Available online at https://sanad.iau.ir/journal/ait
Publisher : Ya.C., Islamic Azad University
Artificial Intelligence Tools in Software and Data Engineering

OptiGrid: Optimizing Many-Objective Grid-Based Evolutionary Algorithm for Efficient VM Placement in Cloud

Reihaneh Khorsand

Department of Computer Engineering, Isf.C., Islamic Azad University, Isfahan, Iran.

R.khorsand@khuisf.ac.ir

Article type: Research Article

Article history: Received 17 Jun. 2025, Revised 20 Aug. 2025, Accepted 16 Oct. 2025, Published 28 Nov. 2025

***Abstract*—** Virtual machine (VM) placement is a process of dynamically mapping VMs to physical machines (PMs) in Cloud datacentre. For optimal virtual machine placement, there is a need for a many-objective optimization approach in a timely manner that can achieve a trade-off between meeting the cloud service provider's requirements and the user defined QoS parameters. Most of the existing researches on virtual machine placement focus on a single objective or multi objectives into account while there is a lack of effective approaches on problems with more than three objectives, which are often known as many-objective optimization problems. Therefore, in this paper, we present a self-organization framework for virtual machine placement in cloud environment, where key is to orient whether service placement is required or not in each period. In addition, an improved many-objective grid based evolutionary algorithm (iGrEA) is proposed to virtual machine placement, considering the following three objective functions: (1) minimizing energy consumption, (2) minimizing migration time, and (3) minimizing response time. The proposed approach is evaluated via simulation and experiments on real traces. The results show that the proposed approach is more efficient and effective than the other tested approaches.

Keywords: Cloud computing; Virtual machine placement; Many-objective optimization; GrEA evolutionary algorithm.

I. INTRODUCTION

Cloud computing is a popular paradigm that uses the Internet and provides a computing service as a model (Mejahed & Elshrkawey, 2022). Cloud computing provides a shared pool of virtual resources that can be used on demand (Khodayarseresht & Shameli-Sendi, 2023). Cloud service providers in their position have computing facilities called data centers and they are able to provide the necessary space needed for customers' business (Ali et al., 2019). One of the key technologies used in cloud computing is virtualization. Virtualization enables dynamic sharing of physical resources on cloud-based data centers (Li et al., 2019). In virtualization, each virtual machine acts as an independent execution unit just like a real computer mounted on a physical machine (Ali et al., 2019). Virtualization technology also offers a demand-based presentation model in which unlimited computing

Cite this article: Reihaneh Khorsand (2025). OptiGrid: Optimizing Many-Objective Grid-Based Evolutionary Algorithm for Efficient VM Placement in Cloud. ***Journal of Artificial Intelligence Tools in Software and Data Engineering (AITSDE),*** 3 (3), 18-39.

resources are available for applications in the form of processor, memory, disk space, etc. (Infantia-Henry et al., 2022). The process of dynamically mapping virtual machines to physical machines in cloud data centers is called virtual machine placement. In big data center management, many optimization objectives are considered when choosing a placement depends on business goals. Most studies in the field of virtual machine placement focus only on single-objective optimization, however, many real-world problems have different purposes that need to be optimized. One of the major concerns of the virtual machine placement process is the reduction of energy consumption in physical machines while they are running, which has been studied as a goal function in the virtual machine placement literature (Khallouli et al., 2022). Cloud data centers include a large number of physical computing machines that consume significant amounts of energy for their operations. Given the environmental effects and economic impact of energy consumption of physical machines in recent years, the development of energy-aware virtual machine placement techniques to minimize energy consumption in data centers has been highly regarded by cloud service providers (Gabhane et al., 2021). However, reduced power consumption may reduce system performance, leading to breaches of service level agreement during peak download times. Therefore, a balance must be struck between saving energy and maintaining system performance, which requires the quality of user-defined services. Although the optimization problem involves only three objectives—commonly considered the boundary between multi-objective (MOA) and many-objective optimization (MaOA)—we adopt a MaOA framework due to the high conflictuality, non-linear trade-offs, and discrete, heterogeneous nature of the VM placement problem in cloud data centers. Specifically, aggressive minimization of energy consumption often forces excessive VM consolidation, which degrades response time and increases SLA violations, while minimizing migration time favors static placements that hinder dynamic load balancing. These interactions create a complex and irregular Pareto front that challenges conventional MOA algorithms. As empirically demonstrated in Section 4.3, even NSGA-III—a state-of-the-art MaOA designed for ≥3 objectives—struggles to maintain solution diversity and convergence under these conditions. Therefore, employing a grid-based MaOA such as iGrEA is both methodologically justified and empirically necessary. In this paper, a self-organization virtual machine placement framework is designed in Cloud computing. At first, in a observe phase, information of the resources and requests are collected in a continuous manner. Next, in a orient phase, a decision tree algorithm is developed to analyze resource condition and workload for orientation whether service placement is required or not in each period, and then, in a decide phase, the decisions are made regarding virtual machine placement by a grid based evolutionary algorithm. This algorithm is a many-objective optimization problem with more than two objectives (Lopez-Pires & Baran, 2017). Since the search ability for Pareto-based evolutionary multi-objective optimization algorithms deteriorates significantly when it reaches more than two objective functions. Therefore, one of the objectives of Pareto-based optimization algorithms is to increase the selection pressure towards the Pareto front while maintaining a uniform distribution among the solutions (Chen & Li, 2019). The main contributions of this article can be summarized as follows:

- A framework is designed for self-organization virtual machine placement in Cloud computing.
- A decision tree algorithm is developed to analyze resource condition and workload for orientation whether service placement is required or not in each period.

- A many-objective grid based evolutionary algorithm is proposed as a decision maker of the proposed framework for virtual machine placement.
- A series of experiments to evaluate the performance of proposed approach in terms of different metrics are conducted.

The remainder of the article is structured as follows: Section 2 offers a comprehensive survey of related works. In Section 3, the proposed framework for self-organization virtual machine placement is described. Following that, Section 4 presents the evaluation results of the proposed approach. Finally, Section 5 provides a discussion on the conclusions drawn from the study and outlines potential future work.

