Hindawi
International Journal of Intelligent Systems
Volume 2023, Article ID 8378850, 16 pages
https://doi.org/10.1155/2023/8378850

Research Article

Virtual Machine Replica Placement Using a Multiobjective Genetic Algorithm

Marwa F. Mohamed,[1] **Mai Dahshan**,[2] **Kenli Li**,[3] **and Ahmad Salah**[4,5]

[1]*Department of Computer Science, Faculty of Computers and Informatics, Suez Canal University, Ismailia 41522, Egypt*
[2]*School of Computing, University of North Florida, Jacksonville, Florida, USA*
[3]*College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan, China*
[4]*Department of Computer Science, College of Computers and Informatics, Zagazig University, Sharkia, Egypt*
[5]*Department of Information Technology, College of Computing and Information Sciences, University of Technology and Applied Sciences, Ibri, Ad-dhahira, Oman*

Correspondence should be addressed to Ahmad Salah; ahmad@zu.edu.eg

Received 22 February 2023; Revised 29 May 2023; Accepted 3 June 2023; Published 28 June 2023

Academic Editor: Mohammad R. Khosravi

Virtual machine (VM) replication is a critical task in any cloud computing platform to ensure the availability of the cloud service for the end user. In this task, one primary VM resides on a physical machine (PM) and one or more replicas reside on separate PMs. In cloud computing, VM placement (VMP) is a well-studied problem in terms of different goals, such as power consumption reduction. The VMP problem can be solved by using heuristics, namely, first-fit and meta-heuristics such as the genetic algorithm. Despite extensive research into the VMP problem, there are few works that consider VM replication when choosing a VMP. In this context, we proposed studying the problem of optimal VMP considering VM replication requirements. The proposed work frames the problem at hand as a multiobjective problem and adapts a nondominated sorting genetic algorithm (NSGA-III) to address the problem. VM replicas' placement should consider several dimensions such as the geographical distance between the PM hosting the primary VM and the other PMs hosting the replicas. In addition, to this end, the proposed model aims to minimize (1) power consumption, (2) performance degradation, and (3) the distance between the PMs hosting the primary VM and its replica(s). The proposed method is thoroughly tested on a variety of computing environments with various heterogeneous VMs and PMs, including compute-intensive and memory-intensive environments. The obtained results illustrate the performance disparity between the adapted NSGA-III and MOEA/D methods and other methods of comparison, including heuristic and meta-heuristic approaches, with NSGA-III outperforming other comparison methods. For instance, in memory-intensive and in heterogeneous environments, the NSGA-III method's performance was superior to the first-fit, next-fit, best-fit, PSO, and MOEA/D methods by 58%, 62%, 64%, 55%, and 31%, respectively.

1. Introduction

Virtualization is a convenient technology in which physical resources (e.g., CPU, memory, and disk space) can be partitioned over one or more machines through partial or full machine simulation. To maintain availability and increase the performance of VMs, VM replication (VMR) is a technique that duplicates VMs and places replicas in multiple PMs of cloud data centers. If one replica fails, the client's requests will be fulfilled by another replica. VM replication increases efficiency by distributing incoming requests across VM replicas in various PMs, hence balancing and decreasing PM loads [1]. VM replication has been adopted by well-known data centers, namely, Amazon Simple Storage Service (Amazon S3) and Microsoft Azure.

There are several challenges to be considered when addressing the problem of VM replicas' placement. First, multiple duplicates of the same VM should be installed in many PMs (i.e., VM fault-tolerance constraint). Unfortunately, this accelerates the rise in energy usage in cloud

data centers [2]. By decreasing the number of active PMs and shutting down inactive PMs, it is possible to save energy and cut emissions. Consequently, minimizing PM energy consumption is crucial for reducing the overall power consumption of a data center [3].

Second, selecting one replica as the primary replica, which is responsible for processing requests from incoming clients and returning responses, in addition to returning responses, it transmits updates to other replicas [4]. Consequently, the geographical distance between VM replicas has a significant impact on the performance of applications. Moreover, the distance between VM replicas affects the energy required for data transmission. The greater the distance between replicas, the more transmission energy is required. Replicas located further from the primary replica will consume more energy than the replicas closer to the primary replica [5].

A third issue with VM replication is that as more VM replicas are built and deployed in cloud data centers, more resources (such as CPUs, RAM, and I/Os) are utilized; however, PMs have a finite amount of resources. In addition, the diversity of hosted client programs presents additional issues; certain client apps may be memory intensive, while others may be computation-intensive [6]. A fourth issue is connected to the service-level agreement (SLA) [7]. If the service provider fails to provide the required service as outlined in the SLA contract, the client is eligible to receive credits. VM replication is a method for avoiding SLA violations; therefore, it should be considered placing replicas in PMs with the same or comparable performance.

Meta-heuristic algorithms are one of the most effective techniques to overcome the prior obstacles in VM replication placement. Meta-heuristic algorithms have demonstrated countless successes in numerous VM placement studies. Many-objective optimization problems (MaOPs) are optimization problems with more than three objectives [8]. Multiobjective evolutionary algorithms based on decomposition (MOEA/D), particularly the basic NSGA [9] and its modifications, are commonly used to address these difficulties in a wide range of industries. Deb and Jain [10] proposed the NSGA-III algorithm, which uses reference points to replace the NSGA-II-congested distance. The NSGA-III algorithm outperforms the NSGA-II in terms of performance. In fact, the NSGA-III and MOEA algorithms are widely recognized as the best extant many-objective evolutionary algorithms (MaOPs) [11].

In this context, the discussed four issues of the VM replicas' placement problem motivated this work to frame this problem as an optimization problem. Moreover, another motivation for this work was the minimal amount of research undertaken in this area. Due to the superior performance of multiobjective problems, it is proposed to frame this problem as a multiobjective optimization problem and addressing it with the well-known NSGA-III algorithm. In addition, the proposed method was evaluated utilizing a variety of scenarios, including compute-intensive and memory-intensive contexts for cloud computing. The following is a summary of the key contributions of this work:

(1) To our knowledge, this is the first work that considers the problem of VMP with VM replication with the aim of minimizing three objectives, namely, (1) power consumption, (2) performance degrading due to switching to a replica with specification lower than the user's needs, and (3) distance between VM replicas.

(2) The NSGA-III method was adapted to address the problem at hand with a proposed method for repairing infeasible solutions.

(3) Datasets were suggested to evaluate the proposed work with different scenarios.

(4) The obtained results show that the proposed adapted NSGA-III method outperformed the other methods of comparison by a huge margin. Of note, reducing the number of replicas causing performance degrading was the most improved objective, as the improvement is an order of magnitude.

The rest of the paper is organized as follows. Section 2 discussed the related work. Section 3 explains in detail the problem. Section 4 presents a methodology of the problem. Evaluation of the proposed algorithm is presented in Section 5. Finally, the paper is concluded in Section 6.

2. Related Work

Multiple objectives have been examined in relation to the VMP optimization problem. VMP's primary objective is to minimize power consumption, which is a well-studied problem. Other objectives, such as fault tolerance, SLA violations, and distance awareness between VMs, have received less research effort.

Multiple approaches have been developed to solve the VMP optimization problem. Al-Moalmi et al. [12] proposed a computational intelligence method to reduce the number of active hosts and energy consumption. Mirjalili et al. introduced GWO-VMP with grey wolf optimization (GWO) for reducing the number of active physical machines that are used to host a set of VMs. Alharbi et al. [13]

Parvizi and Rezvani [14] developed a multiobjective virtual machine placement (MO-VMP) problem in order to reduce power consumption, the number of active PMs, and total resource wastage. The problem is presented as a nonlinear convex optimization problem. Then, NSGA-III is applied to minimize the MO-VMP time complexity. Tarahomi et al. [15] proposed a microgenetic algorithm to reduce energy consumption, VM migration, SLA violation, and the number of server shut down. Abbasi-khazaei and Rezvani [16] designed a multiobjective VM placement that aims to reduce the energy cost and the cost of producing carbon dioxide.

