

# Energy-Aware Virtual Machine Placement in Cloud Data Centers Using an Enhanced Multi-Objective Grey Wolf Optimizer

Marwa K. Farhan<sup>1</sup> , Haider Abdulkarim<sup>2\*</sup> 

<sup>1</sup>Scholarships and Culture Relations Directorate, Ministry of Higher Education and Scientific Research, Baghdad 10001, Iraq

<sup>2</sup>College of Communications Engineering, University of Technology, Baghdad 10066, Iraq

<sup>1</sup> marwa.farhan@mohesr.edu.iq, <sup>2</sup> haider.a.abdulkarim@uotechnology.edu.iq

## Abstract

Server consolidation and power consumption in data centers become more complex as server loads increase and virtual machine placement (VMP) becomes a key issue. This paper proposes a static VMP method for heterogeneous cloud data centers using the Enhanced Energy-Aware Multi-Objective Grey Wolf Optimizer (EA-MOGWO). It is implemented through energy-aware seeding, alpha-beta-delta-based discrete host representation with leader guidance, adaptive leader inheritance and mutation management, feasibility repair, fixed-reference objective-based normalization, and bounded local consolidation. The fitness function considers normalized server power, the number of active hosts, the migration ratio, and CPU-RAM imbalance, with weights of 0.40, 0.25, 0.20, and 0.15, respectively. We conducted experiments on a custom-made Python simulator with 50 heterogeneous hosts, 100, 200, and 300 VMs, a population size of 15, and 10 independent paired runs with 30 iterations. In all scenarios, EA-MOGWO had the lowest mean weighted fitness. The mean fitness decreased by 22.0%, 11.9%, and 3.6% relative to the PSO baseline, and by 28.8%, 25.8%, and 22.1% relative to the GWO baseline, respectively. The relative power reduction was greatest at 100 VMs and gradually closed as the workload density increased. The results mainly reflect better consolidation from increased migration and CPU-RAM imbalance in lighter workloads. The findings confirm EA-MOGWO as a static VM-placement approach for consolidation purposes in the considered CPU-RAM model and encourage its extension to larger workloads, different SLA parameters, and production traces.

**Keywords:** Cloud Computing, Virtual Machine Placement, Energy Efficiency, Grey Wolf Optimizer, Particle Swarm Optimization, Metaheuristic Optimization, Resource Allocation.

**Article history:** Received: 17 Aug 2026, Accepted: 8 Sep, Published: 15 Sep 2026.

This article is open-access under the CC BY 4.0 license (<http://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Cloud computing is changing how computing resources are provided, allowing users to access scalable infrastructure, platforms, and software services via the Internet. Thanks to the popularity of virtualization technologies, several Virtual Machines (VMs) can share the same physical hardware, maximizing hardware usage, lowering deployment expenses, and providing greater service flexibility [1] and [2]. As a result, cloud computing has emerged as the new paradigm for scientific

computing, enterprise applications, big data analysis, and artificial intelligence. However, these benefits have come with serious energy challenges due to the rapid growth of cloud data centers. Today's hyperscale data centers need vast amounts of electrical power not only for computing hardware but also to cool the machines and other infrastructure.

Therefore, energy use is a key operational expenditure for cloud service providers and raises

environmental concerns about carbon emissions [3],[4]. As a result, energy efficiency has emerged as an important research area in both industry and academia. Virtual Machine Placement (VMP) is one of the most prominent problems in cloud resource management, directly influencing resource utilization, server consolidation, migration overhead, service quality, and energy consumption. Efficient VM placement aims to place virtual machines on physical hosts that satisfy resource constraints while minimizing multiple conflicting objectives. The placement problem is a class of NP-hard combinatorial optimization problems, making it difficult to find the optimal solution in large cloud infrastructures [5]. In response, researchers have proposed various heuristic and metaheuristic optimization algorithms over the past decade. Classical heuristics like First Fit Decreasing (FFD) are fast and make placement choices. However, they are greedy and likely to get stuck in a locally optimal solution.

