Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.058266

ARTICLE

AI-Driven Resource and Communication-Aware Virtual Machine Placement Using Multi-Objective Swarm Optimization for Enhanced Efficiency in Cloud-Based Smart Manufacturing

Praveena Nuthakki[1], Pavan Kumar T.[1], Musaed Alhussein[2], Muhammad Shahid Anwar[3,*], Khursheed Aurangzeb[2] and Leenendra Chowdary Gunnam[4]

[1]Department of Computer Science & Engineering, Koneru Lakshmaiah Education Foundation, Guntur, 522302, India

[2]Department of Computer Engineering, College of Computer and Information Sciences, King Saud University, Riyadh, 11543, Saudi Arabia

[3]Department of AI and Software, Gachon University, Seongnam-Si, 13120, Republic of Korea

[4]Department of Electronics and Communication Engineering, SRM University, Amaravati, 522502, India

*Corresponding Author: Muhammad Shahid Anwar. Email: shahidanwar786@gachon.ac.kr

Received: 08 September 2024 Accepted: 22 November 2024 Published: 19 December 2024

ABSTRACT

Cloud computing has emerged as a vital platform for processing resource-intensive workloads in smart manufacturing environments, enabling scalable and flexible access to remote data centers over the internet. In these environments, Virtual Machines (VMs) are employed to manage workloads, with their optimal placement on Physical Machines (PMs) being crucial for maximizing resource utilization. However, achieving high resource utilization in cloud data centers remains a challenge due to multiple conflicting objectives, particularly in scenarios involving inter-VM communication dependencies, which are common in smart manufacturing applications. This manuscript presents an AI-driven approach utilizing a modified Multi-Objective Particle Swarm Optimization (MOPSO) algorithm, enhanced with improved mutation and crossover operators, to efficiently place VMs. This approach aims to minimize the impact on networking devices during inter-VM communication while enhancing resource utilization. The proposed algorithm is benchmarked against other multi-objective algorithms, such as Multi-Objective Evolutionary Algorithm with Decomposition (MOEA/D), demonstrating its superiority in optimizing resource allocation in cloud-based environments for smart manufacturing.

KEYWORDS

Resource utilization; smart manufacturing; efficiency; inter VM communication; virtual machine placement; cloud computing; multi-objective optimization

Glossary/Nomenclature/Abbreviations

VM	Virtual Machine
PM	Physical Machine
MOPSO	Multi-Objective Particle Swarm Optimization
VMP	Virtual Machine Placement

1 Introduction

Cloud computing is the most widely used computing platform of the decade offering prefabricated service to organizations. Many organizations are adopting cloud models to improve the availability of their business to people around the world. The organization does not need to establish a data center in every region. Because of the cloud properties like on-demand (use only when you need), elasticity (can increase or decrease the resource with easy steps), and pay-as-you-go model. Many organizations are taking an active part in shifting their workload to cloud-based data centers. In recent years many services have been given to consumers as off-the-rack offerings. For example, Amazon Web Service (AWS) provides a service recognition service that has an inbuilt algorithm to perform facial recognition with greater accuracy. The user needs to use their Application Programming Interface (API) and send a request to the service without worrying about the type of algorithm used at the backend. The service will help in identifying similar faces, annotation, or identifying text in images with a JavaScript Object Notation (JSON) reply. The cloud has its footprint on delivering effective computational power to numerous business use cases.

Cloud computing services models are majorly classified into IaaS (Infrastructure-as-a-Service), PaaS (Platform-as-a-Service), and SaaS (Software-as-a-Service). In this manuscript, we are concerned about the Infrastructure as a Service (IaaS) cloud which offers the highest amount of customization compared to PaaS and SaaS [1]. In the IaaS cloud, the user has the freedom to bundle their virtual server, by installing their operating system, configuring the amount of Random Access Memory (RAM) and Central Processing Unit (CPU) needed for running the application, and also the application runtime preference. The terms virtual server or virtual machine are interchangeably used and referring the same entity in IaaS cloud. Virtual machines are space-sharing entities destined to execute in the physical machine or servers. The servers' resources like CPU and RAM are utilized by the virtual machine until its lifecycle. When the virtual machines are terminated the server resources are utilized by another virtual machine. The power consumed by the datacenters is directly proportional to the number of virtual machines hosted in the datacenter and the communication between them VMs. It is estimated that a typical datacenter consumes electricity equivalent to 25,000 households. A report published in [2] states that a data center consumes 73 billion kWh, which is estimated to be 2% of the total energy consumption of the United States. This increase in power utilization is due to inefficiently managed datacenter resources. The increase in electricity utilization contributed highly to the emission of greenhouse gases leading to global warming. In [3], it is estimated that around 45% of the electricity is consumed by the server, 30% of the electricity datacenter is consumed by the networking devices, and the remaining 25% of the electricity is consumed by Heating, ventilation, and air conditioning (HVAC) system. Considering the above situation, our research is focused on reducing server resource wastage and inter-VM communication. Reducing the two also reduces the energy consumed by the HVAC systems.

Cloud providers need efficient algorithms to place the virtual machine as close as possible to a physical machine. This is equivalent to the bin packing problem. The bin is comparable to a Physical machine and VMs are equivalent to the number of items. The stated problem is a non-deterministic polynomial-time hard (NP-Hard) problem and it is proved in the literature. Apart from this VM placement, considering the inter-VM communication increases the complexity of the problem. Even though we placed the VMs as compact as possible the inter-VM communication increases (usually measured in terabytes) it is considered an inferior solution. This problem is equivalent to a muti-dimensional bin packing algorithm. Deterministic algorithms are inefficient in solving NP-Hard problems because deterministic algorithms intend to check every solution possibility (Brute Force) in the large solution space. To solve an NP-Hard problem, we employ bio-inspired algorithms that use guided random components to effectively exploit the large search space. These algorithms are non-deterministic and use heuristics to improve the existing solution even further during each iteration.