II. RELATED WORKS

Cloud computing allows the access to the shared data in cloud resources. In the cloud environment, virtual machine placement is used to maximize resource utilization and increase service quality. Virtual machine placement is a complex and NP-hard problem that has been the subject of much research (Lopez-Pires & Baran, 2017). In data centers, despite the significant number of physical machines and virtual machines, a large number of possible criteria are considered when locating according to the business and optimization objectives. In 2013, Gao et al. proposed the multi-objective ant colony system algorithm (VMPACS) for placing the virtual machine. Despite the scalability of the algorithm in big data centers, by searching in the solution space, it can search for effective solutions to simultaneously improve the waste of resources and energy consumption and reduce costs. The single-objective ant colony optimization (SACO) compared to the first fit decreasing algorithm (FFD) can find a number of solutions with a small number of servers used and a larger number of servers compared to the multi-objective ant colony algorithm. In 2023, Wang et al. implemented a problem-specific three-layer encoding strategy and introduced a decomposition-based multi-objective evolutionary algorithm called MOEA/D-PD. They also proposed the use of a classifier model to filter the offspring solutions in the decision space, ensuring that only promising solutions are evaluated. Moreover, computational resources are dynamically allocated to the sub-problems based on their contributions. In 2018, Gupta and Amgoth proposed the Resource Aware Virtual Machine Placement (RVMP) algorithm with the aim of using resources efficiently on heterogeneous hosts dynamically. The resource utilization factor (RUF) technique is intended to position the virtual machine in order to balance the multi-dimensional resource on each physical machine as well as to reduce the number of active physical machines. In addition, the proposed algorithm using minimum active hosts, reduces resource wastage and energy consumption. In 2022, Zolfaghari et al. used Host Utilization Aware Algorithm (HUA) for virtual machine placement with the aim of efficient resource efficiency, reducing energy consumption, and agreeing service levels. This method is used to calculate the lowest threshold value to detect low-load hosts. In such a way that the number of hosts and restrictions are considered for identifying inactive hosts. Host utilization aware algorithm decision making is efficient on low-load hosts and frees up many hosts, resulting in energy savings while maintaining breaches of service level agreements. In 2023, Khaleel introduced CSSA-DE, an algorithm that combines the sparrow search algorithm and differential evolution optimization. The aim was to enhance the search efficiency in finding suitable virtual machine (VM) placements. Additionally, Khaleel proposed a clustering approach for organizing computing nodes. The integration phase of the algorithm leverages the number of overloaded and

underloaded VMs, thereby minimizing resource fragmentation. In 2021, Tang et al. proposed an architecture for scheduling tasks using the Particle Swarm Optimization (PSO) algorithm with the aim of improving dynamic resource allocation, energy savings, and cost while ensuring quality cloud services. In the search domain algorithm for decision-making, there are no restrictions that provide local search solutions with optimal search. The proposed architecture reduces the number of migrations and simulation time in cloud infrastructure compared to other evolutionary algorithms. In 2019, Abazari et al. applied a Multi-objective Biogeography based optimization/Differential Evolution algorithm called MBBO/DE to solve the problem of virtual machine integration in heterogeneous distributed systems. In 2023, Nabavi et al. introduced the Seagull optimization algorithm for multi-objective virtual machines placement in edge-cloud data centers. Their approach aimed to minimize network traffic by grouping virtual machines on the same physical machines, reducing data transfer through the network. Additionally, they consolidated virtual machines onto fewer physical machines, leading to decreased power consumption and energy savings. In 2013, Adamuthe et al. proposed the Non-dominated Sorting Genetic Algorithm Version 2 (NSGA-II) with the aim of finding the best solutions in the population. Virtual machine placement is done as a multi-objective optimization problem, with the objectives of maximizing profits, load balancing and reducing resource wastage. In this algorithm, with an elite mechanism that includes a combination of the best parents with the best children, it is suitable for solving different scheduling problems. In 2022, He et al. proposed the grid-based evolutionary algorithm that works like a genetic algorithm and aims to select the best solution while maintaining uniformity and even distribution. In this algorithm, a parameter called Grid partitioning has been introduced to partition the Grid environment.

III. PROPOSED FRAMEWORK

Fig. 1 shows the proposed self-organization virtual machine placement framework in cloud computing. Initially, in the proposed framework, requests are queued to be appropriately assigned to virtual machines. Then the four phases of observe, orient, decide and act are implemented as follows:

- **Observe phase**

This phase observes the state of the system in the same time intervals and collects the status of busy virtual machines on each physical machine and the amount of load in each stage by the observe component. The collected data is stored in the database and thus it is available for the next phase, which is orientation.

- **Orient phase**

The task of the orient phase is to decide whether or not to need a virtual machine placement. Based on the resource utilization of the physical machines, detecting the violation of the service level agreements and the workload level, this unit determines whether the cloud needs a virtual machine placement or not, using a decision tree. Each of these three parameters are calculated as follows and used in the decision tree.

a) Resource utilization: In the proposed approach for forming a decision tree, priority is given to resource utilization because this factor is the most effective indicator for optimizing energy consumption. Then, the amount of violation of the service level agreement and workload are discussed.

To calculate the resource utilization of physical machines, first, the amount of MIPS consumed in the virtual machines of each physical machine is calculated according to Equation 1. Then, according to the amount of MIPS available in each physical machine, the utilization is calculated from equation 2.

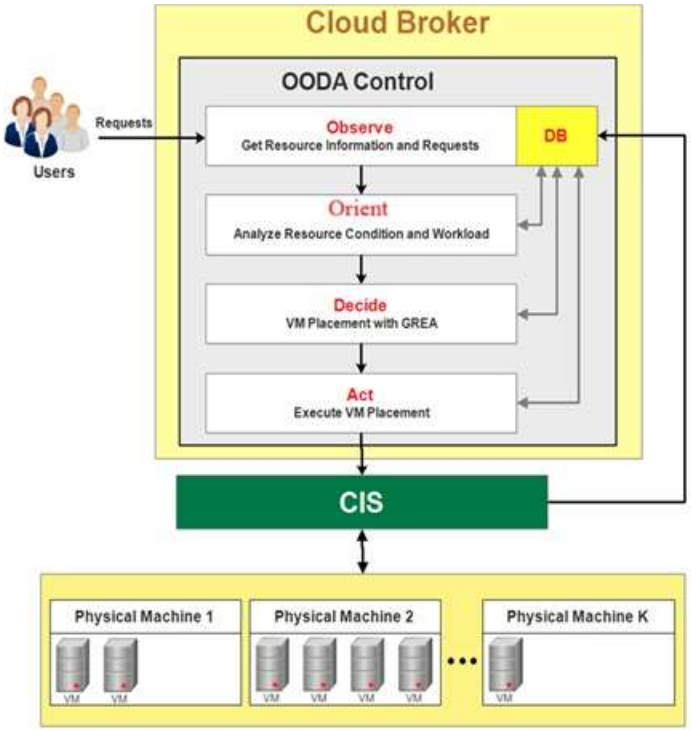


Fig. 1: Proposed self-organization virtual machine placement framework

$$Sum\_MIPS_j = \sum_{i=1}^{M} Allocated\_MIPS_i \quad (1)$$

where M is the number of virtual machines in the *jth* physical machine.

$$Utilization_j = \frac{Sum\_MIPS_j}{Available\_MIPS_j} \quad (2)$$

where $Available\_MIPS_j$ is the amount of MIPS available in *jth* physical machines.

According to the number of physical machines (K), the average total resource utilization is calculated as equation 3.

$$Utilization = \frac{\sum_{i=1}^{K} Utilization_i}{K} \quad (3)$$

Finally, the conditions regarding resource utilization in the decision tree are in the form of equation 4.

$$Utilization = \begin{cases} Under & Utilization < U\_Lt \\ Normal & U\_Lt \leq Utilization < U\_Ht \\ Over & Utilization \geq U\_Ht \end{cases} \quad (4)$$

where U_Lt is the upper threshold limit of utilization and U_Lti is the lower threshold limit of utlization, which characterizes a low-utilization, normal and over-utilization system.

b) Service level agreement violation: If the response time exceeds the deadline for the request, equation 5 shows how to calculate the number of violations of the service level agreement regarding the response time.

$$SLAViolation_i = ResponseTime_i - DeadlineTime_i \tag{5}$$

$$SLAViolation_i = \begin{cases} 1 & SLAViolation_i > 0 \\ 0 & SLAViolation_i \leq 0 \end{cases}$$

$$SLAV = \sum_{i=1}^{N} SLAViolation_i$$

where $ResponseTime_i$ is the response time for the *i-th* request and $DeadlineTime_i$ is the deadline for the *i-th* request. *N* is the number of requests. $SLAViolation_i$ is the presence or absence of a violation of the service level agreement, and $SLAV$ is the total count of the service level agreement.