Recently, various nature-inspired and swarm intelligence optimization algorithms, such as Particle Swarm Optimization (PSO) [6], Ant Colony Optimization (ACO) [7], Whale Optimization Algorithm (WOA) [8], Harris Hawks Optimization (HHO) [5], the Horse Herd Algorithm [9], and Grey Wolf Optimizer (GWO) [10], have been proven effective and efficient in solving complex discrete combinatorial problems by maintaining a balanced exploration-exploitation mechanism with manageable computational overhead.

In this regard, the Grey Wolf Optimizer has gained significant attention because of its minimal control parameters, low mathematical complexity, and strong optimization performance. However, when solving highly constrained optimization problems, such as VM placement in GWO, the original GWO was found to converge prematurely, reducing population diversity. These constraints become even more pronounced when users run a high density of virtual machines in cloud computing environments, where the search space becomes very small and multiple VMs compete for limited physical resources. Recent studies have tried to enhance GWO optimization by adapting parameters, using hybrid optimization methods, and applying

local search methods. While these methods have achieved beneficial results, most studies optimize limited goals and/or test their techniques under constrained conditions. Neglecting multidimensional resource balance often leads to single-resource exhaustion (e.g., RAM saturation while CPU remains underutilized), preventing additional VM allocations on active servers. Minimizing CPU-RAM imbalance ensures proportional resource utilization and mitigates severe bottlenecks on consolidated hosts. Reproducibility also suffers if workload generation, discrete encoding, feasibility handling, Normalization, and algorithm-specific operators are not described adequately.

To address these drawbacks, this paper introduces a new VM placement approach for heterogeneous cloud data centers based on the Enhanced Energy-Aware Multi-Objective Grey Wolf Optimizer (EA-MOGWO). This is achieved through energy-aware population initialization, adaptive leader inheritance and mutation, a feasibility repair mechanism, stable normalized fitness evaluation, and bounded consolidation. The mechanisms steer towards more feasible consolidated placements while maintaining some stochastic exploration.

We set up a Python simulator in the cloud to test the proposed algorithm and compare it with Particle Swarm Optimization (PSO) and a GWO baseline across three heterogeneous workload scenarios. This comparison is based on overall fitness, modeled server power, active physical hosts, convergence behavior, migration, and resource-imbalance trade-offs. However, this work has the following contributions:

1. We propose a novel Enhanced Energy-Aware Multi-Objective Grey Wolf Optimizer (EA-MOGWO) to improve virtual machine placement in heterogeneous cloud data centers.
2. Based on this, we formulate a constrained multi-objective mathematical model under a static CPU-RAM allocation framework to minimize modeled server power, active physical servers, migration ratio, and load imbalance under strict CPU and memory bounds.

3. We propose a feasibility-preserving repair operation, adaptive leader inheritance, and controlled mutations to improve solution quality without compromising worst-case asymptotic order.
4. We compare experimentally against a discrete GWO baseline on paired problem instances and report results in terms of weighted fitness, server power model, number of active hosts, convergence, runtime, migration ratio, and CPU-RAM imbalance.

This paper is organized as follows: Section 2 summarizes the recent studies on energy-aware virtual machine placement. Section 3 mathematically formulates the optimization problem. Section 4 presents the proposed EA-MOGWO algorithm. Section 5 describes the experimental methodology. The results obtained from the experiment are discussed in Section 6, and the future research directions are summarized in Section 7.

## 2. Related Work

Virtual Machine Placement (VMP) has been a focus of substantial research over the last decade because it directly influences resource utilization, energy consumption, operational costs, and service quality in cloud data centers [11]. The VMP problem is NP-hard, and finding an exact optimal solution becomes costly for large infrastructures. Therefore, researchers have focused on heuristic and metaheuristic optimization methods that can generate near-optimal solutions within reasonable computation time [2], [6]. Recent work can be broadly sorted into four main categories: heuristic-based approaches, swarm intelligence algorithms, evolutionary optimization algorithms, and hybrid metaheuristic methods.