In this manuscript, we propose a modified Multi-Objective Genetic Algorithm (MOGA) to place the virtual machine improving data center resource utilization and reducing the inter-VM communication that in turn reduces the operating burden on networking equipment. The highlights of the research article are mentioned below:

1. A modified Multi-Objective Genetic Algorithm (MOGA) for Virtual Machine for Virtual Machine Placement is proposed. Optimal resource utilization and inter-VM communication are addressed using the proposed algorithm.
2. A novel mutation and crossover operator is proposed to improve the efficiency of the algorithm.
3. A fat tree topology-based communication model is adopted for the calculation of total data transfer between the VMs in the data center.
4. Resource utilization and inter-VM communication are modeled as minimization problems. The performance metrics resource wastage and inter-VM communication is compared with MOPSO and Space More Efficient Algorithms (SPEA).

The rest of the paper is organized as follows: Section 2 (Literature Survey) outlines the state of art technologies used for building an energy-efficient data center. Section 3 (Problem Formulation) presents the mathematical modeling of server resource utilization and Inter VM communication as a multi-objective minimization problem. Section 4 (Methodology) presents the modified Particle Swarm Optimization (PSO) algorithm to optimize the above-stated objectives. Section 5 presents Experimental Setup and Results with comparison and performance graphs.

2 Related Work

In this section, we present the state of algorithms/methodologies discussed in recent years. In recent years cloud providers and scientists have been very much focused on reducing the carbon footprint to build green data centers. Datacenter energy consumption is an active research field for many cloud providers and scientists. All the stakeholders are working to achieve better utilization of the data center by refining the existing solutions and practices. Building an energy-efficient data center not only depends on proposing effective algorithms but also on building energy-efficient hardware [4] (DVFS) and using renewable/green energy sources (solar, wind, hydro energy). Without loss of generality, our research work focuses on building an algorithm to effectively utilize data center resources.

The deterministic algorithm presented in [5] is a greater fit for the problem with a smaller search space. The advantage of using such an algorithm provides us with an exact solution. If the search space increases this algorithm takes exponential time to find the solution. In real-time, working with smaller search spaces is very occasional. Also, these deterministic algorithms fail to work in multi-objective space where two or more objective functions need to be optimized simultaneously. The algorithm employed for bin packing greedy algorithms such as Least Fill First Bin Packing [6], Most Fill First Bin Packing [7] and Next Fit Bin Packing [8] can also be used for VM placement problems. Greedy algorithms are also deterministic algorithms and better than exact algorithms but while exploring the search space the solution falls into local minima. Greedy algorithms cannot be parallelized to execute in multiple machines for faster solution finding. The next class of algorithms is a meta-heuristic algorithm. The algorithms are designed to imitate the behaviors of animals birds or insects.

Genetic algorithms are made to imitate the evolutionary process of human beings. Genes of two parents are exposed to the process of crossover and mutation to produce a new offspring. The new offspring is evaluated using the objective function. If the offspring is a better-performing individual, then the solution is considered else the crossover of the mutation is performed again to produce a better-performing individual. Genetic algorithms are mostly widely applied for permutation-based NP-Hard problems including traveling salesperson problems, bin packing problems, flow shop scheduling problems, and even VM placement problems. In [9], the authors proposed a multi-objective fuzzy-induced genetic algorithm to optimize resource utilization and power consumption in cloud data centers. The proposed algorithm was compared with various bin-packing algorithms to show its superiority. In [10], the authors proposed a genetic algorithm and it is implemented in the real-time cloud data center at the Office of the Merchant Marine and Ports of Tunisia (OMMP). The real-time implementation shows a reduction in resource wastage and leads to a profitable business. The authors used uniform crossover and uniform mutation to generate a new offspring. In [11], the authors proposed an improved genetic algorithm to optimize the availability and energy consumption of a data center. A problem-specific selection operator, crossover, and mutation operator are proposed in the manuscript. In [12], the authors proposed a hybrid algorithm where a part of the optimization is carried out by a genetic algorithm and the remaining is carried out by a best-fit bin packing-based algorithm to reduce power utilization and resource wastage. The proposed algorithm is also used to compare well-known instances of the Travelling Salesman Problem (TSP) and Flow shop scheduling problems. Variation of genetic algorithm for solving Virtual Machine Placement (VMP) in a multi-objective perspective is presented in [13]. In [14], the authors proposed a multi-objective variation of the genetic algorithm with decomposition to optimize three objectives simultaneously. Permutation-based partially mapped crossover and multi-point mutation are used because of the larger gene structure (100 and 200 VMs). Apart from the genetic algorithm, other algorithms such as ant colony optimization, firefly algorithm [15], bat algorithm, and particle swarm algorithm [16] are majorly used in VMP. The above state algorithm works well for discrete optimization problems. Unlike genetic algorithms (natively permutation-based), the challenge in adopting these algorithms lies in implementing effective discretization methodologies. The author proposed an order-filling method to discretize the continuous values used along with the bat algorithm. The bat algorithm was initially proposed as a single objective algorithm. The author used the working methodology and proposed a multi-objective version of the bat algorithm.