The conditions for a deadline violation in the decision tree are in the form of equation 6.

$$SLAV = \begin{cases} Yes & ResponseTime_i \geq Deadline \\ No & ResponseTime_i < Deadline \end{cases} \tag{6}$$

c) Workload: The conditions for workload in the decision tree are in the form of equation 7.

$$\text{WL\_Th} = \frac{70 \times K \times M}{100} \tag{7}$$

$$Workload = \begin{cases} Low & WL < \text{WL\_Th} \\ High & WL \geq \text{WL\_Th} \end{cases}$$

where WL_Th is the workload threshold limit, which is the limit for deciding whether the workload is high or low, and it is considered as 70% of the number of virtual machines.

According to the determined conditions, finally, the rules of the decision tree are specified below and the decision tree for deciding on the need for a service placement is shown in Fig. 2.

- **Decide phase**

This phase receives information from the orient phase and makes decisions about the placement of virtual machines by using an improved many-objective grid based evolutionary algorithm named iGrEA. Algorithm 1 illustrates the pseudo-code for our proposed algorithm. In the subsequent sections, we will introduce the proposed algorithm and delve into its workings, explaining its key details and steps.

Step 1: Generating the initial population

In our approach, we start by creating an initial population of elements. Each element is represented by an array, as illustrated in Fig. 3. The value stored in each array element corresponds to the number of busy virtual machines (M).

if *(Utilization=Over) and (SLAViolation =Yes)*

VMP=yes

if *(Utilization=Over) and (SLAViolation =No) and (Workload =High)*

VMP=yes

if *(Utilization=Over) and (SLAViolation =No) and (Workload =low)*

VMP=No

if *(Utilization=Normal) and (SLAViolation =Yes)*

VMP=yes

if *(Utilization= Normal) and (SLAViolation =No) and (Workload =High)*

VMP=yes

if *(Utilization= Normal) and (SLAViolation =No) and (Workload =low)*

VMP=No

if *(Utilization=Under)*

VMP=yes

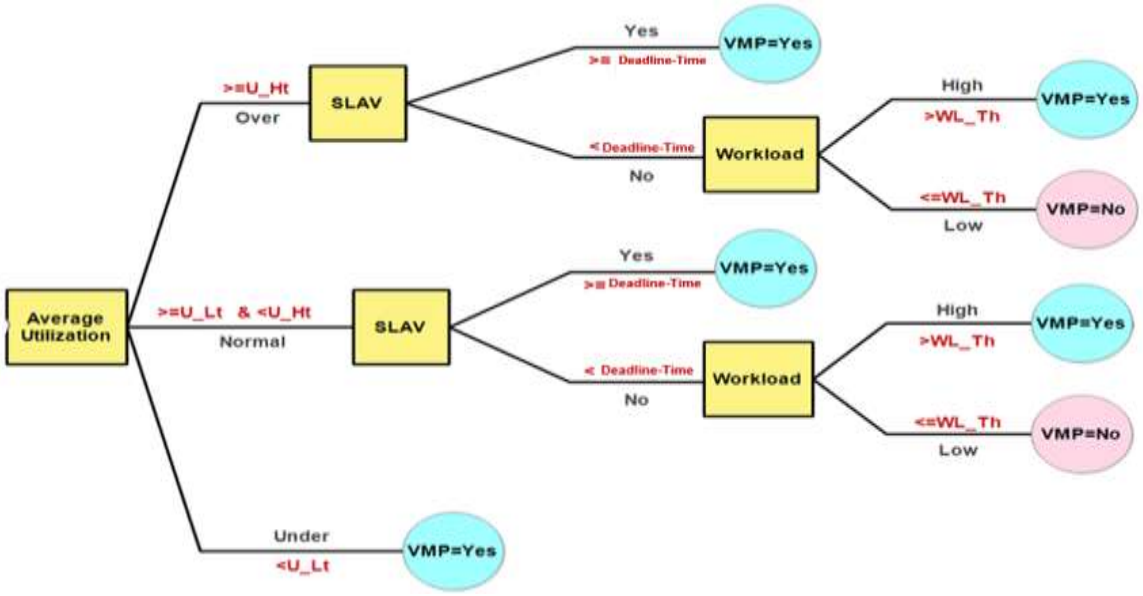


Fig. 2: The structure of the decision tree on the orient phase in the proposed approach

VM1	VM2	VM3	VM4	VM5
H_{11}	H_2	H_8	H_{11}	H_{16}

Fig. 3: The structure of an element of a population

Algorithm 1: Virtual machine placement using the iGrEA algorithm

```
Input: req, VMS
Output: Best pop
// Step 1: Generating the initial population by the number of generations (N) using
the VMS vector
1:      Hosts = Get Hosts ( );
2:      for each  H  in Hosts
3:         for each vm in   H
4:             if vm, Used-MIPS > 0
5:                VMS[c], id = vm, id;
6:                VMS[c], Host id = H, id;
7:                 c = c + 1;
8:          end if
9:     end for
10:  end for
11:  pop = Create-Population(Hosts, VMS, N);
12:  while iter < Generation count do
// Step 2:  Valuation of population elements using the fitness function
13:          F = Fitness-Function(pop ,req);
// Step 3: Grid settings and calculations
14:         Grid-setting(pop ,F);
15:         Grid-Calculate (pop, F);
// Step 4: Tournament-Selection Function
16:          P' = Tournament-Selection(pop ,F);
// Step 5: Environmental selection function
17:          P" ←   Environmental-selection(pop ,P',N);
18:          pop ←  CrossOver(P");
19:          iter + +;
// The algorithm terminates when the number of generations reaches the threshold
20:  end while
// Extracting the most valuable element of the population
21:  return   pop;
```

In this array, the index of each element represents the virtual machine number and the value for it, and for each virtual machine, it is the physical machine ID assigned to it for execution. For example, in Fig. 3, the physical machine H_{11} is considered for two virtual machines 1 and 4, the physical machine H_2 is considered for virtual machine 2. Initially, the status of busy virtual machines (the amount of MIPS consumed by the virtual machine) is received within the same time of requests. Indices of busy virtual machines and physical machines are stored in Virtual Machine Vectors (VMS). The initial population is repeated based on the physical machine ID and virtual machine vector (VMS) to the specified initial population number (N) (line 12) and is placed in the population vector (POP) (lines 1-11).