Alresheedi et al. [17] proposed an alternative multiobjective optimization (MOP) technique that combines the salp swarm and sine-cosine algorithms (MOSSASCA) for selecting a suitable virtual machine placement solution. MOSSASCA aims to increase the mean time before a host shutdown, reduce power consumption, and eliminate

service-level agreement breaches. Using a MOP technique, the proposed approach improves the salp swarm algorithm (SSA) and sine-cosine algorithm (SCA). Utilizing a local search approach, the SCA enhances the performance of the traditional SSA by preventing trapping in a local optimal solution and accelerating convergence. Zhang et al. [18] proposed an approach based on a heuristic ant colony technique. Multiple factors, including the service-level agreement violation rate, resource remaining rate, power consumption rate, failure rate, and fault tolerance cost, are used to construct a model for the initial fault-tolerant placement of VMs in cloud systems' star-topological data centers.

Sharma et al. [19] proposed an energy-efficient approach for VM consolidation. It considers the reliability factor before combining the running VMs to conserve energy in a cloud computing environment prone to failure. Fault tolerance is achieved through the use of both reactive (checkpointing) and proactive (VM migration) techniques. The activation of fault tolerance systems relies on failure prediction based on time series analysis.

Khani et al. [20] aimed to reduce total power usage in data centers by studying the virtual machine replication (VMR) problem. It replicates R copies of each VM and places them on separate PMs in the data centers to ensure that each VM is accessible in the event of a server failure, where R is tuned by the probability of server failure. The authors show that VMR is the same as the least cost flow issue; thus, it can be solved quickly and effectively. The solution can further minimize data center power usage by unifying PMs that hold VM clones and shutting off inactive machines.

Gonzalez and Tang [2] proposed a new fault-tolerant VM placement problem (FT-VMP) to arrange the required number of VM replica copies in cloud data centers. The proposed technique aims to lower the total number of active PMs while meeting VM fault tolerance, VM compatibility with PMs, and PM resource capacity restrictions. Yao et al. [21] proposed a network-aware VM allocation algorithm based on the maximum clique algorithm (MCNVMA) for VMs in the cloud data center. In MCNVMA, each VM can have its unique configuration requirements. In addition, the cost of communication between VMs, which can be gathered, is a necessity. This algorithm searches for solutions that fulfill the user's requirements and provide a short communication path across VMs. According to test findings, the proposed algorithm lowers the cost of communication between VMs, especially in big data centers.

As listed in Table 1, the work proposed in [21] is the only proposed study that takes into account the distance between VMs during the placement procedure. However, energy conservation, fault tolerance, and SLA violation are not considered objectives by the authors. The last row of Table 1 shows the proposed work to bridge the gap of combining the mentioned three objectives as a multiobjective problem. Then, this optimization problem is proposed to be addressed using the adapted NSGA-III algorithm.

3. Problem Definition

Virtualization is an important technology that supports cloud computing and offers significant benefits from PM consolidation. A single PM may host several VMs, each operating independently. VMP aims to reduce cloud computing operating costs and make better use of PMs' resources. VMP maximizes resource utilization by grouping VMs into one or more PMs. It aims to reduce the power usage of the cloud data center. In addition, VMP offers VM replication for fault tolerance and to prevent SLA violations. In this paper, a VMP strategy is proposed to reduce (1) power consumption, (2) performance degradation due to switching to a replica, and (3) the distance between VM replicas. The list of symbols is listed in Table 2.

This paper considers the initial placement or static VM placement algorithms. It submits several VMs M and their replicas R, which are predefined, to a set of fully empty PMs P, where P_i represents the i^{th} PM, $i \in [1, N]$, and $P_i \in P$. Similarly, V is a collection of VMs, where V_j^k represents the j^{th} VM and its k^{th} replica, $j \in [1, M]$, and $k \in [0, R]$, if $k = 0$; it then becomes the primary VM. In addition, an assumption is made that the most important resource requirements of PMs are the CPU, clock rate, and main memory resources. Minimizing the performance degrading objective is significant for avoiding SLA violations. The CPU and memory requirements of PM_i are presented as P_{cpui} and P_{rami}, respectively. Likewise, the requirement of CPU and memory of V_j^k is presented as $V_{\text{cpu}j}^k$ and $V_{\text{ram}j}^k$, respectively. Table 1 summarizes the key parameters and variables used in the proposed model.

The proposed model has four constraints and three objectives. The first constraint is the OS constraint, which requires matching the VM requirements of the host OS with the host PM OS when placing a VM in a PM. The same approach proposed in [22] has been utilized in resolving this issue. It reduces the search space by treating VMs with comparable OS needs as a separate issue. The cloud provider intuitively knows the ratio of VMs based on their OS needs. By this ratio, the cloud provider distributes the PMs among the VMs.

The second constraint is CPU constraint (equation (1)), which ensures that the total consumed capacity of a set of VMs does not exceed the CPU capacity of the host's CPU $P_{\text{cpuCapacity}_i}$.

$$\sum_{j=1,k=0}^{M,R} V_{\text{cpu}_j}^k \leq P_{\text{cpuCapacity}_i}, \tag{1}$$

where $V_{\text{cpu}j}^k$ is the CPU capacity of V_j^k.

The third constraint is the RAM constraint (equation (2)); it ensures that the total consumed capacity of a set of VMs does not exceed the RAM capacity of the host's RAM $P_{\text{ramCapacity}_i}$.

Table 1: A summary of the related work.

Ref.	VMP	VMR	Utilized technique	Objectives
[12]	✓	✗	Grey wolf optimization (GWO)	Minimize the number of active servers that are used to host the virtual machines (VMs)
[13]	✓	✗	Ant colony system (ACS)	Minimize the energy consumption
[15]	✓	✗	Microgenetic	Minimize energy consumption, VM migration, SLA violation, and number of server shutdown
[14]	✓	✗	NSGA-III	Minimize resource loss, energy consumption, and the number of active PM
[16]	✓	✗	Modified memetic algorithm (MA)	Minimize the energy cost and the cost of producing carbon dioxide
[17]	✓	✗	Combines the salp swarm and sine-cosine algorithms (MOSSASCA)	Maximize mean time before a PM shutdown (MTBHS) and minimize power consumption and SLA violations
[18]	✓	✓	Ant colony	Minimize SLA violation, minimize resource wasting, power consumption, and guarantee fault tolerance
[19]	✗	✓	Heuristic algorithms	Minimize the total power consumption and improve VM reliability and availability
[20]	✓	✓	Heuristic algorithms	Minimize the total power consumption and guarantee VM availability
[2]	✓	✓	Integer linear programming (ILP)-based algorithm	Minimize the number of PMs and guarantee fault-tolerant
[21]	✓	✗	Heuristic algorithms based on the maximum clique algorithm	Minimize the communication cost, latency between servers, and communication bandwidths
The proposed work	✓	✓	NSGA-III	Minimize the power consumption, performance degrading due to switching to a replica with specification lower than the user's needs, and the distance between VM replicas

Table 2: Summary of the symbols that are used in our model.

M	Number of VMs
r	Number of replica for individual VM
R	Total number of replicas for all VM, $R = \sum_{i=1}^{M} r_i$
V	It represents a set of VM
T	It represents $M + R$
V_j^k	It represents the j^{th} VM and k^{th} replica. $j \in [1, M]$ and $\in [0, R]$, if $k = 0$, then its the primary VM
$V_{\text{cpu}j}^k$	The CPU capacity of V_j^k
$V_{\text{ram}j}^k$	The memory capacity of V_j^k
N	Number of PMs
n	Number of PMs hosted by VMs
P	It represents a set of PM
P_i	It represents the i^{th} PM, $i \in [1, N]$
$P_{\text{cpu}i}$	The current CPU usage of $P_{\text{cpu}i}$ is equal to the sum of all V_{cpu} in $P_{\text{cpu}i}$
$P_{\text{ram}i}$	The current memory usage of $P_{\text{ram}i}$ is equal to the sum of all V_{ram} in $P_{\text{ram}i}$
$P_{\text{cpuCapacity}i}$	The CPU capacity of P_i
$P_{\text{ramCapacity}i}$	The memory capacity of P_i
S	Solution
S_b	The best solution
gen	Number of generation
pop	Number of population

$$\sum_{j=1,k=0}^{M,R} V_{\text{ram}j}^k \leq P_{\text{ramCapacity}_i}, \tag{2}$$

where $V_{\text{ram}j}^k$ is the RAM capacity of V_j^k.