A multi-objective virtual machine (VM) placement algorithm based on evolutionary methods with decomposition has been proposed [17]. Their approach aims to address multiple conflicting objectives, including resource utilization and communication overhead, through efficient VM placement strategies in cloud data centers. In another study, authors in [18] presented a VM placement algorithm based on a Multi-Objective Evolutionary Algorithm with Decomposition (MOEA/D), aimed at optimizing placement in distributed cloud environments. Their work demonstrated significant improvements in balancing resource allocation, reducing energy consumption, and enhancing cloud infrastructure efficiency. The authors also discuss interoperability issues in cloud computing, which is a key concern in multi-cloud and hybrid-cloud environments [18]. Their study laid the groundwork for addressing system compatibility challenges and improving resource integration across different cloud platforms. A foundational algorithm in this domain was proposed by [10], inspiring many modern swarm-based methods for VM placement. PSO has been widely adopted due to its simplicity and effectiveness in solving multi-dimensional optimization problems.

Finally, a power-aware, performance-guaranteed VM placement algorithm is presented in [11], offering a hybrid solution that combines energy efficiency with performance constraints. Their work provided a comprehensive framework for optimizing energy usage in cloud environments without compromising performance, making it relevant for power-sensitive applications.

3 Problem Formulation

As discussed earlier, VM placement is equivalent to the bin packing problem. The set of virtual machine requests from the cloud consumer must be spawned in the available physical machine without wasting server resources. The VM demands are specified in terms of CPU and RAM. In our research work, we do not consider the storage requirement because it is provided by the centralized storage arrays connected using Storage Area Network (SAN) fabric. VM can execute in any of the Polynomial Machines (PM) in the data center, and the storage can be routed to the PM using the SAN network. Since we do not know the type of physical machine each cloud provider is installed in the data center, to generically represent the utilization in our research article we use a percentage of utilization. For example, consider a physical machine has 10 CPU cores and 20 GB of RAM which is equivalent to 100% of the resources being free in PM, the VM request needs 2 core CPU (20%) and 4 GB RAM (20%). The VM is allocated to the PM as a result 20% CPU is utilized and 20% of the RAM is utilized. The remaining 80% of CPU and RAM can be allocated to some other VMs or it is considered as resource wastage W_i.

$$W_i = \sum_{i=1}^{M} \left[y_i \times \frac{|(\theta_{Pi} - \sum_{j=1}^{N}(x_{i,j}.R_{p,j})) - (\theta_{Mi} - \sum_{j=1}^{N}(x_{i,j}.R_{M,j}))| + \varepsilon}{\sum_{j=1}^{N}(x_{i,j}.R_{p,j}) + \sum_{j=1}^{N}(x_{i,j}.R_{M,j})} \right]$$

$$x_{i,j} = \begin{cases} 1 & \textit{if } VM_j \textit{ is allocated to } PM_i \\ 0 & \textit{otherwise} \end{cases}$$

$$y_i = \begin{cases} 1 & \textit{if } PM_i \textit{ is used} \\ 0 & \textit{otherwise} \end{cases} \tag{1}$$

The total available resource of physical machine (PM_i) is denoted with the pair (θ_{Pi}, θ_{Mi}), where θ_{Pi} denotes the total available CPU and θ_{Mi} represents total available RAM. There can be any number of PM in a datacenter, out of that which not hosted with VMs/workloads are hibernated. The term y_i is a binary variable where ($y_i = 0$) denotes the PM_i is switched off. Hence the entire equation become null and excluded from calculating wastage. The term $x_{i,j}$ denoted the virtual machine VM_j hosted in Physical machine PM_i. The term ($R_{p,j}, R_{M,j}$) denotes the VM request CPU and RAM. An example calculation for resource wastage is illustrated below Fig. 1.

In the above illustration, three VMs are considered. The first two virtual machines are placed in PM_1 and the third virtual machine is placed in PM_2. The PM_3 is switched off since it is not hosted with any VMs. The total datacenter wastage is the sum of wastage of all PMs. The entire calculation is model in the Eq. (1) is also called as objective function or fitness function.

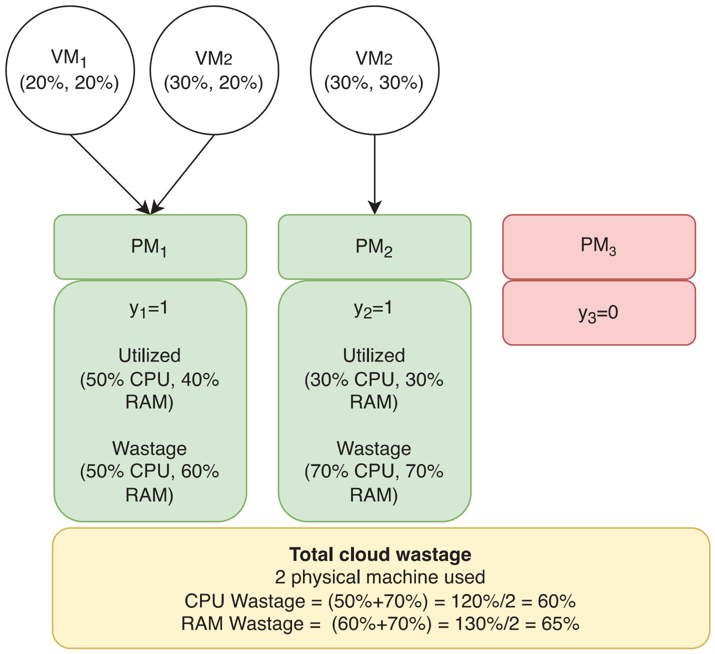
VM1 (20%, 20%)
VM2 (30%, 20%)
VM2 (30%, 30%)
PM1
PM2
PM3
y1=1
Utilized (50% CPU, 40% RAM)
Wastage (50% CPU, 60% RAM)
y2=1
Utilized (30% CPU, 30% RAM)
Wastage (70% CPU, 70% RAM)
y3=0
Total cloud wastage
2 physical machine used
CPU Wastage = (50%+70%) = 120%/2 = 60%
RAM Wastage = (60%+70%) = 130%/2 = 65%

