	5	939.47	125
500 VM	311.11	2	0.36	5	346.64	2	886.33	198

Table 4 shows the results of the hypervolume metric for four algorithms in four problem instances. According to Riquelme et al. [29], hypervolume is the most commonly used metric to compare multi-objective optimization algorithms. The hypervolume metric measures the volume enclosed by solutions on the Pareto set and a reference point chosen by the user. It measures the quality and spread of the solutions on the Pareto set. The algorithm with a higher hypervolume value produces a better Pareto set. In our examples, the reference point is selected by combining the worst values found from all algorithms for both objectives, therefore guaranteeing every solution in the Pareto sets will dominate it. According to the results, although SPEA-II generates a relatively low number of nondominated solutions, it achieves the second highest hypervolume across all problem instances, outperformed only by ALNS. NSGA-II and ϵ-NSGA-II display a more variable performance, with NSGA-II clearly surpassing ϵ-NSGA-II only in the largest problem instance. Overall, ALNS consistently achieves the best hypervolume scores, attributed to its destroy-repair cycle that balances exploration and exploitation. Furthermore, as a single-solution based search algorithm, ALNS does not rely on maintaining a population, making it more scalable and computationally efficient than population-based methods.

4.2. Modern Server Experiments

This section focuses on experimental results using modern server configurations, specifically HP ProLiant G9 and G10 physical machines. Newer-generation servers offer improved CPU and memory capacities along with enhanced energy efficiency. Similar to the previous experiments, Table 5 lists two types of PM configurations that are used in modern problem instances.

Table 5. Modern PM Specifications.

PM type	CPU	Mem	$P_{\max}$
HP ProLiant G9	82800 MIPS	64GB	276
HP ProLiant G10	70000 MIPS	48GB	233

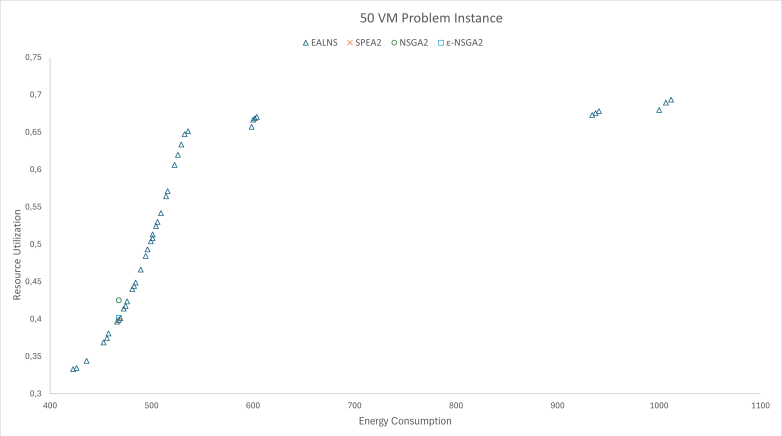


Figure 6. 50 VM problem instance.

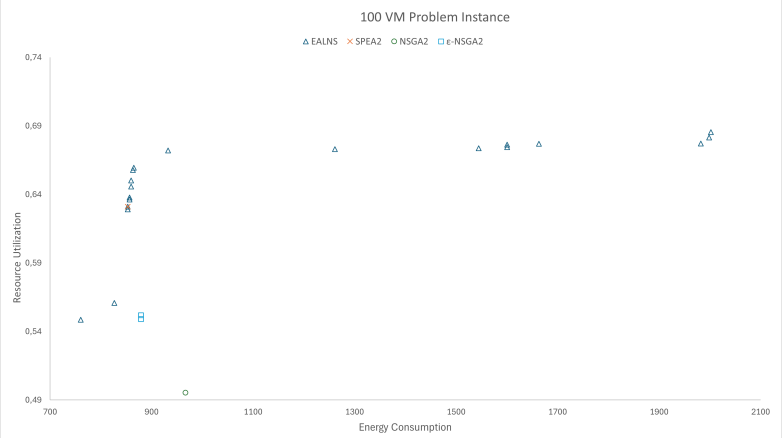


Figure 7. 100 VM problem instance.