Step 2: Valuation of population elements using the fitness function

The initial population value (pop) is determined by the fitness function (Line 13). First, a logical array is created for the physical machines selected for the element. For example, for the element of Fig. 3, the logical array of physical machines would be as shown in Fig. 4. If a physical machine is selected for each virtual machine, the value of the physical machine is set to 1 and, 0 if not selected.

H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}	H_{12}	H_{13}	H_{14}	H_{15}	H_{16}	H_{17}	H_{18}	H_{19}	H_{20}
0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0

Fig. 4: The selection of physical machines (H) for the element of Fig. 1

The act of valuing population elements is performed using the fitness function. The fitness function is based on three objective functions of energy consumption, migration time and response time as follow.

a) Energy consumption: The energy consumption resulting from the placement for each element is calculated according to Equation 8.

$$Power = \sum_{j=1}^{K} H_j \times (((P_j^{Busy} - P_j^{Idle}) \times \sum_{x_i \in X | xi = j} D_{x_i}^{CPU}) + P_j^{Idle}) \quad (8)$$

where, P_j^{Busy} and P_j^{Idle} represent the energy consumption of the physical machine j in fully busy and idle mode, respectively. $D_{x_i}^{CPU}$ is the efficiency of the ith virtual machine that the jth physical machine is selected as the host.

b) Migration Time: The time of information transport from a physical machine to another physical machine. The migration time of a virtual machine from physical machine *i* to physical machine *r* is calculated according to Equation 9.

$$Time_j^{i,r} = T_{Transport}^{i,r} + T_{Down}^{i,r} \quad (9)$$

where $T_{Transport}^{i,r}$ is the time that it takes for virtual machine information to transfer from physical machine *i* to physical machine *r*. $T_{Down}^{i,r}$ is the time to turn off the virtual machine on the ith server. According to the calculation of the migration time of a virtual machine according to Equation 9, the calculation time of all virtual machines in an element is calculated according to Equation 10.

$$Migration = \sum_{j=1}^{K} H_j \times \sum_{x_i \in X | xi = j} Dis(r,j) \times Time_{X_i}^{rj} \quad (10)$$

where, *r* is the index of the physical machine that hosts the virtual machine x_i and $Dis(r,j)$ is the distance between the physical machine *r* and *j*.

c) Response time: The time considered for responding to the request. When the placement is appropriate, the time to respond to requests is reduced.

Step 3: Grid settings and calculations

Grid settings are calculated based on the initial population and the vector obtained from the fitness function (lines 14 and 15). In the Grid settings for the elements, the grid ranking (GR), the grid congestion distance (GCD) and the grid coordinate point distance (GCPD) are calculated (Abazari et al., 2019), which is shown in Fig. 5 to better understand these concepts. Grid ranking is based on the grid position of the elements. Grid ranking refers to the disparity in objective values between two solutions. When one element outperforms its competitor in most objectives, it is assigned a lower ranking value. Equation 11 defines the rank for each element as the aggregate of the grid values across all objectives.

$$GR(x) = \sum_{k=1}^{M} G_k(x) \quad (11)$$

where, $G_k(x)$ represents the grid value of the element x in the K^{th} objective and M in the number of objective. For example, in Fig. 5, with respect to elements A and C, C has a worse rank value than A (6 vs. 4) because the benefit of the objective function $f2$ is less than the loss of the objective function $f1$.

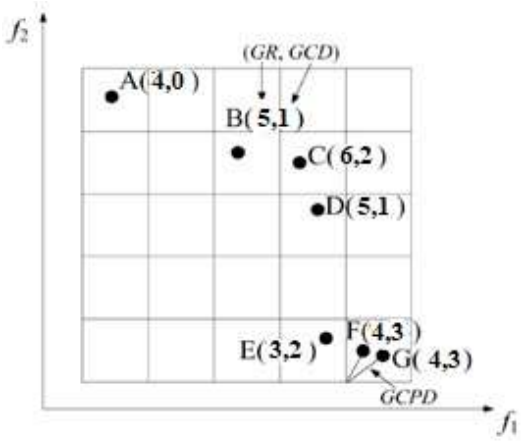


Fig. 5: Valuation of solutions according to GR, GCD and GCPD (Gabhane et al., 2021)

In valuation, it is crucial to estimate the density of solutions as it greatly impacts the search process. Well-distributed solutions are highly desirable. The congestion distance of the Grid is influenced by the neighborhood range, denoted as M. A larger neighborhood range encompasses more solutions. On the other hand, the congestion distance of the information Grid indicates the position of solutions within the neighborhood, meaning that more distant neighbors have a smaller influence. For instance, in Fig. 5, considering elements C and F, the GCD for C is smaller compared to F (2 vs. 3), despite having the same number of neighbors. The density estimation is defined according to the neighbor's distribution of a solution (Equation 12).

$$GCD(x) = \sum_{y \in N(x)} \left(M - GD(x,y)\right) \tag{12}$$

where, $N(x)$ is the set of neighbors of the element x, and $GD(x, y)$ represents the grid difference between the elements x and y, and M is the number of objectives. The distance between an element and its turning point in any grid cell is called the grid coordinate point distance (Equation 13).

$$GCPD(x) = \sqrt{\sum_{k=1}^{M} \left(\left(F_k(x) - (lb_k + G_k(x) \times d_k)\right) / d_k\right)^2} \tag{13}$$

where, $G_k(x)$ represents the grid value of element x in the k^{th} objective, while $F_k(x)$ represents its corresponding actual value. lb_k signifies the lowest border of the grid, and d_k represents the width of a grid cell. Lastly, M denotes the total number of objectives. The selection functions employ three grid-based criteria to compare elements: GR, GCD, and GCPD.

Step 4: Tournament-Selection Function

The dominant elements are selected as the first front (dominant Pareto) based on the strict dominance or Grid value to be transferred to the archive population (Line 16). The Grid dominance is performed by comparing elements based on their grid value (Equation 14).

$$Definition\ (Grid\ Dominance) : Let\ x, y \in P, x \prec grid^{y} :\Leftrightarrow \forall i \in (1, 2, \cdots, M) : G_i(x) \leq G_i(y) \wedge \exists j \in (1, 2, \cdots, M) : G_j(x) < G_j(y) \quad (14)$$

In the constructed Grid environment with a population size of P, M represents the number of objectives. $x \prec grid^{y}$ refers to the grid dominance of element y by element x. For minimization problems, the grid value of element x dominates the grid value of element y only if element x is superior to element y in at least one aspect, while element y is not better than element x in any aspect. The iGrEA algorithm leverages Grid dominance to prevent the storage of elements that have been dominated earlier than their competitors. This ensures that Grid-dominated elements still have a chance to enter the archive collection. First, an empty $N \times N$ matrix is considered, and first the elements are selected that are on the first Pareto front and have a strong dominance over all elements of the population. In this approach, the dominance between solutions in a population is determined by comparing their fitness function value (F) or grid value (G*)*. If solution *i* dominates solution *j*, the dominance matrix will have a value of one in row *i* and column *j*. Conversely, if solution *j* dominates *i*, the value in row *j* and column *i* will be one. If there is no clear dominance, the grid congestion distance (GCD) is used for comparison, and the dominance matrix is quantified accordingly. By calculating the sum of each column in the dominance matrix (as shown in Fig. 6), we can determine the number of times each solution is dominated by the population. These values are then sorted in ascending order. The solutions with the fewest dominations are selected as the Pareto front and added to the archive population called "picked." Algorithm 2 provides the pseudo-code for the Tournament-Selection Function.