Figure 1: Resource wastage calculation—illustration

The next objective formulation is concerned with the inter VM communication. A cloud consumer may deploy multiple virtual machine in a cloud datacenter. Each of these VMs need to communicate with each other to complete a given task. When the VMs are placed in different PMs the communication from one VM should pass thorough a set networking devices to reach the destination. This in turns compromises the Quality of Service (QoS) for the VMs executing in cloud. At the same time, the networking devices are heavily utilized, and the power consumption will also increases. It is estimated in [3] that 30% of electricity is consumed by the networking devices due to misplacement of VMs. It is desired to place inter communicating VMs in single physical machine to reduce the impact on networking devices. For our research work we considered the datacenter follows fat tree topology as shown in the Fig. 2. The physical machines are denotes with the numbering PM_1 to PM_{16}. An access swich can accommodate only two PMsand the number of ports in only two ($P = 2$). Each PM can accommodate number of VMs based on resource request. The access switch is further connected to two aggregation switches to avoid single point of failure. Fat tree topology is most widely used datacenter architecture because the datacenter can be easily scaled to support to multiple PM without affecting the exiting implementation. The term hops is defined as the number interconnecting networking device a data need to be routed to reach the destination. The volume of data need to be routed from a VM_i to VM_j is denoted as $d_{i,j}$.

The data transfer from source to destination has four special cases mentioned in the Table 1. The case C_1 is the most desired placement which does not use any networking devices (hops = 0). But, this kind of placement is rarely possible because the sum of requested resource of all VMs from a single cloud consumer mostly exceeds the resource capacity of PM. The Case C_2 is most obvious placement where the communication will pass through only one networking devices to the destination. But when the number virtual machines from single cloud consumer are more, we cannot contain them all in two physical machines. The number of hops needed to reach the destination is 2. The Case C_4 is considered as worst case scenario in cloud architecture. Consider the VM_i is placed in PM_3 and VM_j is placed in PM_{10}. The data need to be routed through access switch, aggregation switch, core switch and once again to aggregation switch, access switch and to the destination. The communication trace is depicted with dotted line in Fig. 2. This placement increases impact on networking devices and number of hops needed for this communication is 6.

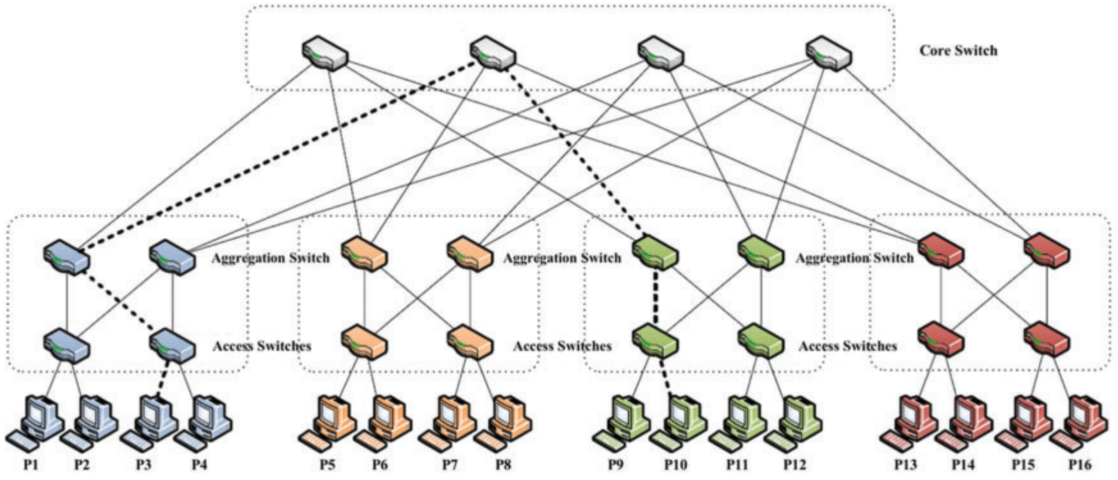

Figure 2: Flat tree topology

Table 1: Data transfer rate on links

Case	Positions	Example	Hops	Data rate on links
C_1	Under same physical machine	Two virtual machine placed in PM_1	0	$0 \times d_{i,j}$
C_2	Under same access switch	VM_1 is hosted in PM_1 then VM_2 is placed in PM_2	2	$2 \times d_{i,j}$
C_3	Under same aggregation switch	VM_1 is hosted in PM_1 then VM_2 is placed in PM_3	4	$4 \times d_{i,j}$
C_4	Under same core switch	VM_1 is hosted in PM_3 then VM_2 is placed in PM_{10}	6	$6 \times d_{i,j}$

For example, if VM_i is hosted in PM_3 then VM_j is placed in PM_{10} as depicted in the Fig. 2 with dotted lines then 6 hops are need to reach the destination. $P_{i,j}$ denotes the physical machine where the VM_i and VM_j is hosted. The data transfer volume is calculated as below:

$$d_{i,j} = hop\left(P_{i,j}\right) \times d(VM_i, VM_j) \quad (2)$$

If 500 MB of data need to transferred between 2 VMs say VM_i and VM_j that are hosted in in PM_3 and in PM_{10} then

$$d_{i,j} = hop\left(P_{3,10}\right) \times d\left(VM_i, VM_j\right) = 6 \times 300 = 1800MB$$

For the entire datacenter, the data transfer volume can be calculated using the below equation (minimization Objective function).

$$D = \sum_{i=1}^{n} \sum_{j=1}^{n} hop\left(P_{i,j}\right) \times d(VM_i, VM_j) \tag{3}$$

Both the objectives namely resource wastage and inter-VM communication are discussed in this section with a neat illustration. The problem we have here is conflicting objectives. The minimization of resource wastage objective may hurt inter-VM communication objectives. This kind of problem cannot be solved using traditional bin packing or deterministic algorithms. In the upcoming section, we discuss a multi-objective optimization and the algorithm.