The last constraint (equations (3) and (4)) restricts the placement of the VM and its replicas on the same PM.

$$\forall\left(V_a^c, V_b^d\right) \text{ if a = b, then b is the replica of a and c} \neq \text{d}, \tag{3}$$

$$\text{if } V_a^c \in P_i \text{ then } V_{b^i}^d, \tag{4}$$

where a and b are integer numbers $\in [0, M]$ and c and d are integer numbers $\in [0, R]$.

The main objectives of the proposed model are to minimize the number of PMs (power consumption) $f(e)$ (equation (5)), the distance between the VM and its replicas $f(d)$ (equation (6)), and the performance degradation caused by the PM hosting the replica having lower capabilities (e.g., a lower core speed) $f(c)$ (equation (7)). In this model, it is assumed that the three objectives are of same importance. Thus, the weights of all objectives should be equal.

$$\min f(e) = \sum_{i=1}^{N} P_i, \tag{5}$$

$$\min f(d) = \sum_{j=1}^{M} \sum_{k=1}^{R} \left(\left|V_j^0 - V_j^k\right|\right), \tag{6}$$

$$\min f(c) = \sum_{j=1}^{M} \sum_{k=1}^{R} \begin{cases} 1, & \text{if PM hosting is } V_j^k \text{ less powerful than the PM hosting } V_j^0, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

4. The Proposed NSGA-III Method

In this paper, the NSGA-III method was utilized to solve the multiobjective optimization problem (equations (5), (6), and (7)). The implementation details of NSGA-III are explained in [23]. The NSGA-III algorithm is similar to other genetic algorithms in that it employs sampling, crossover, and mutation processes. In this context, it is proposed using random integer sampling, half uniform crossover, and polynomial mutation techniques for these operations [24].

4.1. Solution Construction. The initial solutions of the NSGA-III are generated at random. Each solution is represented as a one-dimensional array with a length of T,

where $T = M + R$. The array indices reflect the identifiers of VMs, while the array values represent the identifiers of PMs. These indices control the placement of each VM within a PM. A PM can host several VMs, but a VM can only be hosted by one PM. The array values are in the range $[0, N-1]$, where N is the number of existing PMs.

Figure 1 illustrates an example of three solutions, where 7 VMs are assigned to 4 PMs. Three out of seven VMs are replicas ($R = 3$); V_4 and V_5 are replicas of V_1 and V_6 is the only replica of V_3. V_0 and V_2 have no replicas. The first solution $S1$ is an infeasible solution since the VM_1 and its replica VM_5 are placed on the same PM, i.e., PM_1. The second solution $S2$ is a feasible solution for the reason that it meets all the proposed constraints (e.g., RAM, CPU, and replication constraints). However, it is not a recommended solution, as it does not minimize the problem objectives (i.e., $f(e)$, $f(d)$, and $f(c)$). The third solution $S3$ is an optimal solution due to the fact that it produces the minimum $f(e)$, $f(d)$, and $f(c)$ values. In the next subsection, it will be explained how $f(e)$, $f(d)$, and $f(c)$ are calculated in detail.

4.2. Population Updating and Repairing. As depicted in Figure 2, the model parameters are initially assigned (e.g., the number of generations gen and the population size pop). The NSGA-III algorithm then generates solutions at random, which may or may not be feasible. All infeasible solutions should be modified to meet the problem's constraints. After that, for all feasible solutions, the fitness or objective functions are evaluated to determine the best solution S_b. When the model generates the gen generations, the best solution S_b is returned.

Algorithm 1 converts an infeasible solution into a feasible solution through the following steps: First, it finds the VM that has been incorrectly allocated to the PM, and then it unassigned the VM to the PM. Second, it allocates the unassigned VM into a valid PM. Notably, there are two cases for VMs: (1) a VM with replica(s) and (2) a VM without replica(s). In the first case, when the VM has replicas, it is placed in the valid PM that has the minimum distance from its replica. In the second case, when the VM has no replica, the proposed algorithm selects a valid PM that has the highest core capacity.

After correcting the infeasible solutions, the objective functions are computed to select the optimal solutions S. The optimal solutions are chosen by minimizing the number of PMs, the distance between the VM and its replica, and the degradation of SLA. Algorithm 2 evaluates the first objective by counting the number of unique PMs in solution S. For example, the number of distinct PMs of $S1$ is 4 (0, 1, 2, and 3), $S2$ is 4 (0, 1, 2, and 3), and $S3$ is 3 (3, 0, and 2), as depicted in Figure 1.

The second objective is evaluated by Algorithm 3, which calculates the total distance between the primary VM host and its replica host. For example, the total distance between the primary VM host and the replicas (host) of $S2$ is 38 (Figure 1), which is equivalent to the sum of the following distances (10, 8, and 20). The value 10 is the distance between P_1 and P_0, which are the hosts of V_1 and its replica host V_4. The value 8 is the distance between P_1 and P_2, which are the hosts of V_1 and its replica host V_5. The value 20 is the distance between P_3 and P_1, which are the hosts of V_3 and its replica host V_6.

The third objective is evaluated by Algorithm 4, which counts the number of times the performance can potentially degrade. This may occur if the PM hosting the replica has lower capabilities (e.g., slower core speed) than the PM hosting the primary VM. For example, the degrading performance of $S2$ is 1 because the core speed of P_0 which is the V_4 host is higher than P_1 which is the V_1 host (Figure 1). While the degrading performance of $S3$ is 0; this is because all of the hosts of the primary VMs have lower core speeds than the hosts of their replicas.

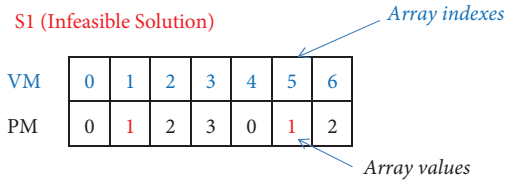
4.3. Dataset. To validate the proposed model, a dataset has been compiled containing the capabilities of PMs, VMs, the VMs map, and the distance between PMs. PMs' specifications dataset is generated as follows:

First, realistic PMs' specifications were collected (e.g., RAM, CPU, and clock rate). For example, PM_0 has 1 TB RAM, 28 cores CPU, and 2.5 GHz clock rate (Figure 1). Each collection of RAM, CPU, and clock rate specifications is known as a PM's category or type. For instance, 1 TB RAM, 28 cores CPU, and 2.5 GHz clock rate represents PM type 1. Second, the percentage k of each type in the dataset was detected. If the dataset has four types, then one possible combination of these four types is as follows, $k_1 = 30\%$, $k_2 = 20\%$, $k_3 = 40\%$, and $k_4 = 10\%$ of N. Third, each type k was repeated a number of times in the PM dataset. Fourth, the PMs were randomly rearranged and the consecutive PMs are not necessarily of the same PM type. These steps are used to generate VMs' specification datasets as well.

In the dataset, the distances between PMs are generated randomly as well. For example, in Figure 1, the distance matrix consists of N rows and N columns. The distance between PM and itself is zero; in other words, the matrix diagonal has only zeros. The distance between P_0 and P_1 is the same as the distance between P_1 and P_0. The mapping matrix represents the replica of a specific VM. For example, in Figure 1, V_4 and V_5 are replicas of V_1. The number of replicas for each VM (e.g., 0, 1, or 2) is selected randomly.