Step 5: Environmental selection function

The environment selection function is utilized to complete the archive population (line 17). It extends the selection within the solution space by penalizing accompanying elements with the selected elements during the competitive selection stage. The process begins by creating an empty archive population, followed by adding the output solutions of the GR-Adjustment function to the archive population. The best population is then excluded from the original population and calculated for the remaining population (Rem_pop) using grid ranking value adjustment, grid congestion distance, and network coordinate point distance. Finally, the best solutions are selected from the population using the Findout-best function. These steps are repeated until the archive population matches the original population.

Findout-best function: The algorithm 3 shows the pseudo-code of findout-best function. This function selects the first solution as the preferred solution (Line 1 of Algorithm 3). For all remaining solutions from the initial population and the selected population, the value of the Grid rank, the Grid congestion distance and the coordinate point distance are calculated and the value of the selected solution is compared with all the solutions (lines 2-14 of Algorithm 3). If the selected solution has less value in each of these properties, this solution is selected as the preferred solution (Line 15 of Algorithm 3).

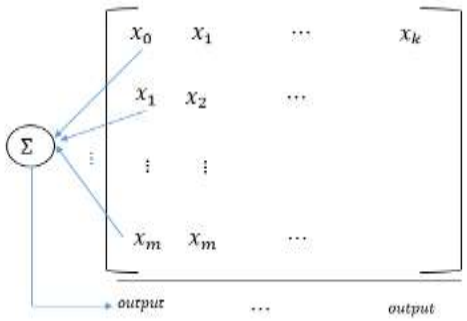


Fig. 6: Dominance matrix in the competitive selection function

1. Algorithm 2: Tournament_Selection()

2. Input: pop, F
3. Output: Picked

```
1:  for  i = 1  to  N − 1  do
2:      z = 1; t = 0; Dominate is Array N × N: {0}
3:      for  j = i + 1  to  N  do
4:          for  k = 1  to  5  do
5:              if ( pop(i).F(k) > pop(j).F(k) ) or ( G(pop(i).F(k)) > G(pop(j).F(k)))
6:                  z + +
7:              else if (pop(i).F(k) < pop(j).F(k) ) or ( G(pop(i).F(k)) < G(pop(j).F(k)))
8:                  t + +
9:          end  if
10:     end  for
11:     if (z > 1  and  t = 0)
12:         Dominate[i][j] = 1
13:     else  if (z = 1  and  t > 0)
14:         Dominate[j][i] = 1
15:     else if ( GCD(pop(i)) < GCD(pop(j)))
16:         Dominate[i][j] = 1
17:     else if ( GCD(pop(i)) > GCD(pop(j)))
18:         Dominate[j][i] = 1
19:     end if
20:   end  for
21: end for
22: for  i = 1  to  N  do
23:   grade[i] = 0
24:   for  j = 1  to  N  do
25:       grade[i] = grade[i] + Dominates[j][i]
26:   end for
27: end for
28: Sort_grade = Sort(grade)
29: for  i = 1  to  N  do
30:     if(grade[i] == Sort_grade[0])
31:         Picked.Add(pop[i])
```

35. 32: end if
36. 33: end for
37. 34: Return Picked

GR-Adjustment Function: Algorithm 4 provides the pseudo-code for the grid ranking function. This function assigns grid rank values to solutions, dispersing the selection of solutions in the problem space by penalizing those close to the selected solutions. The value of *M* is determined based on the number of objectives relevant to the problem. When a solution is placed in a cell with selected solutions, a penalty of *M + 2* is applied and added to the grid rank value (lines 1-3 of Algorithm 4). Solutions dominating the selected population are also penalized with a value of M (lines 4-6 of Algorithm 4). Solutions that neither match the selected solutions nor dominate them have a penalty of zero (lines 7-9 of Algorithm 4). The penalty amount for each of these solutions is determined (Line 10 of Algorithm 4). In the event that the penalty score is lower than the difference between the number of goals and the distance between the solution and the selected solutions, appropriate penalties are assigned to the solution (lines 11-12 of Algorithm 4). Subsequently, the same penalty is applied to all solutions dominated by the selected solution (lines 13-19 of Algorithm 4). Finally, the penalty value is added to their grid rank value (lines 20-22 of Algorithm 4). The algorithm terminates after a certain number of generations, at which point the solution with the highest value is selected as the best placement mode. During the selection of multiple virtual machine sets (P') using the environmental selection function, the crossover operator is applied to the set P', creating a new population consisting of parents and children called P'' (Line 18 of Algorithm 1). The steps are iteratively executed until the algorithm satisfies the termination criterion, which is the attainment of the specified number of generations. (lines 19-20 of Algorithm 1). Finally, the most valuable element within the population is extracted as the final result (Line 21 of Algorithm 1).

- **Algorithmic Distinction from Standard GrEA**

The proposed iGrEA introduces a critical enhancement over the standard Grid-based Evolutionary Algorithm (GrEA) [He et al., 2022] through its **GR-Adjustment function** (Algorithm 4). While standard GrEA relies solely on grid ranking, grid congestion distance, and grid coordinate point distance to maintain diversity, it does not actively penalize solutions residing in grid cells already occupied by selected elites. In contrast, iGrEA explicitly modifies the grid rank of such solutions to enforce spatial separation in the objective space.

The penalty values used in Algorithm 4—namely $M + 2$for solutions co-located with selected individuals, and $M$for solutions dominating selected individuals in crowded regions—are **mathematically grounded** in the dimensionality of the objective space. For a problem with $M = 3$objectives (energy, migration time, and response time), the maximum possible grid rank of any solution in an unoccupied neighboring cell is $M + 1 = 4$. By assigning a penalty of $M + 2 = 5$to solutions sharing a cell with a selected solution, iGrEA guarantees that such solutions are strictly ranked worse than any candidate in adjacent cells, thereby preventing premature clustering. Similarly, a penalty of $M = 3$is applied to dominant elites in dense regions to avoid redundant selection of near-identical placements. This design ensures both strong selection pressure toward the Pareto front and effective diversity preservation.

Theoretically, this mechanism introduces a repulsion force in the objective space: whenever a solution is accepted into the archive, its surrounding grid cells become less attractive for immediate re-selection. This directly mitigates a

known weakness of GrEA—loss of diversity under discrete, constrained search spaces like VM placement, where feasible solutions are limited and often clustered.

- **Act phase**

The task of this phase is to implement the decision sent by the decision-making unit. In this way, the placement of virtual machines is implemented by this phase. Upon termination of iGrEA, a set of non-dominated placements is returned. To support real-time decision-making, the cloud operator may specify QoS weights (α, β, γ). A simple weighted aggregation is then applied offline to rank the final Pareto-optimal solutions and select the single best configuration for deployment.