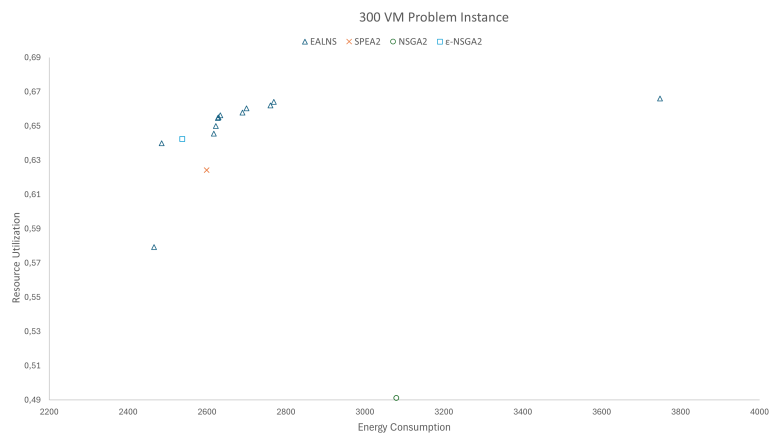


Figure 8. 300 VM problem instance.

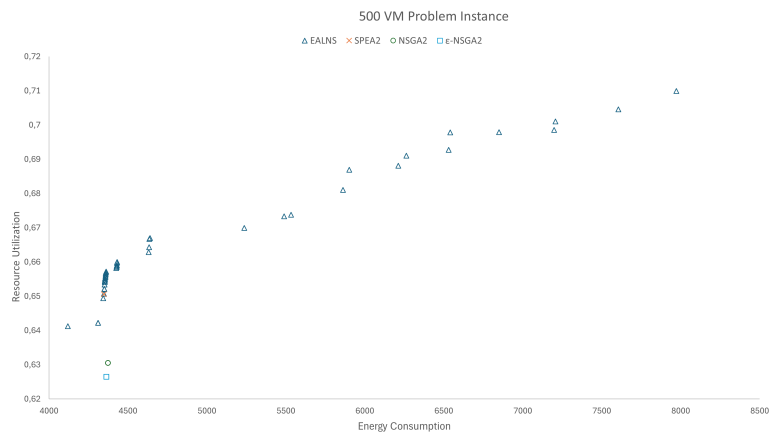


Figure 9. 500 VM problem instance.

The results of the experiment for modern server configurations are presented in Figures 6, 7, 8, and 9. These configurations more accurately represent real-world data centers. In these scenarios, NSGA-II, ϵ-NSGAII, and SPEA-II struggle to generate more than one nondominated solution. In contrast, the ALNS algorithm maintains good quality and diversity of non-dominated solutions even under the increased capacity and complexity of modern physical machines. This highlights the robustness of ALNS in handling varying hardware configurations. Furthermore, the scalability of the ALNS algorithm is apparent not only in the legacy configuration, where it successfully generates a high number of nondominated solutions with increasing problem sizes, but also in the modern configuration, where it demonstrates similarly strong performance even in the 500 VM problem instance.

Table 6. Hypervolume and Number of Solutions on Modern Configuration.

	NSGA-II		ϵ-NSGAII		SPEA-II		ALNS	
Problem Instance	HV	NoS	HV	NoS	HV	NoS	HV	NoS
50 VM	50.08	1	37.44	1	35.69	1	173.13	41
100 VM	0	1	63.50	2	155.93	1	209.48	19
300 VM	0	1	182.99	1	152.80	1	215.89	12
500 VM	14.51	1	0	1	87.43	1	213.73	43

Table 6 presents the hypervolume results obtained from experiments conducted with modern server configurations. The ALNS outperforms all other comparison algorithms across all problem instances with modern configurations in terms of hypervolume, demonstrating both quality and diversity of the obtained Pareto sets. Due to NSGA-II and ϵ-NSGAII often generating only a single nondominated solution, their hypervolume is reported as zero in cases where that solution lies at the reference point in the objective space, offering no dominated volume.

5. Conclusion

This study looks at the static VMP setting where the suggested ALNS algorithm can be added to VM provisioning modules, especially when dealing with large workloads in batches. The research adapts the state-of-the-art ALNS algorithm to solve the multi-objective VMP problem, focusing on energy management and resource utilization. To our knowledge, ALNS was never utilized in VMP to create a Pareto set of solutions. The ALNS algorithm adapted to VMP with five destroy and five repair operators to generate new solutions by relaxing the current solution and then trying to repair it via different operators. These operators are all selected adaptively according to their performance on runtime. Probabilities for choosing an operator are updated after a predetermined number of non-improvement iterations on the best solution. A weight value is used to run the algorithm with different weights to increase the spread of solutions on the Pareto set. The experimental work was done with an equal time limit for all algorithms to provide a fair competition. Experimental results in both legacy and modern server configurations show that ALNS significantly improves the quality and diversity of the Pareto set compared to state-of-the-art algorithms such as SPEA-II, NSGA-II, and ϵ-NSGAII, demonstrating its robustness and adaptability across varying PM capabilities and configurations.