4.4. The Proposed Algorithm Complexity Analysis. Each iteration in the proposed algorithm has two main steps: (i) repair the generated solution and (ii) evaluate objective functions (e.g., $f(e)$, $f(d)$, and $f(c)$). To repair infeasible solutions, Algorithm 1 consists of one loop and one nested loop. The first loop finds the VM that has been incorrectly allocated to the PM, and Algorithm 1 time complexity is $O(T)$, where T is the total number of VMs and replica $T = M + R$. The second nested loop allocates the unassigned VM into a valid PM and its time complexity at the worst case is $O(T \times N)$, where N is the number of PMs.

Algorithm 2 evaluates the first objective by getting the number of unique PMs in solution S, its complexity is $O(1)$. The second objective is evaluated by Algorithm 3, which calculates the total distance between the primary VM host



Distance	PM_0	PM_1	PM_2	PM_3
PM_0	0	10	15	5
PM_1	10	0	20	8
PM_2	15	20	0	7
PM_3	5	8	7	0

Req	RAM	Core	CR
PM_0	1TB	28	2.50
PM_1	1TB	24	3.1
PM_2	2TB	48	4.5
PM_3	2TB	32	2.0

Map	Replica 1 with	Replica 2 with
VM_0	--	--
VM_1	4	5
VM_2	--	--
VM_3	6	--

Req	Core	RAM
VM_0	2	8G
VM_1	4	16G
VM_2	4	8G
VM_3	8	64G

Figure 1: Example of three solutions (infeasible, feasible, and optimal solutions) of the proposed model.

and its replica host r. It consists of one nested loop. The outer loop traces the primary VM, and its complexity is $O(M)$. The inner loop traces its replica host; thus, its time complexity is $O(r)$. So, the complexity of Algorithm 3 is $O(M \times r)$. Similarly, the third objective is evaluated by Algorithm 4 that calculates the total performance degrading between the primary VM host and its replica host r; thus, its time complexity is $O(M \times r)$.

5. Experimental Results

5.1. Setup. Six experiments were applied to measure the performance of the model. The proposed optimization model NSGA-III was implemented and other metaheuristic models such as MOEA/D and particle swarm optimization (PSO) using the pymoo [25] library in Python. The NSGA-III default values of the pymoo library were used for this reasoning. One of these values is the scale factor or mutation parameter; this parameter range is (0, 2]. The higher values of this parameter increase exploration and the lower values increase the exploitation. The default value of the mutation parameter balances exploration and exploitation. The experiments were conducted on a PC with Intel (*R*) Core(TM) i7-4770S CPU @ 3.10 GHz Ram 8.00 GB. The adaptive NSGA-III algorithm was compared to the PSO [26], MOEA/D [27], first-fit, next-fit, and best-fit [28].

The best algorithm of those aforementioned algorithms is selected based on the following equation:

$$\text{Fitnessvalues}(F) = w \times (f(e)/100) + w \times (f(d)/1,000) + w \times (f(c)/10), \tag{8}$$

where weight $w = 0.33$. Each of the three objectives has the same weight (i.e., one-third). As the first objective, $f(e)$ is four orders of magnitude, the second objective is five orders of magnitude, and the third objective is three orders of magnitude; we unified the value of each objective to only two orders of magnitude in equation (8). The used weights for the three objectives are the same to indicate equal importance of the three objectives.

5.2. Heterogeneous Environment. In this experiment, the efficiency of the model in heterogeneous VM and PM environments was examined. Five different types of VMs and PMs are used to evaluate the proposed model and discover the best solution. Table 3 lists the heterogeneous VMs' and PMs' specifications, where M equals 7,500, N equals 4,000, and R represents 25% of M. Table 4 indicates the best outcomes for the three objective functions (i.e., number of

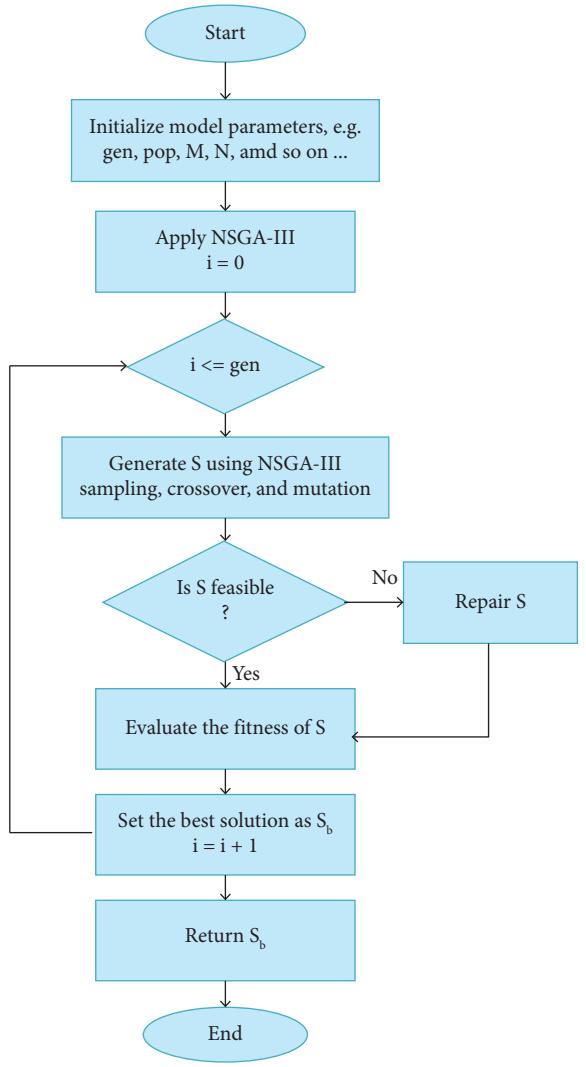


Figure 2: Flowchart of the proposed model.

```
Input: S
Output: S
count = 0;
for VMindex: = 0 to length (S)−1 do
  if checkConstrains (VMindex, S [VMindex]) == 0 then
    UpdatedList [count] = VMindex;
    /*Del the host of VM from the Solution*/
    S[VMindex] = −1
    count ++;
  end
end
for j: = 0 to count do
  VMindex = UpdatedList [j];
  if HasReplica (VMindex) == 1 then
    /*Select the valid PM with the minimum distance with replicas*/
    PMIndex = SelectPMMinDis (VMindex);
    S[VMindex] = PMIndex;
  end
  else
```

Algorithm 1: Continued.

```
    /*Sort descending the PM based on core capacity*/
    PMList = SortDesPM (S)
    for k: = 0 to length (PMList) do
      PMIndex = PMList [k]/*check its valid to add VM to PM*/if checkvalidity (VMindex, PMIndex) then
        S[VMindex] = PMIndex;
        break
      end
    end
  end
end
```

Algorithm 1: Repair solution.

```
Input: S
Output: n
n = length (unique (S))
return (n)
```

Algorithm 2: $f(e)$: Number of used PMs.

```
Input: S, V, Distance
Output: tolDistance
tolDistance = 0
for i: = 0 to V − 1 do
PrimaryPM = S[i]
Replica = getReplicas (i)
for j: = 0 to length (Replica)−1 do
ReplicaIndex = Replica[j]
ReplicaPM = S[ReplicaIndex]
tolDistance = tolDistance + Distance [PrimaryPM, ReplicaPM]
end
end
return (tolDistance)
```

Algorithm 3: $f(d)$: total distance between VMs and its replica.

```
Input: S, V
Output: Count
Count = 0
for i: = 0 to V − 1 do
PrimaryPM = S[i]
Replica = getReplicas (i)
for j: = 0 to length (Replica)−1 do
ReplicaIndex = Replica [j]
ReplicaPM = S[ReplicaIndex]
if ClockRate (PrimaryPM) > ClockRate (ReplicaPM) then
Count ++;
end
end
end
return (Count)
```

Algorithm 4: $f(c)$: the performance degrading due to the PM hosting the replica is with lower capabilities.