Algorithm 3: The pseudo-code of findout-best function

```
Input: Rem_pop
Output: Best
best = Rem_pop[1]
for i = 2 to |Rem_pop| do
    if (GR(Rem_pop[i]) < GR(best))
        best = Rem_pop[i]
    else if (GR(Rem_pop[i]) == GR(best))
        if (GCD(Rem_pop[i]) < GCD(best))
            best = Rem_pop[i]
        else if (GCD(Rem_pop[i]) == GCD(best))
            if (GCPD(Rem_pop[i]) < GCPD(best))
                best = Rem_pop[i]
            end if
        end if
    end if
end for
return best
```

IV. EVALUATION OF WXPERIMENTS

In real large-scale infrastructure, repeatable experiments are very difficult and costly therefore, the evaluation of experiments is performed in a cloud computing environment, which is more efficient than other computational models. In this section, the test environment setting and experiments of the proposed method with two multi-objective algorithms NSGA-II (Gu & Wang, 2020) and BBODE (He et al., 2022) are examined.

Algorithm 4: GR_Adjustment( )

```
Input: pop, picked
Output: GR
foreach p in Equal(pop, Picked)
    GR(p) = GR(p) + (M + 2)
end for
foreach p in Great(pop, Picked)
```

```
63.      5:      GR(p) = GR(p) + M
64.      6:   end for
65.      7:   foreach (p in Ngreat(pop, Picked)) and (p not in Equal(pop, Picked))
66.      8:      PD(p) = 0
67.      9:   end for
68.     10:   foreach (p in (NGE ∩ Ngreat(pop, Picked))) and (p not in Equal(pop, Picked))
69.     11:      if (PD(p) < (M − GD(p, Picked))
70.     12:         PD(p) = M − GD(p, Picked)
71.     13:         foreach (r in Great(pop, p)) and (r not in (Great(pop, Picked) ∪ Equal(pop, Picked)))
72.     14:            if (PD(r) < PD(p))
73.     15:               PD(r) = PD(p)
74.     16:            end if
75.     17:         end for
76.     18:      end if
77.     19:   end for
78.     20:   foreach (p in Ngreat(pop, Picked)) and (p not in Equal(pop, Picked))
79.     21:      GR(p) = GR(p) + PD(p)
80.     22:   end for
```

- **Test environment setting**

The experiments are simulated in a cloud-based computing environment. CloudSim provides key classes for defining data centers, including virtual machines, computing resources, user policies, and management policies. The configuration and type of physical machines and virtual machines are considered to be heterogeneous. Explanations of the specifications of virtual machines and physical machines have been given in Tables 1 and 2, respectively.

Table 1.The specifications of virtual machines

Machine Name	MIPS	Ram(GB)	Storage	BW
T1	500	1	1GB	100 Mbps
T2	1000	2	4GB	1 Gbps
T3	2000	4	3*4 GB	1 Gbps
T4	2000	6	1*16 GB	1 Gbps
T5	2500	8	64 GB	1 Gbps
T6	4000	8	2*64 GB	1 Gbps
T7	4500	16	2*64 GB	1 Gbps
T8	5000	32	4*64 GB	1 Gbps
T9	5500	16	8*16 GB	1 Gbps

All experiments in the data center are performed with five physical machines and nine types of virtual machines and cloud linux operating system with virtual machine management (XEN). The values presented in Table 3 were selected for the best compromise between search capability and convergence speed. Experiments are performed on real workloads, which were randomly selected from three datasets of FIFA (Dataset), ClarkNet (Dataset), NASA (Dataset). For an equitable evaluation with error reduction, the tests have been repeated several times per work and the final results reported are averaged.

Table 2. Specifications of physical machines

Parameter	Value	Parameter	Value
P_j^{Busy}	150w	Live migration time	Random (8-16ms)
P_j^{Idle}	90w	Time to shut down the virtual machine	Random (4-8ms)
Dis	4ms	Virtual machine startup time	Random(5-10ms)

Table 3. The parameters setting

CPU	#Core	MIPS	Ram (GB)	Bandwidth
Intel Xeon	4	4096	16	1Gbit/s
Intel Xeon	8	8192	32	1Gbit/s
Intel Xeon	8	16384	64	1Gbit/s

- **Evaluation of experiments**

To assess the performance of the proposed algorithm, three factors including energy consumption, migration time and response time are taken into account. A comparison was made between the proposed algorithm and two multi-objective algorithms, NSGA-III and BBODE, using various workload samples. In the workload experiments, they were randomly selected from three real data sets: FIFA, ClarkNet, and NASA. Details of the parameters of the iGREA and NSGA-III algorithms are presented in Table 4 and the Bio-Algorithm (BBODE) in Table 5. The recombination coefficient in population-based algorithms to obtain new children is the result of the combination operation on the parents.

Table 4: The parameters setting of the iGREA and NSGA-III algorithms

Termination condition	Number of generations	Initial population size
generations count	200	100

Table 5: The parameters setting of the BBODE algorithm

Termination condition	Foreign immigration rate	Internal migration rate	Number of generations	Initial population size
Number of generations	0.3	0.7	200	100

a)Evaluation of the proposed algorithm's energy consumption at three workloads

In this experiment, we compare the performance of the proposed algorithm with two other algorithms, NSGA-III and BBODE, in terms of average energy consumption. The number of requests is arranged in ascending order over a specific time period. Figure 5 displays the average energy consumption of the compared algorithms, which rises as the number of requests increases. The proposed algorithm demonstrates lower energy consumption compared to the

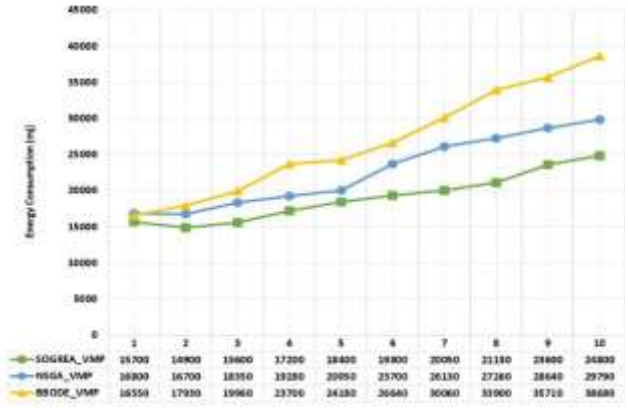
other two algorithms. While both NSGA-III and BBODE algorithms perform well with two objective functions, the iGREA algorithm outperforms them when the number of objective functions increases. It achieves superior results by increasing the number of objectives.

b) Evaluation of the proposed algorithm's migration time at three workloads

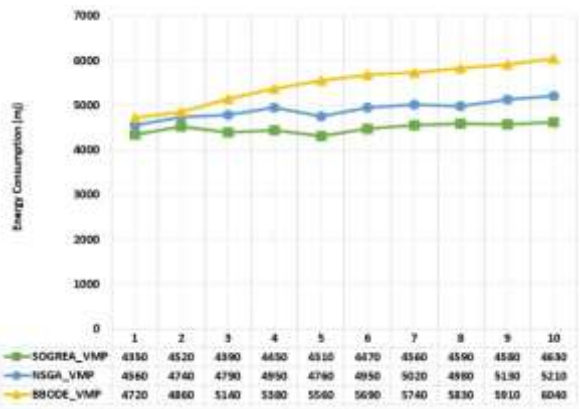
In the second scenario of the experiments, the algorithms are compared based on the migration time. All three algorithms are population-based, trying to find the best solution to improve the objective functions. As shown in Fig. 6, in the first period, the two algorithms NSGA-III and BBODE have the same migration time, while the migration time of the proposed algorithm is less than them. As the number of applications increases, the migration time increases. This is because as the number of requests to access resources on the hosts increases, so does the data transfer time between the two physical machines. Also, increasing the distance between the two hosts is effective in increasing the migration time. The proposed algorithm, with its convenient placement for virtual machines, prevents unnecessary migration and improves system performance and service quality.