### 2.1 Heuristic-Based VM Placement

Traditional, widely used heuristic algorithms include First Fit (FF), Best Fit (BF), First Fit Decreasing (FFD), and Best Fit Decreasing (BFD), which are simple and computationally inexpensive. These strategies aim to consolidate virtual machines onto fewer physical machines to reduce energy use. Heuristic methods are fast, but they are greedy and often converge to suboptimal placements, especially in heterogeneous cloud environments with

significantly different resource requests [5]. Researchers have proposed incorporating resource-awareness and workload-balancing strategies into FFD to improve it. However, these approaches are still limited in their ability to perform a global search and do not always deliver good results for large enterprise cloud infrastructure.

### 2.2 Swarm Intelligence Algorithm

Many researchers have found swarm intelligence to be one of the most successful methods for solving VM placement problems because it balances exploration and exploitation. A widely used baseline for population-based optimization is Particle Swarm Optimization (PSO), proposed by Kennedy and Eberhart [7], which features a simple update and early search. Meanwhile, self-adaptive PSO variants have been further tailored for cloud server consolidation [13]. However, these search updates may produce invalid host assignments in constrained VMP, where continuous particle positions must be mapped and handled according to feasibility.

Mirjalili et al. proposed the Grey Wolf Optimizer (GWO) [6] based on alpha, beta, and delta leaders. A recent VMP study based on GWO proposed an adaptive-boundary strategy. It demonstrated improved solution quality and stability for VM packing traces [5]. Researchers have also extended other GWOs for energy sustainability and VM-migration decisions [14]. However, the standard GWO has a few disadvantages for discrete VM placement problems [15]. Continuous position updates can create feasibility issues by placing infeasible positions that fail to meet resource constraints, and the linear convergence parameter can reduce exploration over time, increasing the risk of early convergence in dense cloud workloads.

### 2.3 Evolutionary Optimization Algorithms

Another field where evolutionary algorithms have been extensively studied is that of VM placement for energy-aware scheduling. Genetic Algorithms (GA) are relevant because assignment vectors can be represented directly and diversified through crossover and mutation operations. The adaptive thresholds were combined with VM placement to minimize energy consumption and SLA violations in a 2024 GA-based study. GA is flexible in terms of

discrete encoding, and its results may be sensitive to population size, crossover, mutation, and stopping rules. Other bio-inspired techniques such as ant-colony and hybrid metaheuristics have been applied to VMP and consolidation [4],[16]. Usefulness depends not only on final objective values but also on feasibility preservation, parameter burden, and the computational cost of the auxiliary local-search or prediction part.

## 2.4 Hybrid Metaheuristic Approaches

In recent years, hybrid optimization algorithms, based on a combination of complementary search methods, have become increasingly popular [17]. Other recent works on cloud sustainability focus on migration control, workload uncertainty, and renewable-energy-aware decisions [14], [18], [19], [20]. This broader body of work indicates that no single placement metric fits all: aggressive consolidation can reduce server power by pushing migration and concentration. Therefore, multi-objective placement is a compromise problem. Applying weighted aggregation yields a reproducible scalar objective. However, it depends on the weights and normalization references used. Thus, balancing optimization quality with the algorithm's computational simplicity remains an important research problem. Table 1 summarizes a comparison of Recent Energy-Aware VM Placement Approaches.

While standard Multi-Objective Grey Wolf Optimizers (MOGWO) [21] typically rely on Pareto-archive non-dominated sorting or equal objective weighting, prior VMP studies [22] rarely evaluate host consolidation alongside explicit migration overhead and multidimensional resource imbalance simultaneously under fixed scalar Normalization.

## 2.5 Research Gap

Existing GWO-based VMP algorithms [5],[22] predominantly rely on continuous position rounding

or simple boundary constraints (Existing methods). However, these mechanisms frequently generate resource-infeasible mappings in highly dense heterogeneous clouds (Limitation), leaving the problem of simultaneously optimizing power, migration overhead, and multidimensional resource imbalance unresolved (Unresolved problem). Addressing this requires an integrated framework combining discrete leader guidance, explicit CPU/RAM feasibility repair, and bounded consolidation (Methodological requirement). EA-MOGWO addresses this gap by combining energy-aware seeding, adaptive leader inheritance/mutation, and deterministic Repair into a single scalarized optimizer (Proposed solution).