Table 3: Specification of heterogeneous environment.

Type		Memory	Number of cores	Clock rate (GHz)	k (%)
PM	1	768 GB	28	2.50	30
	2	2 TB	32	2.0	15
	3	1 TB	48	4.5	10
	4	1 TB	24	3.1	25
	5	2 TB	96	3.6	20
VM	1	8 GB	2	2.5	25
	2	16 GB	4	3.5	25
	3	72 GB	30	3.6	20
	4	64 GB	8	2.5	25
	5	976 GB	46	2.3	5

Table 4: Experimental results of heterogeneous environment

Model		Number of PMs	Distance	Performance degrading	F
	First-fit	2,066	22,559.00	556.00	32.61
	Next-fit	3,546	22,578.00	597.00	38.85
	Best-fit	3,403	18,750.00	604.00	37.35
gen 4,000 pop 25	NSGA-III	2,629.67 ± 51.83	19,244.33 ± 21.57	101.33 ± 26.76	18.37
	PSO	2,839.33 ± 3.21	20,319.00 ± 54.53	550.00 ± 19.67	34.23
	MOEA/D	2,610.00 ± 27.22	18,934.67 ± 14.98	183.00 ± 9.64	20.90
gen 5,000 pop 50	NSGA-III	2,524.50 ± 6.36	19,022 ± 48.08	55 ± 11.31	16.42
	PSO	2,824.00 ± 4.24	20,156.00 ± 70.71	560.00 ± 1.41	34.45
	MOEA/D	2,566.50 ± 9.19	18,906.50 ± 12.02	181.00 ± 12.73	20.68
gen 10,000 pop 50	NSGA-III	**2,438.50 ± 58.69**	**18,853.00 ± 14.14**	**31.00 ± 11.31**	**15.29**
	MOEA/D	2,599.50 ± 3.54	18,895.00 ± 33.94	187.00 ± 25.46	20.98

where bold numbers indicate the best result.

PMs of equation (5), distance between PMs hosting the replicas of equation (6), and performance degrading of equation (7)) in bold. As shown in Table 4, the NSGA-III algorithm yields the highest performance with the lowest fitness values. The NSGA-III algorithm's score is lower than those of other algorithms by 43.66%, 52.71%, 50.81%, 46.32%, and 12.10% for first-fit, next-fit, best-fit, PSO, and MOEA/D, respectively. The NSGA-III algorithm achieves the lowest possible level for the number of PMs and performance degrading. Best-fit achieves the minimum distance since it selects the hosted PM based on distance. First-fit and next-fit perform the worst in comparison to the other algorithms. This can be linked to the fact that the first-fit heuristics searches the first n valid PMs to host VMs while next-fit searches valid PMs sequentially. Notably, NSGA-III and MOEA/D performed similarly in lowering the distance between replicas and the number of PMs, while the NSGA-III algorithm significantly outperformed MOEA/D in reducing the performance degrading PMs (i.e., clock rate). Specifically, Table 4 demonstrates that NSGA-III reduced performance degradation by almost 600% compared to the MOEA/D technique. As this experiment was repeated several times, the reported results are the mean and mean plus and minus the standard deviation of the NSGA-III improvement over the closest algorithm MOEA/D. To test the confidence interval (CI) of the reported mean of improvement of the NSGA-III over the MOEA/D, the confidence interval analysis was conducted on the obtained results. The CI analysis shows that the reported mean of improvement of the NSGA-III over the MOEA/D has 15.1% with ± 0.6% error margin and 95% confidence interval. The obtained solutions are following the constraints in equations (1) to (3).

5.3. Memory Intensive in Heterogeneous Environments. The efficiency of the proposed model was evaluated with more complicated scenarios using memory-intensive requirements. Table 5 lists the VMs' and PMs' specifications, where M equals 7,500, N equals 4,000, and R represents 25% of M. NSGA-III, MOEA/D, and PSO parameters are pop = 50 and gen = 10, 000. Table 6 shows the experimental results of the proposed model in the memory-intensive heterogeneous environment relative to other algorithms. The bold results represent the best outcomes for the three objective functions. The obtained solutions are following the constraints in equations (1) to (3). The NSGA-III algorithm successful achieved the lowest (i.e., the best) fitness values, as shown in Figure 3. An observation was made that there is a performance gap between the NSGA-III algorithm on one side and the other algorithms. For instance, the results of the NSGA-III's fitness values for 4,000 generations and population size of 25 are lower than those of the other algorithms by 58.29%, 62.65%, 62.55%, 55.70%, and 31.02% for first-fit, next-fit, best-fit, PSO, and MOEA/D, respectively. As the search space increases, the performance gap between the NSGA-III algorithm on one side and the MOEA/D and PSO algorithms on the other side continues to increase. The search space increases by increasing the number of generations and population size. Thus, the performance gaps

Table 5: Specifications of memory intensive in heterogeneous environments.

Type		Memory	Number of cores	Clock rate (GHz)	k (%)
PM	1	768 GB	28	2.50	30
	2	2 TB	32	2.0	15
	3	1 TB	48	4.5	10
	4	1 TB	24	3.1	25
	5	2 TB	96	3.6	20
VM	1	64 GB	2	2.5	25
	2	128 GB	4	3.5	25
	3	256 GB	30	3.6	20
	4	512 GB	8	2.5	25
	5	976 GB	46	2.3	5

Table 6: Experimental results of memory-intensive heterogeneous environments

	Number PM	Distance	Performance degrading	F
First-fit	2,460	22,526	561	34.06
Next-fit	3,626	22,521	565	38.04
Best-fit	3,441	18,750	618	37.94
NSGA-III	**2,310 ± 1.40**	**18,804 ± 1.40**	**11.50 ± 2.10**	**14.21**
PSO	2,759.50 ± 7.78	19,548.5 ± 0.71	500.50 ± 47.38	32.07
MOEA/D	2,557.50 ± 6.36	18,840.50 ± 2.12	180 ± 8.49	20.60

where bold numbers indicate the best result.

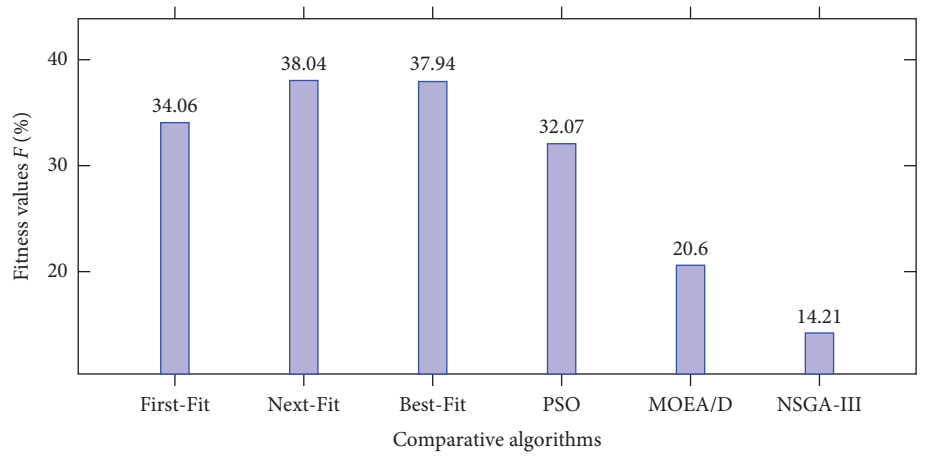


Figure 3: The fitness values of the proposed adaptive NSGA-III algorithm and comparative algorithms on memory intensive in a heterogeneous environment.

between the NSGA-III algorithm with 5,000 and 10,000 generations relative to other algorithms are greater than the performance gap of the NSGA-III algorithm with 4,000 generations.