4 Multi-Objective Optimization

The multi-objective optimization problem required specialized concepts to process and identify optimal values. In comparison with the single objective optimization problem, the fitness function produces only one value to be minimized, i.e., scalar quantity. But, in multi-objective optimization for a given problem instance, it is evaluated by more than one fitness function to produce a vector. For example, for two objective optimizations the output produced by the fitness function contains two values $[y_1, y_2]$.

Consider the input instance of the permutation problem is $x = [x_1, x_2, \ldots, x_n]$ where n denotes number of VMs. The multi objective optimization function is denoted with $f(x) = [f_1(x), f_2(x), \ldots, f_k(x)]$ where k denotes number of objectives. Each of the objective have equality and inequality constraints. These constraints are vital in reducing the search space. For example, a VM say VM_i should not be placed in multiple PM is an inequality constraint. The input instance is evaluated using the fitness function to produce the output vector y, $x \to f_k(x) \to y$ where $y = [y_1, y_2, \ldots, y_k]$. All the input instance exists in decision space or search space, once it is evaluated using the fitness function the outputs are plotted in objective space.

The comparison of solution is a major issue in multi-objective optimization problems. For example, in multi-objective optimization the outputs are vector quantity. Consider two outputs of bi-objective optimization problem in the decision space as $A = \{4, 5\}$, and $B = \{5, 4\}$. Out of the two solution A and B which one is better performing solution? In comparing solution A and B we cannot say A is better than B or B is better than A, because considering minimization problem the A performs better in 1st objective and B performs better in 2nd objective. The solution A and B are non-dominated solution. The collection of multiple non dominated solution plotted in objective space is called as Pareto Optimal Front. For continuous optimization problem the pareto front appears similar as shown in the Fig. 3, but for discrete optimization problems, the steady curve is unattainable.

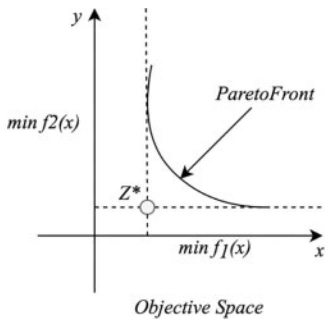

Figure 3: Illustration of pareto optimal front for bi-objective minimization problem

5 Proposed Methodology

In this section, we cover the modified multi-objective version of the Particle Swarm Optimization algorithm (PSO). Initially, the particle swarm optimization algorithm was proposed to solve single objective continuous optimization problems. Throughout the literature, there are many variations of PSO algorithms that exist for multi-objective continuous optimization problems. As a part of our research, we have a discrete version of the multi-objective PSO algorithm with improved exploration methodology.

The PSO algorithm is a swarm-based heuristic algorithm that can be observed in many of the creatures like flocks of birds searching for food. PSO methodology concentrates on specific to gain collective intelligence. Each permuted virtual machine sequence is called a particle. Each of the particles has two components called personal best and current fitness value. For the whole population of the particle, there exists only one global best. Initially, each particle is evaluated using the objective function to get its fitness values. During the first iteration, both the personal best and current fitness value of the particle remain the same. The global best is the best-performing particle in the entire population. From the next iteration onwards, each of the particles will shift its position to find the optimal solution guided by the global best and the personal best value. The importance given to the global best and personal best is controlled by the parameters c_1 and c_2.

$$v_i(t+1) = wv_i(t) + c_1r_1[p_i(t) - x_i(t)] + c_2r_2[g(t) - x_i(t)] \tag{4}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{5}$$

The current position of the particle is denoted with $x_i(t)$ and next position of the particle is denoted with $x_i(t+1)$ where x_i is the ith particle. Such that we initialize n number of particles called population. The personal best of the population is denoted with $p_i(t)$ and the global best solution is denoted with $g(t)$. w is the weight initialized to 0.9 and gradually decreases to 0 and the number of iteration increases.

$$\mathrm{v_i\,(t+1) = wv_i\,(t) + c_1r_1\,[p_i\,(t) - x_i\,(t)] + c_2r_2\,[g\,(t) - x_i\,(t)]} \tag{6}$$

Algorithm 1: Multi-Objective Particle Swarm Optimization (MOPSO) Algorithm

```
Inputs: Number of Virtual Machine ($N_{vm}$), Maximum Iteration ($MaxIt$), Number of Particles ($N$), Objective function $f()$, Algorithmic parameters ($w, c_1, c_2$), Traffic Pattern Matrix ($TPM$), Virtual Machine Dataset $VM_{DS}$, $g(t) = \infty$.
Initialize the number of particles using randperm( )function
//Evaluate each particle and update the current fitness value
for $i = 1$ to $x_n$
   Assign $p_i(t) = f(x_i)$
     if $x_i(t).cost < g(t).cost$ then
        Update the global best value with $x_i(t)$
end for
Determine the non-dominated solution based on the concepts discussed in Section 4.
for $i = 1$ to $MaxIt$
   for $j = 1$ to $x_n$
     Calculate the velocity $v_i(t+1)$ based on the Eq. (4).
     Update the current position of the particle $x_i(t+1)$ using the Eq. (5).
```

(Continued)

Algorithm 1 (continued)

```
        Calculate the objective value using the Eqs. (1) and (3).
        if $rand()$ <0.5 then
            randomly shuffle the positions of the virtual machine in the particle $x_i$.
            find its fitness value.
            Apply concept of domination along with newly generated solution
            if $new.x_i$ dominates all solution then
                assign $x_i = new.x_i$
        if $new.x_i$ dominates all solution then
            assign $x_i = new.x_i$
        if $p_i(t+1) < p_i(t)$ then
            assign $p_i(t+1) = p_i(t)$
        if $p_i(t+1) < g(t)$ then
            assign $g(t) = p_i(t+1)$
    end for
end for
Output: $g(t)$
```