## 2.6 Motivation of This Work

Given these drawbacks, this paper introduces an Enhanced Energy-Aware Multi-Objective Grey Wolf Optimizer (EA-MOGWO) for virtual machine placement in heterogeneous cloud data centers. The proposed algorithm combines four complementary enhancement mechanisms, which are different from the current methods:

- Energy-aware population initialization.
- Inheritance and mutation control that adapts.
- Feasibility-preserving repair operator.
- Stable normalized multi-objective fitness evaluation.

Together, these mechanisms maintain feasibility, preserve stochastic diversity, and gradually steer the search toward leader-guided exploitation while keeping the implementation small. In addition, we test EA-MOGWO with paired problem instances against PSO and GWO baselines using the same fitness function, feasibility rules, population size, and iteration budget. The analysis covers the overall objective and the primary trade-offs, but makes no inferences about the 10-run experiment.

**Table 1:** Comparison of Recent Energy-Aware VM Placement Approaches

| Ref.                | Approach                 | Primary focus                  | Validation context          | Relevant limitation                                                |
|---------------------|--------------------------|--------------------------------|-----------------------------|--------------------------------------------------------------------|
| Alourani et al. [4] | GA + adaptive thresholds | Energy and SLA violations      | Simulation across VM scales | GA/threshold parameter sensitivity                                 |
| Feng et al. [5]     | Adaptive-boundary GWO    | Minimize physical servers      | Azure VM packing traces     | Objective centers on server count; different model from this study |
| Chen et al. [23]    | Combined-trend VMC       | Energy/SLA-aware consolidation | Workload trend prediction   | Requires temporal workload information                             |

| Ref.                | Approach                       | Primary focus                                | Validation context             | Relevant limitation                                       |
|---------------------|--------------------------------|----------------------------------------------|--------------------------------|-----------------------------------------------------------|
| Han et al. [24]     | $\Gamma$ -robust consolidation | Energy/resource allocation under uncertainty | Huawei Cloud data              | Different robust-optimization formulation                 |
| Seddiki et al. [14] | Knowledge-acquisition GWO      | Energy-sustainability/VM migration           | Custom cloud simulation        | Targets renewable-energy decisions rather than static VMP |
| This work           | EA-MOGWO                       | Power, active hosts, migration, imbalance    | Custom paired static workloads | Static CPU-RAM model; no SLA/network/thermal dynamics     |

### 3. Problem Formulation and Mathematical Model

#### 3.1 Problem Definition

Although designated as EA-MOGWO, the formulation utilizes a fixed weighted scalarization function rather than a Pareto-archive non-dominated sorting mechanism. This scalarized multi-criteria approach guarantees deterministic ranking and low runtime complexity  $O(T.P.N.M)$  while accounting for four conflicting operational metrics.

Virtual Machine Placement (VMP) distributes a group of virtual machines onto various physical hosts with different resource capacities while meeting resource-capacity requirements and reducing modeled server power, active host count, migration overhead, and resource imbalance. Suppose the following sets denote the sets of physical hosts and virtual machines:

$$\mathcal{H} = \{H_1, H_2, \dots, H_M\} \quad (1)$$

$$\mathcal{V} = \{V_1, V_2, \dots, V_N\} \quad (2)$$

Each host has limited CPU and RAM resources, and each VM requires a specific amount of CPU and RAM. The mathematical formulation is limited to these two resources, since they are the resource dimensions explicitly implemented in the simulator used in the experiments reported here.

#### 3.2 Decision Variables and Assignment

A candidate placement is a vector (discrete vector), where each component specifies a physical host to which one VM was placed:

$$\mathbf{x} = (x_1, x_2, \dots, x_N) \quad (3)$$

where  $x_i$  belongs to  $\{1, \dots, M\}$ . The equivalent binary assignment variable can be defined for the constrained formulation as follows:

$$z_{ij} = \begin{cases} 1 & , \text{if } x_i = j \\ 0 & , \text{otherwise} \end{cases} \quad (4)$$