Several directions remain open for future research. First, while the current ALNS-based algorithm demonstrates reasonable scalability up to 500 virtual machines, addressing larger problem sizes, such as those in hyperscale data centers, will require further enhancements. One possible solution is to develop a parallel version of ALNS to accelerate convergence and support real-time responsiveness at scale. Second, the existing multi-objective problem formulation consists of two objectives, which can be extended to support a more comprehensive optimization model. Finally, future work may explore the adaptation of the proposed solution to dynamic VMP settings, where VM arrivals, terminations, migrations, physical machine failures, and load fluctuations must be handled in real time. Such an extension would enable practical deployment in dynamic cloud resource management scenarios.

Author contributions

Both authors contributed to the conception and design of the study and to the methodology. Tolga Bugra Altuntas implemented the software, carried out the experiments, performed the formal analysis, and prepared the original draft. Dindar Öz supervised the work, validated the results, and reviewed and edited the manuscript. Both authors have read and approved the final version of the manuscript for publication.

Acknowledgments

The authors would like to thank the editor and reviewers for their valuable comments and suggestions, which helped improve the quality and clarity of this manuscript. This research received no external funding.

Use of generative-AI tools declaration

The authors used generative-AI tools only for language editing, and improving the clarity of the manuscript during the revision stage. The authors reviewed, edited, and verified all AI-assisted content and take full responsibility for the final content of the article.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Y. Qin, H. Wang, S. Yi, X. Li, L. Zhai, Virtual machine placement based on multi-objective reinforcement learning, *Appl. Intell.,* **50** (2020), 2370–2383. https://doi.org/10.1007/s10489-020-01633-3
2. S. Yang, P. Wieder, R. Yahyapour, S. Trajanovski, X. Fu, Reliable Virtual Machine Placement and Routing in Clouds, *IEEE Trans. Parallel Distrib. Syst.,* **28** (2017), 2965–2978. https://doi.org/10.1109/TPDS.2017.2693273
3. S. Kumaraswamy, N. Mydhili, Bin packing algorithms for virtual machine placement in cloud computing: A review, *Int. J. Electr. Comput. Eng.,* **9** (2019), 512–524. https://doi.org/10.11591/ijece.v9i1.pp512-524
4. T. Tlili, S. Krichen, Best Fit Decreasing Algorithm for Virtual Machine Placement Modeled as a Bin Packing Problem, in *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, IEEE, Rome, Italy, 2023, 1261–1266. https://doi.org/10.1109/CoDIT58514.2023.10284347