Table 6 shows that the NSGA-III algorithm, with 10,000 generations, reduced performance degrading, and reduced the number of PMs by about 1,600% and 7%, respectively, in comparison with the MOEA/D algorithm. A conclusion can be drawn that other algorithms are influenced by the memory-intensive requirements, which resulted in more PMs being utilized to host these VMs. Increasing the number of VMs leads to increasing the distance objective as well.

5.4. Compute Intensive in Heterogeneous Environments. In this experiment, the efficiency of the proposed model was examined in a compute-intensive heterogeneous VM and PM setting utilizing five different types of VMs and PMs. The specifications for heterogeneous VMs and PMs are detailed in Table 7, where M equals 6,000, N equals 4,000, and R represents 25% of M. The parameters for NSGA-III and PSO are pop = 50 and gen = 10, 000. The results of all algorithms for the compute-intensive experiments are depicted in Table 8. The results demonstrate that the NSGA-III algorithm outperforms the other algorithms, as shown in Figure 4. When comparing the fitness values of the NSGA-III algorithm to that of the other algorithms, there is a significant

Table 7: Specification of compute intensive in the heterogeneous environment.

Type		Memory	Number of cores	Clock rate (GHz)	k (%)
PM	1	768 GB	28	2.50	10
	2	2 TB	32	2.0	15
	3	1 TB	48	4.5	15
	4	1 TB	72	3.1	30
	5	2 TB	96	3.6	30
VM	1	8 GB	16	2.5	40
	2	16 GB	30	3.5	40
	3	32 GB	46	3.6	10
	4	48 GB	62	2.5	5
	5	976 GB	70	2.3	5

Table 8: Experimental results of compute-intensive heterogeneous environments

	Number of PMs	Distance	Performance degrading	F
First-fit	2,978	17,836	470	31.22
Next-fit	3,787	18,050	459	33.60
Best-fit	3,505	15,000	463	31.80
NSGA-III	**2,552.50 ± 17.70**	**15,002**	**18 ± 4.20**	**13.97**
PSO	2,861 ± 12.70	15,257.50 ± 29	365 ± 19.80	26.52
MOEA/D	2,662 ± 12.73	15,027.50 ± 3.54	140 ± 2.83	18.36

where bold numbers indicate the best result.

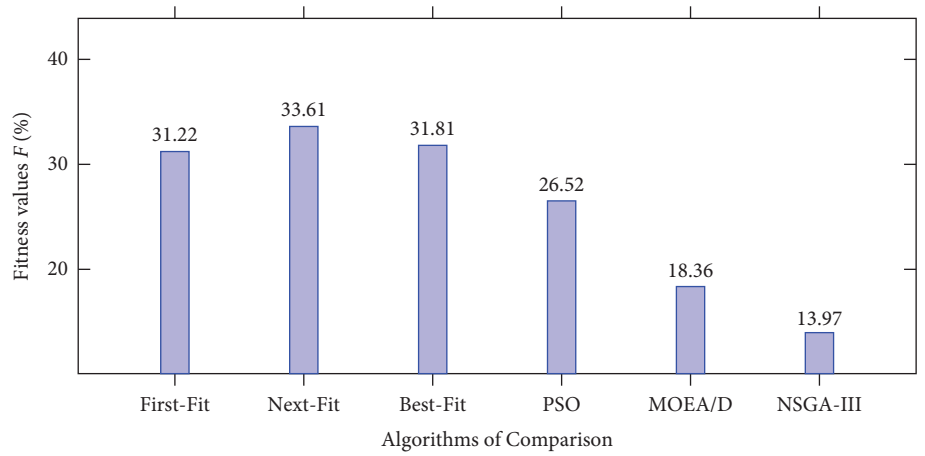


Figure 4: The fitness values of the proposed algorithm NSGA-III and comparative algorithms on compute intensive in a heterogeneous environment.

difference. It is lower than other algorithms by 55.26%, 58.43%, 56.07%, 47.33%, and 23.94% for first-fit, next-fit, best-fit, PSO, and MOEA/D, respectively. The NSGA-III algorithm generated the smallest number of PMs and performance degrading. This indicates that NSGA-III performs well in a compute-intensive context.

5.5. Proposed Method's Performance Analysis. In this experiment, we demonstrate how changing the population size affects the outcome. Furthermore, the utilized NSGA-III algorithm's behavior is investigated as the number of generation increases. In Figure 5, the number of PMs, distance, number of replicas causing performance degrading, and fitness values are plotted against the number of population pop for different generation gen values, gen equals 1,000, 2,000, and 3,000. As can be seen, the lowest fitness value is 17.66. This value is achieved by the highest pop and gen values, where pop = 50 and gen = 3,000. In general, the higher the pop and gen values, the lower the distance, number of PMs, and performance degrading. This is logical because when the numbers of pop and gen increase, the NSGA-III algorithm takes more time to search for the best solution with the highest performance (i.e., the larger the search space).

5.6. Heterogeneous Environment with Different SLA Scenarios. Allocating VM replicas to a PM with fewer hardware requirements relative to PM hosting the primary replica may lead to an increase in the SLA violation due to performance degrading. Therefore, the VM replica should be allocated to

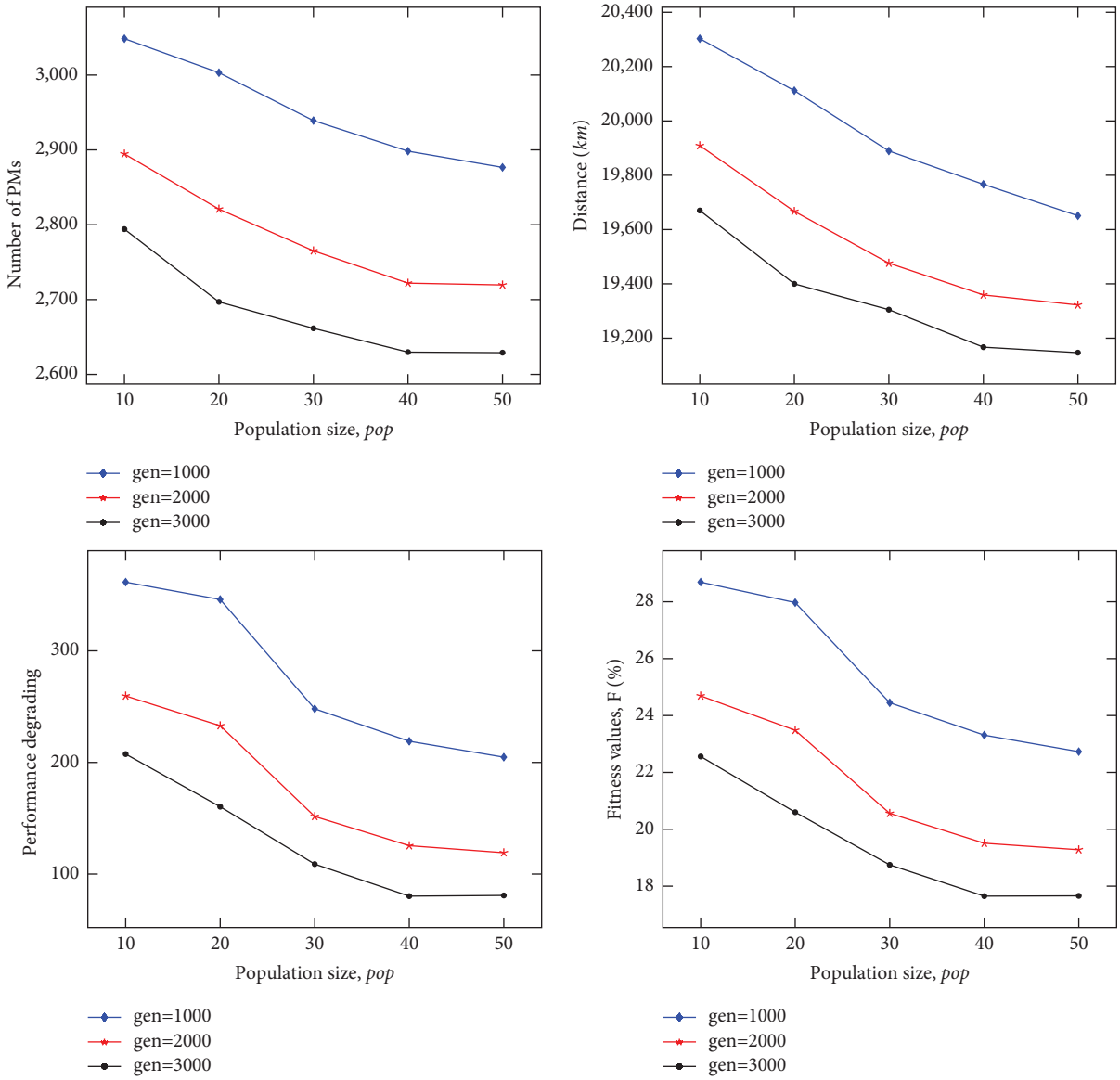


Figure 5: The performance of NSGA-III on different population sizes and a different number of generations.