Each VM needs to be mapped to exactly one physical host:

$$\sum_{j=1}^M z_{ij} = 1, \quad \forall i \quad (5)$$

#### 3.3 Resource-Capacity Constraints

The total CPU and RAM requirements of VMs assigned to each host must remain within that host's effective capacity. The operational headroom for the experiments was limited to 90% nominal CPU capacity and 95% nominal RAM capacity.

$$\sum_{i=1}^N z_{ij} d_i^{cpu} \leq C_j^{cpu}, \quad \forall j \quad (6)$$

$$\sum_{i=1}^N z_{ij} d_i^{ram} \leq C_j^{ram}, \quad \forall j \quad (7)$$

#### 3.4 Host Utilization

The CPU and RAM utilization ratios of host  $j$  are computed as follows based on the VM demands assigned to the host:

$$u_j^{cpu} = \frac{\sum_{i=1}^N z_{ij} d_i^{cpu}}{C_j^{cpu}} \quad (8)$$

$$u_j^{ram} = \frac{\sum_{i=1}^N z_{ij} d_i^{ram}}{C_j^{ram}} \quad (9)$$

Both utilization ratios are in the interval  $[0,1]$  for a feasible placement. The scalar utilization used for multidimensional reporting is the average of the CPU utilization and RAM utilization:

$$u_j = \frac{u_j^{cpu} + u_j^{ram}}{2} \quad (10)$$

#### 3.5 Active-Host Model

A host is active as long as at least one VM is allocated to it. It has a Boolean state of activation, and the number of active hosts is defined:

$$y_j = \begin{cases} 1, & \text{if } \sum_{i=1}^N z_{ij} > 0 \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

$$H_{active} = \sum_{j=1}^M y_j \quad (12)$$

### 3.6 Server-Power Model

The simulator adopts the most common model used in energy-aware consolidation: utilization-based linear server-power model [3] and [8]. The number reported in the experiments is instantaneous IT power in watts, not time-integrated energy. The simulator considers an inactive host offline at the placement level. Thus, the power consumption of host  $j$  is:

$$P_j = y_j(P_j^{idle} + (P_j^{max} - P_j^{idle})u_j^{cpu}) \quad (13)$$

The total instantaneous IT server power in Watts ( $W$ ) is:

$$P_{total} = \sum_{j=1}^M P_j \quad (14)$$

### 3.7 Resource-Imbalance Measure

The imbalance measure measures the difference between the percentage of CPU and RAM usage on live hosts. The lower the number, the more proportional the use of the two resource dimensions:

$$B = \frac{\sum_{j=1}^M y_j |u_j^{cpu} - u_j^{ram}|}{\max(1, H_{active})} \quad (15)$$

### 3.8 Migration Ratio

Let  $X^0$  be the first feasible placement. A VM is considered "migrated" if the final host differs from the initial host. The indicator of migration and the normalized migration ratio are:

$$m_i = \begin{cases} 1, & \text{if } x_i \neq x_i^0 \\ 0, & \text{otherwise} \end{cases} \quad (16)$$

$$R_{mig} = \frac{\sum_{i=1}^N m_i}{N} \quad (17)$$

### 3.9 Stable Objective Normalization

The objectives are measured in different units and do not fall within the same range. This is done by using fixed reference denominators, so that the same placement always gets the same normalized score, regardless of the current optimizer population:

$$(\bar{P}_{norm}) = \min(1, P_{total}/P_{ref}) \quad (18)$$

$$\bar{H} = \frac{H_{active}}{M} \quad (19)$$

$$\bar{M} = R_{mig} \quad (20)$$

$$\bar{B} = B \quad (21)$$

### 3.10 Multi-Objective Fitness Function

These four normalized criteria are summed using a weighting system. The highest weightage is given to Normalized server power, followed by Active-host consolidation, Migration preservation, and CPU-RAM balance:

$$F(X) = w_1 \bar{P}_{norm} + w_2 \bar{H} + w_3 \bar{M} + w_4 \bar{B} \quad (22)$$

The weight allocation ( $w_1=0.40$ ,  $w_2=0.25$ ,  $w_3=0.20$ ,  $w_4=0.15$ ) reflects practical operational priorities in cloud data centers: direct power reduction and server consolidation carry the highest cost impact, while migration penalties and resource balancing act as secondary regularization terms. Sensitivity analysis indicates that increasing  $w_3$  (migration) above 0.35 suppresses aggressive consolidation. In contrast, higher  $w_1$  prioritizes host shutdowns at the expense of temporary migration overhead. The weights are experimental design parameters and add up to one. Lower fitness values indicate better overall placement.