5. R. S. Moorthy, U. Fareentaj, T. K. Divya, An Effective Mechanism for Virtual Machine Placement using Aco in IAAS Cloud, *IOP Conf. Ser.: Mater. Sci. Eng.,* **225** (2017), 012227. https://doi.org/10.1088/1757-899X/225/1/012227
6. K. Braiki, H. Youssef, Multi-Objective Virtual Machine Placement Algorithm Based on Particle Swarm Optimization, in *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, IEEE, Limassol, 2018, 279–284. https://doi.org/10.1109/IWCMC.2018.8450527
7. J. Lu, W. Zhao, H. Zhu, J. Li, Z. Cheng, G. Xiao, Optimal machine placement based on improved genetic algorithm in cloud computing, *J. Supercomput.,* **78** (2022), 3448–3476. https://doi.org/10.1007/s11227-021-03953-8
8. Y. Gao, H. Guan, Z. Qi, Y. Hou, L. Liu, A multi-objective ant colony system algorithm for virtual machine placement in cloud computing, *J. Comput. Syst. Sci.,* **79** (2013), 1230–1242. https://doi.org/10.1016/j.jcss.2013.02.004
9. K. Erdoğdu, K. Karabulut, Bi-objective green vehicle routing problem, *Int. Trans. Oper. Res.,* **29** (2022), 1602–1626. https://doi.org/10.1111/itor.13044
10. E. Demir, T. Bektaş, G. Laporte, An adaptive large neighborhood search heuristic for the Pollution-Routing Problem, *Eur. J. Oper. Res.,* **223** (2012), 346–359. https://doi.org/10.1016/j.ejor.2012.06.044
11. S. T. W. Mara, R. Norcahyo, P. Jodiawan, L. Lusiantoro, A. P. Rifai, A survey of adaptive large neighborhood search algorithms and applications, *Comput. Oper. Res.,* **146** (2022), 105903. https://doi.org/10.1016/j.cor.2022.105903
12. S. Ropke, D. Pisinger, An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows, *Transp. Sci.,* **40** (2006), 455–472. https://doi.org/10.1287/trsc.1050.0135
13. S. Challita, F. Paraiso, P. Merle, A Study of Virtual Machine Placement Optimization in Data Centers, in *Proceedings of the 7th International Conference on Cloud Computing and Services Science (CLOSER)*, SciTePress, Porto, Portugal, 2017, 343–350. https://doi.org/10.5220/0006236503430350
14. S. Azizi, M. Shojafar, J. Abawajy, R. Buyya, GRVMP: A Greedy Randomized Algorithm for Virtual Machine Placement in Cloud Data Centers, *IEEE Syst. J.,* **15** (2021), 2571–2582. https://doi.org/10.1109/JSYST.2020.3002721
15. J. Wang, J. Yu, R. Zhai, X. He, Y. Song, GMPR: A Two-Phase Heuristic Algorithm for Virtual Machine Placement in Large-Scale Cloud Data Centers, *IEEE Syst. J.,* **17** (2023), 1419–1430. https://doi.org/10.1109/JSYST.2022.3187971
16. K. Braiki, H. Youssef, Fuzzy-logic-based multi-objective best-fit-decreasing virtual machine reallocation, *J. Supercomput.,* **76** (2020), 427–454. https://doi.org/10.1007/s11227-019-03029-8
17. C. Bhatt, S. Singhal, Multi-objective reinforcement learning for virtual machines placement in cloud computing, *Int. J. Adv. Comput. Sci. Appl.,* **15** (2024). https://doi.org/10.14569/IJACSA.2024.01503105
18. D. Pisinger, S. Ropke, A general heuristic for vehicle routing problems, *Comput. Oper. Res.,* **34** (2007), 2403–2435. https://doi.org/10.1016/j.cor.2005.09.012

19. A. P. Rifai, H. T. Nguyen, S. Z. Md Dawal, Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling, *Appl. Soft Comput.,* **40** (2016), 42–57. https://doi.org/10.1016/j.asoc.2015.11.034

20. W. Liao, L. Zhang, Z. Wei, Multi-objective green meal delivery routing problem based on a two-stage solution strategy, *J. Clean. Prod.,* **258** (2020), 120627. https://doi.org/10.1016/j.jclepro.2020.120627

21. J. Dong, H. Wang, S. Cheng, Energy-performance tradeoffs in IaaS cloud with virtual machine scheduling, *China Commun.,* **12** (2015), 155–166. https://doi.org/10.1109/CC.2015.7084410

22. Z. Tang, Y. Mo, K. Li, K. Li, Dynamic forecast scheduling algorithm for virtual machine placement in cloud computing environment, *J. Supercomput.,* **70** (2014), 1279–1296. https://doi.org/10.1007/s11227-014-1227-5

23. R. Basmadjian, P. Bouvry, G. Da Costa, L. Gyarmati, D. Kliazovich, S. Lafond, et al., Green Data Centers, in *Large-Scale Distributed Systems and Energy Efficiency*, John Wiley & Sons, 2015, 159–196. https://doi.org/10.1002/9781118981122.ch6

24. A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing, *Future Gener. Comput. Syst.,* **28** (2012), 755–768. https://doi.org/10.1016/j.future.2011.04.017

25. E. Falkenauer, A hybrid grouping genetic algorithm for bin packing, *J. Heuristics,* **2** (1996), 5–30. https://doi.org/10.1007/BF00226291

26. E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the Strength Pareto Evolutionary Algorithm, *TIK-Report 103*, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, 2001. https://doi.org/10.3929/ethz-a-004284029

27. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. Evol. Comput.,* **6** (2002), 182–197. https://doi.org/10.1109/4235.996017

28. V. Devireddy, P. Reed, Efficient and reliable evolutionary multiobjective optimization using ϵ-dominance archiving and adaptive population sizing, in *Genetic and Evolutionary Computation – GECCO 2004*, Lecture Notes in Computer Science, vol. 3103, Springer, Berlin, Heidelberg, 2004, 390–391. https://doi.org/10.1007/978-3-540-24855-2_38

29. N. Riquelme, C. Von Lücken, B. Baran, Performance metrics in multi-objective optimization, in *2015 Latin American Computing Conference (CLEI)*, IEEE, Arequipa, Peru, 2015, 1–11. https://doi.org/10.1109/CLEI.2015.7360024

AIMS Press