c)Evaluation of the proposed algorithm's response time at three workloads

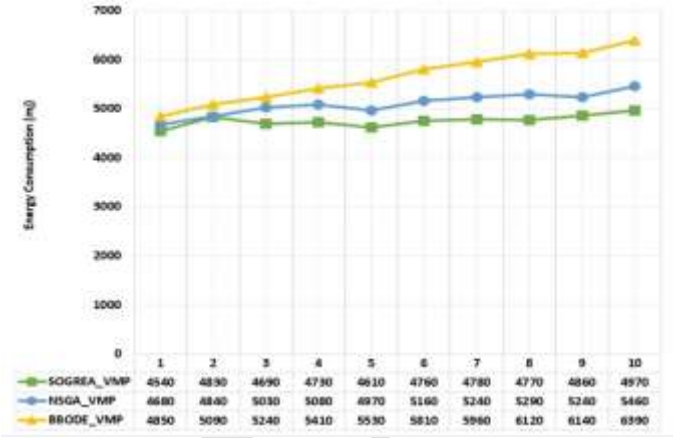
Another important aspect of this article is the response time. Response time is a crucial performance metric when it comes to service quality. It plays a vital role in determining the quality of service. Figure 7 shows a comparison between the three algorithms based on response time. Increasing the waiting time for requests to access resources increases traffic and increases the response time to requests. Optimal decision-making performance and the structure of the proposed algorithm cause the proper distribution of virtual machines in physical machines, which accelerates the speed of responding to requests. The deadline is determined by the service level agreement and the requests are allocated between the virtual machines according to the available resources. Response time increases when virtual machines do not have sufficient resources. The migration of virtual machines leads to an equitable distribution of resources. The results show that with the optimal use of resources, the proposed method reduces the response time compared to NSGA-III and BBODE algorithms.Under high workload conditions (e.g., peak intervals in FIFA and NASA traces), the advantage of iGrEA becomes more pronounced due to its enhanced diversity preservation in many-objective space. While NSGA-III and BBODE suffer from solution crowding near local optima—as typical in >3 objective landscapes—iGrEA's adaptive grid penalty mechanism (Section 3.3.5) maintains a broader exploration front. This enables more balanced trade-offs among energy, migration, and response time when resources are scarce. Specifically, the GR-Adjustment function prevents premature convergence to suboptimal placements that over-consolidate VMs (saving energy but increasing response time), or over-distribute them (reducing latency but raising migration overhead). Thus, iGrEA's structural modifications yield increasingly superior QoS compliance as workload intensity rises.



a) FIFA workload

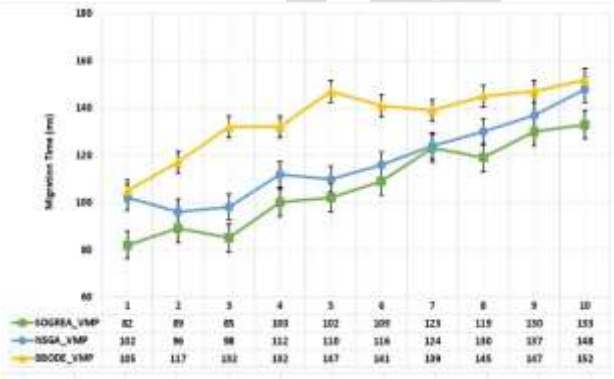


b) NASA workload

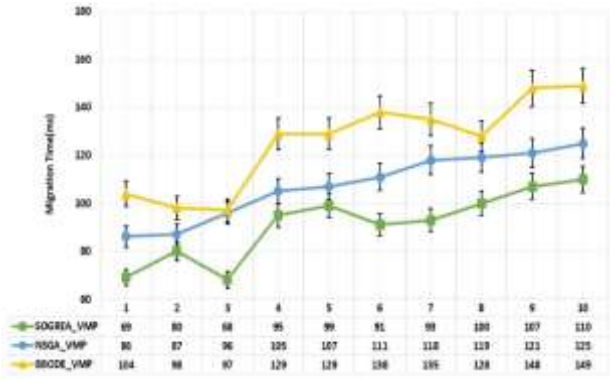


c)ClarkNet workload

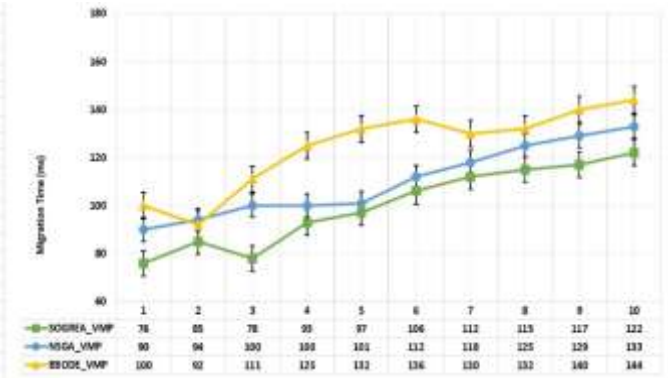
Fig. 5: Energy consumption of the proposed algorithm with different workloads.



a) FIFA workload

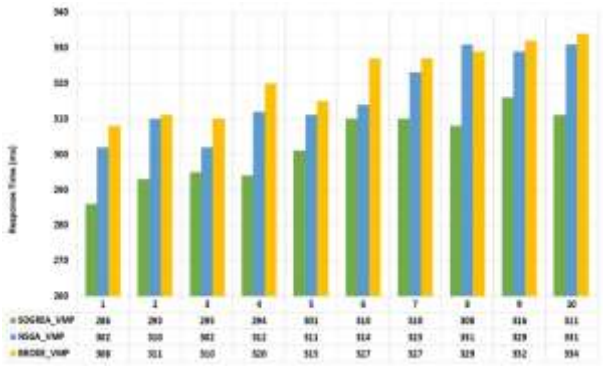


b) NASA workload

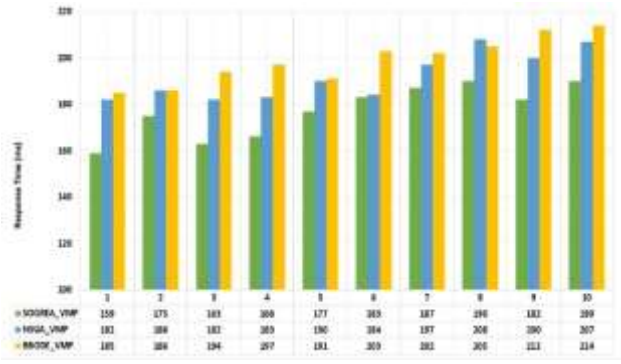


C) ClarkNet workload

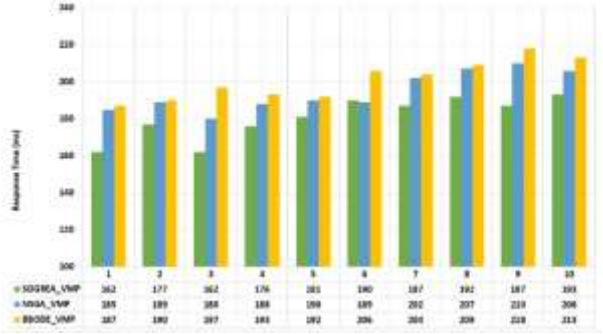
Fig. 6: Average migration time of the proposed algorithm with different workloads.



a) FIFA workload



b) NASA workload



c)Clarknet workload

Fig. 7: Average response time of the proposed algorithm with different workloads.