### 3.11 Feasibility Repair and Final Optimization Problem

The swarm updates can create infeasible host assignments. The repair operator runs before evaluation; it moves VMs from overloaded hosts to potential locations that meet CPU and RAM requirements. The evaluated objective and final optimization problem are:

$$F_{eval}(X) = F(\mathcal{R}(X)) \quad (23)$$

$$X^* = \arg \min_X F_{eval}(X) \quad (24)$$

The unique-assignment constraint in Equation (5), the RAM constraint in Equation (7), and the CPU constraint in Equation (6) are not present in the optimization in Equation (24), which is accompanied by binary assignment variables. This results in a multi-objective, discrete VM-placement problem subject to constraints that can be evaluated using PSO, GWO, and the proposed EA-MOGWO.

### 3.12 Equation Provenance

Equations (4) and (7) are based on standard binary and multidimensional capacity-constrained allocation (binary placement) and capacity-constrained placement formulations, as used in

VMP research [5]. Equations (13)- (14) use the well-known linear server-power model [3] [8] for utilization. This study makes the following modeling choices: the combined utilization in Equation (10), the imbalance measure in Equation (15), fixed-reference Normalization in Equations (18-21), the weighted objective in Equation (22), and the repair-based evaluation in Equation (23).

#### 4. Proposed EA-MOGWO Methodology

##### 4.1 Overview

EA-MOGWO is a variant of GWO for constrained VM placement, which is discrete and leader-guided. It keeps the alpha-beta-delta hierarchy with the proviso that the continuous GWO position equations are not directly applied to host identifiers. Instead, a placement component is inherited by a leader with a time-varying probability, along with mutation, feasibility repair, and bounded consolidation. It implements four main mechanisms: energy-aware seeding, adaptive leader-guided exploration control, feasibility repair, and fixed-reference multi-objective evaluation, and includes a bounded local consolidation stage. The placement encoding, capacity limits, repair policy, and fitness function are identical for PSO, GWO, and EA-MOGWO.

##### 4.2 Standard Grey Wolf Optimizer

The original continuous GWO formulation relies on distance updates across continuous position spaces (Equations 25–33). To avoid structural mapping ambiguity, we present these continuous equations purely as mathematical background. In our discrete implementation, we omit continuous arithmetic on integer host IDs; search agents update assignments via direct discrete leader-guided inheritance and probability operators (Section 4.5).

$$A = 2a r_1 - a \quad (25)$$

$$C = 2r_2 \quad (26)$$

$$D_\alpha = \text{norm}(C_1 X_\alpha - X) \quad (27)$$

$$D_\beta = \text{norm}(C_2 X_\beta - X) \quad (28)$$

$$D_\delta = \text{norm}(C_3 X_\delta - X) \quad (29)$$

$$X_1 = X_\alpha - A_1 D_\alpha \quad (30)$$

$$X_2 = X_\beta - A_2 D_\beta \quad (31)$$

$$X_3 = X_\delta - A_3 D_\delta \quad (32)$$

$$X(t+1) = (X_1 + X_2 + X_3) / 3 \quad (33)$$

In this case,  $r_1$  and  $r_2$  are random vectors in  $[0,1]$ , and  $a$  is the canonical GWO exploration-

control parameter. Equations (25-33) are provided as background. However, they are not directly used in the discrete implementation, as they are taken from Mirjalili et al. [6].