a PM with at least the same hardware specification as the PM hosting the primary replica. In this experiment, two SLA scenarios were compared. Scenario 1 acts as Algorithm 4 where the VM replicas are allocated on PM with at least the same hardware specification as the PM hosting the primary replica while in scenario 2, the VM replicas allocated to PMs have the s same hardware specification as the PM hosting the primary replica. Five different types of VMs and PMs are used to evaluate the proposed model and to discover the best scenario. Table 3 lists the heterogeneous VMs' and PMs' specifications, where M equals 7,500, N equals 4,000, and R represents 25% of M. The NSGA-III algorithm's parameters are pop = 50 and gen equals (e.g., 2,000, 4,000, 6,000, 8,000, and 10,000). As shown in Figure 6, scenario 1 is superior to scenario 2 in terms of performance degrading and fitness values. This is logical, as increasing the search space leads to enhancing the NSGA-III algorithm's performance.

5.7. Statistical Analysis. In this experiment, analysis of variance (ANOVA) is applied in order to compare the means of six groups (e.g., first-fit, next-fit, best-fit, PSO, NSGA-III, and MOEA/D results). These groups were compared in terms of the three objectives (e.g., $f(e)$, $f(c)$, and $f(d)$) and fitness values. The results of the ANOVA test are shown in Table 9. The following are the H_0 and H_1 hypotheses for the ANOVA test: H_0: for each group, the means are the same. H_1: in the two groups, there are differences in the mean.

To examine the confidence level of the reported results, 95% of the confidence interval was utilized. Therefore, the hypothesis H_0 is accepted if the significance parameter (abbreviated sig.) or p value is greater than 0.05. Table 9 shows that the sig. value for objective function, $f(d)$, is more than 0.05. Therefore, for the distance objective, the hypothesis H_0 is accepted. This indicates that, with a 95% confidence interval, the average distance between VM and its

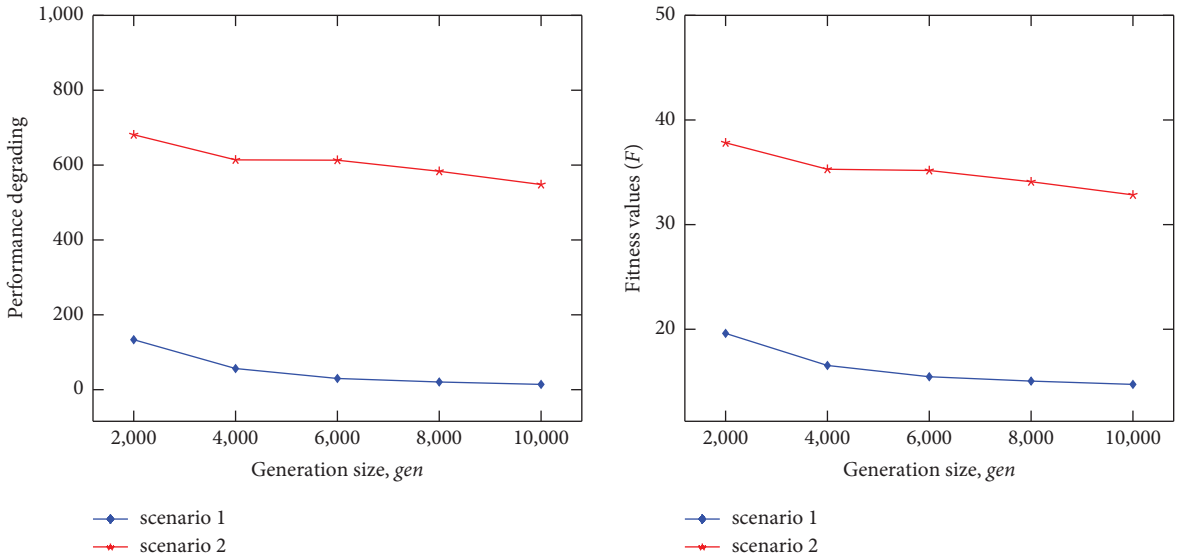


Figure 6: The performance of NSGA-III on different SLA scenarios with a different number of generations.

Table 9: The ANOVA test regarding utilization of first-fit, next-fit, best-fit, PSO, NSGA-III, and MOEA/D algorithms.

	Source of variation	SS	df	MS	F	Sig.	F crit
Number of PMs	Between groups	4015382.74	5	803076.55	19.50	$2.18915E-05$	3.11
	Within groups	494128.83	12	41177.40			
	Total	4509511.57	17				
Distance	Between groups	44173994.44	5	8834798.89	1.48	0.266183005	3.11
	Within groups	71496377.67	12	5958031.47			
	Total	115670372.11	17				
Performance degrading	Between groups	791787.28	5	158357.46	36.67	$7.26942E-07$	3.11
	Within groups	51827.67	12	4318.97			
	Total	843614.94	17				
Fitness values	Between groups	1240.29	5	248.06	36.70	$7.23549E-07$	3.11
	Within groups	81.12	12	6.76			
	Total	1321.40	17				

Table 10: The LSD test regarding utilization of first-fit, next-fit, best-fit, PSO, NSGA-III, and MOEA/D algorithms.

	Algorithm 1	Algorithm 2	Diff.	lwr.ci	upr.ci	Sig.
Number of PMs	NSGA-III	Best-fit	−1016.00	−1377.00	−655.00	0.000051
	NSGA-III	First-fit	−67.67	−428.66	293.33	0.69017
	NSGA-III	MOEA/D	−172.67	−533.66	188.33	0.31789
	NSGA-III	Next-fit	−1219.33	−1580.33	−858.34	0.0000087
	PSO	NSGA-III	381.17	20.17	742.16	0.04016
Performance degrading	NSGA-III	Best-fit	−541.50	−658.41	−424.59	$3.20E-07$
	NSGA-III	First-fit	−508.83	−625.75	−391.92	$6.30E-07$
	NSGA-III	MOEA/D	−148.83	−265.75	−31.92	0.0168
	NSGA-III	Next-fit	−520.17	−637.08	−403.25	$5.00E-07$
	PSO	NSGA-III	455.00	338.09	571.91	$2.10E-06$
Fitness values	NSGA-III	Best-fit	−21.20	−25.83	−16.58	$3.60E-07$
	NSGA-III	First-fit	−18.14	−22.77	−13.52	$1.90E-06$
	NSGA-III	MOEA/D	−5.49	−10.12	−0.87	0.02377
	NSGA-III	Next-fit	−22.34	−26.97	−17.72	$2.10E-07$
	PSO	NSGA-III	16.53	11.90	21.15	$5.00E-06$

replica for each of the six algorithms (i.e., first-fit, next-fit, best-fit, PSO, NSGA-III, and MOEA/D results) are almost similar.