III. CONCLUSION AND FUTURE WORKS

In recent years, cloud computing is one of the most common computing phenomena for hosting and providing services over the Internet and in demand in the information technology industry. Given the growth of cloud computing infrastructure due to the increasing demand for applications to perform computing in data centers, increasing system performance as well as achieving conflicting goals requires new solutions. Cloud service providers pay a lot of attention to profitability, one of the main challenges of which is to reduce operating costs. However, energy management in cloud data centers is one of the most important issues. There are many ways to reduce energy consumption, and virtual machine placement is one of the most effective methods. To optimal virtual machine placement, there is a need for a many-objective

optimization approach in a timely manner that can achieve a trade-off between meeting the cloud service provider's requirements and the user defined QoS parameters. In this paper, we presented a self-organization framework for virtual machine placement in cloud environment, where key is to orient whether service placement is required or not in each period. In addition, an improved many-objective grid based evolutionary algorithm (iGrEA) is proposed to virtual machine placement, considering the three objective functions. The proposed approach evaluated via simulation and experiments on real traces. Based on the results, it is clear that the proposed approach has better performance than the two compared algorithms and improves energy consumption, migration time and response time. The results also show that the proposed algorithm is effective for solving many-objective problems with a large number of physical and virtual machines in data centers. As a future work, other many-objective evolutionary algorithms can be used simultaneously to locate the virtual machine. Decision trees can be used to increase accuracy in selecting the best placement solutions.

Declarations

- No funds, grants, or other support was received.
- The authors have no conflicts of interest to declare that are relevant to the content of this article.

REFERENCES

[1] Rasoulpour Shabestari, E., & Shameli-Sendi, A. (2025). An Intelligent VM Placement Method for Minimizing Energy Cost and Carbon Emission in Distributed Cloud Data Centers. *Journal of Grid Computing*, *23*(1), 12.

[2] Rawat, P. S., Gaur, S., Barthwal, V., Gupta, P., Ghosh, D., Gupta, D., & Rodrigues, J. J. C. (2025). Efficient virtual machine placement in cloud computing environment using BSO-ANN based hybrid technique. Alexandria Engineering Journal, 110, 145-152.

[3] Adamuthe, A. C., Pandharpatte, R. M., & Thampi, G. T. (2013, November). Multiobjective virtual machine placement in cloud environment. In 2013 international conference on cloud & ubiquitous computing & emerging technologies (pp. 8-13). IEEE.

[4] Shah, M., Rajwar, D., Dehury, J. P., & Kumar, D. (2025). VM Placement in Cloud Computing Using Nature-Inspired Optimization Algorithms. In Nature-Inspired Optimization Algorithms for Cyber-Physical Systems (pp. 251-282). IGI Global Scientific Publishing.

[5] Gao, Y., Guan, H., Qi, Z., Hou, Y., & Liu, L. (2013). A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. Journal of computer and system sciences, 79(8), 1230-1242.

[6] Gupta, M. K., & Amgoth, T. (2018). Resource-aware virtual machine placement algorithm for IaaS cloud. The Journal of Supercomputing, 74, 122-140.

[7] Gu, Z. M., & Wang, G. G. (2020). Improving NSGA-III algorithms with information feedback models for large-scale many-objective optimization. Future Generation Computer Systems, 107, 49-69.

[8] Gabhane, J. P., Pathak, S., & Thakare, N. M. (2021). Metaheuristics algorithms for virtual machine placement in cloud computing environments—a review. Computer Networks, Big Data and IoT: Proceedings of ICCBI 2020, 329-349.

[9] He, M., Xia, H., Chen, H., & Ma, L. (2022). An Inhomogeneous Grid-Based Evolutionary Algorithm for Many-Objective Optimization. IEEE Access, 10, 60459-60473.

[10] Infantia Henry, N., Anbuananth, C., & Kalarani, S. (2022). Hybrid meta-heuristic algorithm for optimal virtual machine placement and migration in cloud computing. Concurrency and Computation: Practice and Experience, 34(28), e7353.

[11] Khallouli, W., & Huang, J. (2022). Cluster resource scheduling in cloud computing: literature review and research challenges. The Journal of supercomputing, 1-46.

[12] Khaleel, M. I. (2023). Efficient job scheduling paradigm based on hybrid sparrow search algorithm and differential evolution optimization for heterogeneous cloud computing platforms. Internet of Things, 100697.

[13] Khodayarseresht, E., & Shameli-Sendi, A. (2023). A multi-objective cloud energy optimizer algorithm for federated environments. Journal of Parallel and Distributed Computing, 174, 81-99.

[14] Li, L., Gao, J., & Mu, R. (2019). Optimal data file allocation for all-to-all comparison in distributed system: A case study on genetic sequence comparison. INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL, 14(2), 199-211.

[15] López-Pires, F., & Barán, B. (2017). Many-objective virtual machine placement. Journal of Grid Computing, 15(2), 161-176.

[16] Mejahed, S., & Elshrkawey, M. (2022). A multi-objective algorithm for virtual machine placement in cloud environments using a hybrid of particle swarm optimization and flower pollination optimization. PeerJ Computer Science, 8, e834.

[17] Nabavi, S., Wen, L., Gill, S. S., & Xu, M. (2023). Seagull optimization algorithm based multi-objective VM placement in edge-cloud data centers. Internet of Things and Cyber-Physical Systems, 3, 28-36.

[18] Tang, X., Shi, C., Deng, T., Wu, Z., & Yang, L. (2021). Parallel random matrix particle swarm optimization scheduling algorithms with budget constraints on cloud computing systems. Applied Soft Computing, 113, 107914.

[19] Wang, X., Lou, H., Dong, Z., Yu, C., & Lu, R. (2023). Decomposition-based multi-objective evolutionary algorithm for virtual machine and task joint scheduling of cloud computing in data space. Swarm and Evolutionary Computation, 101230.

[20] Zolfaghari, R., Sahafi, A., Rahmani, A. M., & Rezaei, R. (2022). An energy-aware virtual machines consolidation method for cloud computing: Simulation and verification. Software: Practice and Experience, 52(1), 194-235.