The first term $wv_i(t)$ is equivalent to the current velocity of the particle multiplied by the weight w. Then the equation is divided into two major parts (as shown in Algorithm 1) called exploration and exploitation. Exploration means searching the space entire so that the particle will not strike into local minima. Exploitation means searching for the best solution within a confined search area. Initially, the exploration term will be given more weightage than exploitation as the number of iterations increases, exploration is minimized and we concentrate only on exploitation. The above algorithm is made to run for 100 iterations $MaxIt = 100$ with the population size $N = 100$. The corresponding minimization objectives are measured and the Pareto-optimal solution is collected in an external array. During each iteration, the best-performing particles are subjected to the concept of domination. The final results are tabulated in Section 6.

6 Experimental Setup and Results

The concepts described above were executed in MATLAB version R2022a in a computer system with the specification of an Intel core i3 processor with 16 GB of RAM running Windows operating system. The dataset needed to carry out the experimentation was statistically generated using the generating VM algorithm mentioned. Using this algorithm 200 virtual machines are generated by changing the reference probability values to (0 and 1) and the reference CPU and RAM is fixed $\overline{R_m} = 50$ and $\overline{R_c} = 50$. Using the parameter we generated two datasets with the resultant correlation coefficients of -0.7843 (Strong negative) and 0.7571 (Strong positive).

The inter-VM communication matrix is generated randomly, and the same dataset is used for running all the experiments and multiple independent runs. The VMs along the rows denote the source and the VM along the columns denotes the destination. A matrix is created of size $200 * 200$ and the values inside denote data volume. All the data volumes are represented in MBs and round off to the nearest 10's (11 MB is rounded to 10 MB).

The proposed algorithm MOPSO is compared with MOEA/D in Table 2 which shows the superiority of finding the minimal Pareto front solution. Also, the plots in Fig. 4 show that MOPSO explores the search space more effectively compared to MOEA/D. The solutions represented in red are the non-dominated solutions. Since it is a discrete optimization problem the curve is discontinuous. The plots in Fig. 4a,b represent the results obtained for the strong negative dataset and Fig. 4c,d represent the results obtained for the strong positive dataset. Fig. 4a,c are the results of the MOPSO algorithm and Fig. 4b,d are the results of the MOEA/D algorithm.

Table 2: 30 unique non-dominated solutions collected via independent runs

	MOPSO				MOEA/D			
	Strong positive		Strong negative		Strong positive		Strong negative	
N.D. solution	Resource wastage	InterVM.C	Resource wastage	InterVM.C	Resource wastage	InterVM.C	Resource wastage	InterVM.C
1	5.8087	5.5473	13.004	5.5798	5.0773	5.5523	13.6396	5.5571
2	5.5667	5.5492	13.6952	5.5508	7.1938	5.5292	12.6929	5.5731
3	6.0352	5.5462	13.2778	5.5742	5.7185	5.534	11.2201	5.5746
4	6.2173	5.5317	13.6523	5.5687	7.21	5.5237	15.73	5.5471
5	5.1579	5.5649	12.6	5.5916	4.8081	5.5685	14.3803	5.5489
6	6.0589	5.5391	14.627	5.5481	5.8435	5.5328	18.1012	5.5395
7	5.2243	5.5506	12.4265	5.5933	7.0338	5.5313	11.7388	5.584
8	5.6027	5.545	12.7573	5.5897	6.5272	5.5321	12.0351	5.5746
9	5.0939	5.1579	13.7678	5.5676	5.0035	5.5592	12.8283	5.5744
10	6.1954	5.5412	13.8698	5.5689	4.5092	5.5826	12.8628	5.5643
11	7.2336	5.5391	13.7769	5.5695	4.5309	5.5571	12.8755	5.5508
12	5.4784	5.5584	14.937	5.5634	4.6172	5.6017	14.2315	5.5546
13	5.2527	5.5651	12.2899	5.5851	4.6781	5.5412	18.3802	5.5454
14	5.461	5.5649	14.4308	5.5666	4.8002	5.5593	15.188	5.5506
15	5.1579	5.5462	14.9924	5.5632	5.0082	5.5449	12.2527	5.5622
16	5.5355	5.5342	13.157	5.5702	5.4272	5.5468	11.7787	5.5887
17	5.1585	5.5578	12.6322	5.5813	5.6186	5.5361	11.8747	5.583
18	5.1391	5.5674	14.6709	5.5687	5.6549	5.5311	12.64	5.5561
19	6.4327	5.5361	14.5368	5.5695	5.6812	5.5424	14.5911	5.554
20	4.8975	5.5363	14.7723	5.5679	5.7438	5.5347	12.6363	5.5599
21	6.7992	5.5321	12.2454	5.5872	5.8439	5.5367	11.6016	5.5798
22	4.8499	5.5618	14.8549	5.537	5.9085	5.5336	13.5892	5.5565
23	5.3958	5.5613	15.1404	5.5521	6.029	5.5317	14.8164	5.5561
24	5.7831	5.5389	12.7634	5.5586	6.1101	5.5239	11.8939	5.5758
25	4.8244	5.5738	17.9699	5.5519	6.8806	5.5195	15.233	5.5473
26	5.5561	5.5574	18.5899	5.5441	6.085	5.5305	12.9108	5.5601
27	5.2947	5.5668	14.8721	5.5618	4.3649	5.5529	11.9508	5.5496
28	6.3629	5.5384	12.6904	5.563	4.898	5.5523	11.6988	5.5794
29	4.8129	5.5866	17.0495	5.5494	5.0053	5.5361	11.8746	5.5738
30	5.3615	5.5641	14.9155	5.5529	6.1212	5.5201	12.8344	5.5685
Min	4.3649	5.1579	11.2201	5.537	4.8129	5.5195	12.2454	5.5395
Max	7.0233	5.5866	18.2589	5.5332	7.21	5.6017	18.3802	5.5887
S.Dev	0.5994	0.0733	1.5797	0.0125	0.8241	0.0184	1.8149	0.0135
Avg	5.5916	5.5387	12.1655	5.5602	5.5977	5.5426	13.336	5.563