In addition, as listed in Table 9, the sig. value for the objective functions $f(e)$, $f(c)$, and fitness values f is less than 0.05. Therefore, hypothesis H_0 is rejected for the number of PMs, performance degrading, and fitness values. This indicates that there is a difference in the mean of the two algorithms at least for $f(e)$, $f(c)$, and f. However, the ANOVA test gives no indication of the mean value for any method. Thus, a post hoc Fisher's least significant difference (LSD) analysis was applied. LSD can be used to determine whether group means are different from one another. Table 10 lists the LSD multiple comparisons for $f(e)$, $f(e)$, and f. Notably, the focus was mainly placed on the NSGA-III algorithm, as it outperformed other algorithms in the previous experiment. The results clearly show that for $f(c)$, the NSGA-III algorithm's mean is closer to the MOEA/D and first-fit algorithms than that of the other algorithms. This demonstrates how the NSGA-III algorithm is superior to the next-fit, best-fit, and PSO methods on $f(e)$. For $f(c)$ and f, the NSGA-III algorithm is superior to the first-fit, next-fit, best-fit, PSO, and MOEA/D methods.

6. Conclusions

In this work, it was proposed to address the problem of VMP in the context of VM replication. Proposing to utilize the NSGA-III algorithm to confront this problem, the VMP was addressed as a multiobjective optimization problem. Construction solutions were proposed to fit the NSGA-III method. In addition, a repair algorithm to restore the infeasible solutions generated by the utilized methods was suggested. Subsequently, a set of datasets with different requirements in terms of memory and computing was proposed. These datasets cover different scenarios such as heterogeneous PMs and VMs, compute-intensive environment, and memory-intensive environment. The proposed method is examined on these datasets, and their performance is compared against both heuristic and metaheuristic methods (e.g., MOEA/D and PSO). The experimental results show a superior performance of the adapted NSGA-III method where the performance gap was in the range of 23% to 62% relative to the other methods of comparison. The utilized NSGA-III massively outperformed the MOEA/D algorithm in reducing the performance degrading, and NSGA-III outperformed MOEA/D for the overall performance as well.

The future directions are two-fold. First, the proposed model will include additional objectives such as application awareness to consider placing the VM according to the match between its running application(s) and the PM's specifications. Second, the problem solution can be explored using a reinforcement learning model and then compared to the proposed GA-based method.

Data Availability

The datasets generated during the current study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. F. Mohamed, "Service replication taxonomy in distributed environments," *Service Oriented Computing and Applications*, vol. 10, no. 3, pp. 317–336, 2016.

[2] C. Gonzalez and B. Tang, "FT-VMP: fault-Tolerant virtual machine placement in cloud data centers," in *Proceedings of the 2020 29th International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–9, Honolulu, HI, USA, August 2020.

[3] J. Hu, J. Luo, and K. Li, "Opportunistic energy cooperation mechanism for large internet of things," *Mobile Networks and Applications*, vol. 23, no. 3, pp. 489–502, 2018.

[4] H. F. ElYamany, M. F. Mohamed, K. Grolinger, and M. A. Capretz, "A generalized service replication process in distributed environments," in *Proceedings of the 5th International Conference on Cloud Computing and Services Science*, Liverpool UK, January 2015.

[5] M. Elshrkawey, S. M. Elsherif, and M. Elsayed Wahed, "An enhancement approach for reducing the energy consumption in wireless sensor networks," *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 2, pp. 259–267, 2018.

[6] H. Goudarzi and M. Pedram, "Energy-efficient virtual machine replication and placement in a cloud computing system," in *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing*, pp. 750–757, Honolulu, HI, USA, June 2012.

[7] W. Hussain, O. Sohaib, M. Naderpour, and H. Gao, "Cloud marginal resource allocation: a decision support model," *Mobile Networks and Applications*, vol. 25, no. 4, pp. 1418–1433, 2020.

[8] A. García-Nájera and A. López-Jaimes, "An investigation into manyobjective optimization on combinatorial problems: analyzing the pickup and delivery problem," *Swarm and Evolutionary Computation*, vol. 38, pp. 218–230, Feb. 2018.

[9] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[10] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, Part I: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[11] Q. Liu, X. Liu, J. Wu, and Y. Li, "An improved NSGA-III algorithm using genetic K-Means clustering algorithm," *IEEE Access*, vol. 7, pp. 185239–185249, 2019.

[12] A. Al-Moalmi, J. Luo, A. Salah, and K. Li, "Optimal virtual machine placement based on grey wolf optimization," *Electronics*, vol. 8, no. 3, p. 283, 2019.

[13] F. Alharbi, Y. C. Tian, M. Tang, W. Z. Zhang, C. Peng, and M. Fei, "An ant colony system for energy-efficient dynamic virtual machine placement in data centers," *Expert Systems with Applications*, vol. 120, pp. 228–238, 2019.

[14] E. Parvizi and M. H. Rezvani, "Utilization-aware energy-efficient virtual machine placement in cloud networks using NSGA-III meta-heuristic approach," *Cluster Computing*, vol. 23, no. 4, pp. 2945–2967, 2020.

[15] M. Tarahomi, M. Izadi, and M. Ghobaei-Arani, "An efficient power-aware VM allocation mechanism in cloud data centers: a micro genetic-based approach," *Cluster Computing*, vol. 24, no. 2, pp. 919–934, 2021.

[16] T. Abbasi-khazaei and M. H. Rezvani, "Energy-aware and carbon-efficient VM placement optimization in cloud data-centers using evolutionary computing methods," *Soft Computing*, vol. 26, no. 18, pp. 9287–9322, 2022.

[17] S. S. Alresheedi, S. Lu, M. Abd Elaziz, and A. A. Ewees, "Improved multiobjective salp swarm optimization for virtual machine placement in cloud computing," *Human-centric Computing and Information Sciences*, vol. 9, no. 1, pp. 15–24, 2019.

[18] W. Zhang, X. Chen, and J. Jiang, "A multi-objective optimization method of initial virtual machine fault-tolerant placement for star topological data centers of cloud systems," *Tsinghua Science and Technology*, vol. 26, no. 1, pp. 95–111, 2021.

[19] Y. Sharma, W. Si, D. Sun, and B. Javadi, "Failure-aware energy-efficient VM consolidation in cloud computing systems," *Future Generation Computer Systems*, vol. 94, pp. 620–633, 2019.

[20] P. Khani, B. Tang, J. Han, and M. Beheshti, "Power-efficient virtual machine replication in data centers," in *Proceedings of the 2016 IEEE International Conference on Communications (ICC)*, pp. 1–7, Kuala Lumpur, Malaysia, May 2016.

[21] Y. Yao, J. Cao, and M. Li, "A network-aware virtual machine allocation in cloud datacenter," in *Proceedings of the IFIP International Conference on Network and Parallel Computing*, pp. 71–82, Shanghai China, September 2013.

[22] A. Al-Moalmi, J. Luo, A. Salah, K. Li, and L. Yin, "A whale optimization system for energy-efficient container placement in data centers," *Expert Systems with Applications*, vol. 164, Article ID 113719, 2021.

[23] J. Blank, K. Deb, and Proteek Chandan Roy, "Investigating the normalization procedure of nsga-iii," in *Evolutionary Multi-Criterion Optimization, 229–240*, K. Deb, E. Goodman, A. Carlos et al., Eds., Springer International Publishing, Midtown Manhattan, NY, USA, 2019.

[24] J. Blank and K. Deb, "Pymoo: multi-objective optimization in Python," *IEEE Access*, vol. 8, pp. 89497–89509, 2020.

[25] Pymoo, "PYMOO:Multi-objective Optimization in Python," 2020, https://pymoo.org/.

[26] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the ICNN'95-International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, WA, Australia, November 1995.

[27] Q. Zhang and H. Li, "MOEA/D: a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[28] G. Gigerenzer and W. Gaissmaier, "Heuristic decision making," *Annual Review of Psychology*, vol. 62, no. 1, pp. 451–482, 2011.