Minimizing the impact on networking devices during inter-VM communication leads to reduced latency, improved throughput, better resource utilization, and decreased packet loss, all of which enhance overall system performance. This optimization is essential for maintaining the efficiency and reliability of cloud environments, especially in scenarios where real-time data exchange and high performance are critical.

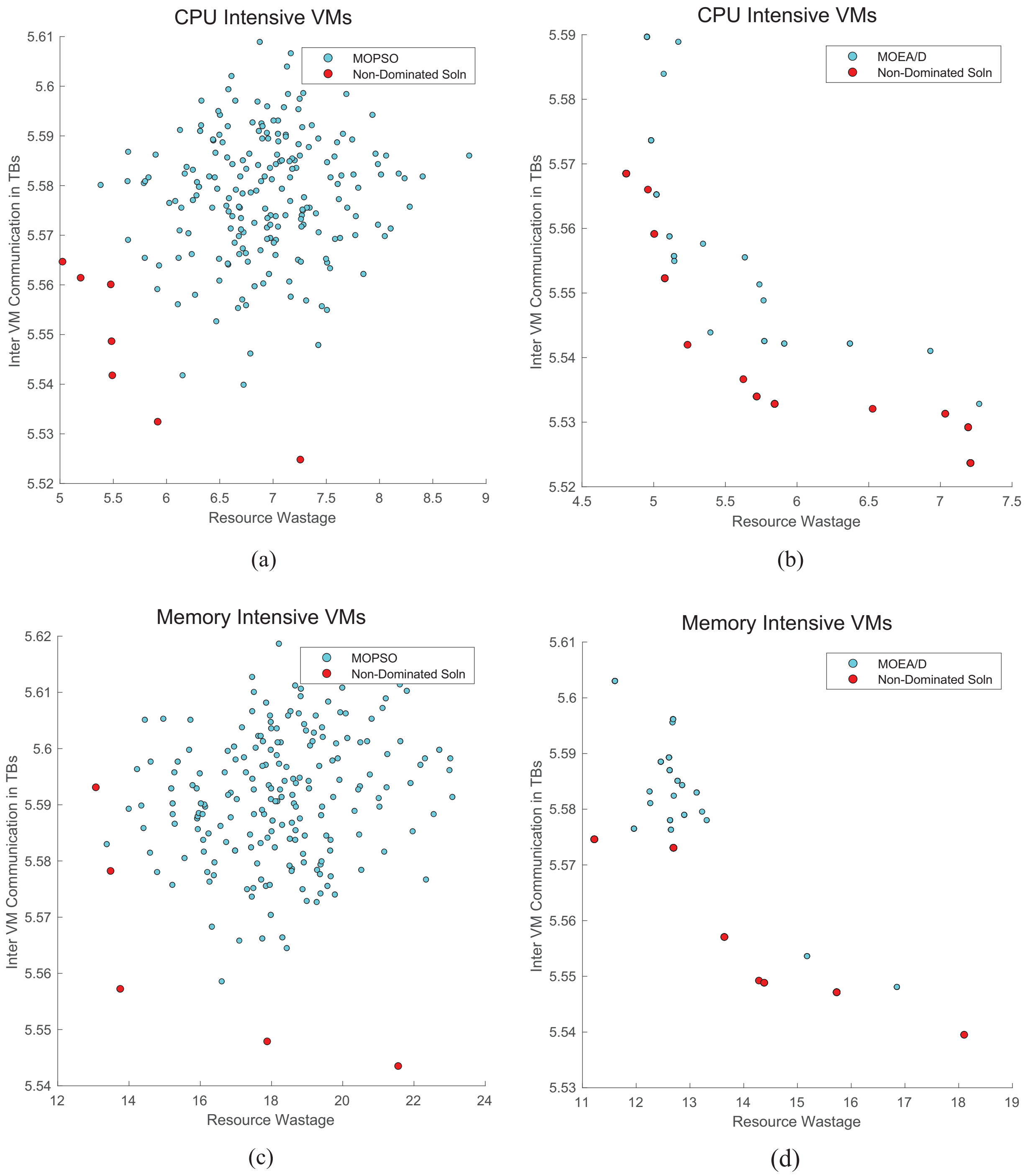

Figure 4: MOPSO *vs*. MOEA/D in solution exploration to attain global minima

7 Conclusion

This paper addresses the multi-objective version of virtual machine placement considering resource wastage and inter-VM communication as a minimization objective. The problem is designed for flat tree topology with 200 VM are scheduled to be placed in the physical machines. The mathematical model for the problem is illustrated with an example. The NP-Hard problem is effectively tackled by the proposed MOPSO algorithm, which performs considerably better than the MOEA/D algorithm. The algorithm is designed based on the concept of particle swarm optimization and adopted to address multi-objective problems. The experimental results show that the proposed algorithm explores the search space more effectively and can able to find more unique Pareto optimal solutions. In the future, we like to consider various networking architectures measuring the networking impact with the occurrence of VM migration.

Future work will involve refining the experimental setup by incorporating detailed performance metrics such as energy consumption and runtime analysis, as well as comparing with a broader range of multi-objective optimization techniques. Additionally, statistical validation will be used to demonstrate the significance of the results, ensuring a robust and scalable solution for cloud data center optimization.

Acknowledgement: None.

Funding Statement: This Research is funded by Researchers Supporting Project Number (RSPD2025R 947), King Saud University, Riyadh, Saudi Arabia.

Author Contributions: The authors confirm their contribution to the paper as follows: Praveena Nuthakki: Investigation, writing—original draft, conceptualization, software. Pavan Kumar T.: Conceptualization, supervising, writing original draft, formal analysis. Musaed Alhussein: Conceptualization, methodology. Mahammad Shahid Anwar: Formal analysis, supervising, writing—review and editing. Khursheed Aurangzeb: Methodology, design, editing, writing—review and editing. Leenendra Chowdary Gunnam: Writing, original drafting, software. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data used to support the findings of this study are available from the corresponding author upon request.

Ethics Approval: This article does not contain any studies with human participant and animals performed by author.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] 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. Syst. Appl.*, vol. 120, pp. 228–238, 2019. doi: 10.1016/j.eswa.2018.11.029.

[2] A. Shehabi *et al.*, "United states data center energy usage report," 2016. Accessed: May 15, 2022. [Online]. Available: https://escholarship.org/content/qt84p772fc/qt84p772fc.pdf

[3] A. Greenberg *et al.*, "VL2: A scalable and flexible data center network," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 4, pp. 51–62, 2009. doi: 10.1145/1592568.1592576.

[4] J. Krzywda, A. Ali-Eldin, T. E. Carlson, P. O. Östberg, and E. Elmroth, "Power-performance tradeoffs in data center servers: DVFS, CPU pinning, horizontal, and vertical scaling," *Future Gener. Comput. Syst.*, vol. 81, pp. 114–128, 2018. doi: 10.1016/j.future.2017.10.044.

[5] J. W. Lin, C. H. Chen, and C. Y. Lin, "Integrating QoS awareness with virtualization in cloud computing systems for delay-sensitive applications," *Future Gener. Comput. Syst.*, vol. 37, pp. 478–487, 2014. doi: 10.1016/j.future.2013.12.034.

[6] K. Fleszar and K. S. Hindi, "New heuristics for one-dimensional bin-packing," *Comput. Oper. Res.*, vol. 29, no. 7, pp. 821–839, 2002. doi: 10.1016/S0305-0548(00)00082-4.

[7] J. Sgall, "Online bin packing: Old algorithms and new results," in *Language, Life, Limits*, A. Beckmann, E. Csuhaj-Varjú, and K. Meer, Eds. Cham: Springer, Jun. 2014, vol. 8493. doi: 10.1007/978-3-319-08019-2_38.

[8] M. Labbé, G. Laporte, and S. Martello, "An exact algorithm for the dual binpacking problem," *Oper. Res. Lett.*, vol. 17, no. 1, pp. 9–18, 1995. doi: 10.1016/0167-6377(94)00060-J.

[9] J. Xu and J. A. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *IEEE/ACM Int. Conf. Green Comput. Commun. Int. Conf. Cyber Phys. Soc. Comput.*, Hangzhou, China, IEEE, Dec. 2010, pp. 179–188.

[10] M. Riahi and S. Krichen, "A multi-objective decision support framework for virtual machine placement in cloud data centers: A real case study," *J. Supercomput.*, vol. 74, no. 7, pp. 2984–3015, 2018. doi: 10.1007/s11227-018-2348-z.

[11] J. Lu, W. Zhao, H. Zhu, J. Li, Z. Cheng and G. Xiao, "Optimal machine placement based on improved genetic algorithm in cloud computing," *J. Supercomput.*, vol. 78, no. 3, pp. 3448–3476, 2022. doi: 10.1007/s11227-021-03953-8.

[12] A. S. Abohamama and E. Hamouda, "A hybrid energy-aware virtual machine placement algorithm for cloud environments," *Expert. Syst. Appl.*, vol. 150, 2020, Art. no. 113306. doi: 10.1016/j.eswa.2020.113306.

[13] J. Peake, M. Amos, N. Costen, G. Masala, and H. Lloyd, "PACO-VMP: Parallel ant colony optimization for virtual machine placement," *Future Gener. Comput. Syst.*, vol. 129, pp. 174–186, 2022. doi: 10.1016/j.future.2021.11.019.

[14] K. Balaji, P. S. Kiran, and M. S. Kumar, "Resource aware virtual machine placement in IaaS cloud using bio-inspired firefly algorithm," *J. Green Eng.*, vol. 10, pp. 9315–9327, 2020.

[15] S. Parida, B. Pati, S. C. Nayak, and C. R. Panigrahi, "eMRA: An efficient multi-optimization based resource allocation technique for infrastructure cloud," *J. Ambient Intell. Humaniz. Comput.*, vol. 14, no. 10, pp. 1–19, 2022.

[16] A. Gopu and V. Neelanarayanan, "Multiobjective virtual machine placement using evolutionary algorithm with decomposition," in *Proc. 6th Int. Conf. Big Data Cloud Comput. Chall.*, Singapore, Springer, 2020, pp. 149–162.

[17] A. Gopu and N. Venkataraman, "Optimal VM placement in distributed cloud environment using MOEA/D," *Soft Comput.*, vol. 23, no. 21, pp. 11277–11296, 2019. doi: 10.1007/s00500-018-03686-6.

[18] R. K. Chaudhary, R. Kumar, K. Aurangzeb, J. Bedi, M. S. Anwar and A. Choi, "Enhancing clustered federated learning using artificial bee colony optimization algorithm for consumer IoT devices," *IEEE Trans. Consum. Electron.*, 2024. doi: 10.1109/TCE.2024.3478349.