##### 4.3 Solution Encoding

$N$  VMs are assigned to  $M$  hosts in each search agent. An integer host identifier is called a component  $x_i$ . When the child is created during population generation, it is created from a viable parent, and for each VM it can have the same VM hosting as alpha, beta, or delta. Random host identifiers replace the mutated components, and the repaired vector is then evaluated:

$$X = (x_1, x_2, \dots, x_N), \quad x_i \in \{1, 2, \dots, M\} \quad (34)$$

Where  $x_i$  gives the physical host to which VM <sub>$i$</sub>  is mapped; no continuous-to-discrete rounding step occurs in the GWO baseline or EA-MOGWO search operator because all the host identifiers are discrete. The repair procedure after leader inheritance and mutation ensures validity.

##### 4.4 Energy-Aware Population Initialization

EA-MOGWO initializes the energy-aware FFD-like placement, the common initial feasible placement, and a consolidated version of the energy-aware Seed as structure candidates. The remaining population is created as random feasible placements. This is done deliberately to boost EA-MOGWO in the initial search region. Interpret the convergence plots not as evidence of faster convergence in the iterations themselves, but as the combined effect of the initial search region and the iterative search.

##### 4.5 Adaptive Leader-Guided Search Control

The inheritance probability of a VM assignment from one of the three leaders gradually moves toward 0.85, meaning that exploitation is increasingly strengthened as the iteration goes on:

$$p_{\text{leader}}(t) = 0.45 + 0.40 \frac{t}{T-1} \quad (35)$$

EA-MOGWO simultaneously decreases its mutation probability over the run:

$$mu_{EA}(t) = 0.08 \left(1 - \frac{t}{T}\right) + 0.01 \quad (36)$$

$$mu_{GWO} = 0.04 \quad (37)$$

The GWO baseline has the same leader-inheritance probability, but a fixed mutation rate of 0.04. The differences between EA-MOGWO and

other algorithms lie in the adaptive mutation, structured initialization, and periodic consolidation. These schedules are specific to each implementation and are not to be confused with  $a$  parameter of the canonical GWO.

#### 4.6 Feasibility Repair Operator

Assignments can exceed the operational CPU and RAM limits due to leader inheritance, random mutation, and/or PSO rounding. Each candidate is then repaired by reducing the demand of the most dominant resource and moving the candidate VM to a feasible host based on incremental power and utilization cost. If no such host exists, use a feasible construction with the FFD-like approach as a backup.

##### Host and VM Selection Rules for Repair:

1. Overloaded Host Selection: Select host  $j$  with  $\max(\mu_j^{\text{cpu}}, \mu_j^{\text{ram}}) > 1.0$  (or exceeding 0.90/0.95 threshold limits).
2. VM Selection: Pick the VM  $i$  on host  $j$  with the largest ratio  $d_i^{\text{cpu}}/C_j^{\text{cpu}} + d_i^{\text{ram}}/C_j^{\text{ram}}$ .
3. Candidate Host Ranking: Rank candidate target hosts  $k \neq j$  in ascending order of predicted power  $\Delta P_k$ .
4. Tie-Breaking: If two hosts yield identical  $\Delta P_k$ , select the host with the lower active VM count  $y_k$ ; if still tied, select the smaller host index  $k$ .
5. Failure Fallback: If no candidate host satisfies capacity limits, allocate VM  $i$  using a deterministic First Fit Decreasing (FFD) scan across all active hosts.

#### 4.7 Multi-Objective Fitness Evaluation

All repaired placements are assessed based on the fixed-reference objective in Section 3. The scalar fitness is a combination of normalized server power, count of active hosts, migration ratio, and CPU-RAM imbalance:

$$F(x) = 0.40 P_{\text{norm}}(x) + 0.25 H_{\text{norm}}(x) + 0.20 M_{\text{norm}}(x) + 0.15 B(x) \quad (38)$$

A smaller fitness value indicates that the four objectives are being balanced. Fixed normalization denominators ensure that the objective value for the same placement is the same for both the optimizer population and each iteration.

#### 4.8 Lightweight Local Improvement

The algorithm makes at most two consolidation attempts every fifth EA-MOGWO iteration. The least-loaded active host is selected for evacuation; other VMs are rebalanced to the feasible hosts with the same cost-based best-fit criterion. Keep the consolidated candidate only if its final weighted fitness is still worse than the unrevised offspring. This bounded stage reinforces exploitation but can increase migration more than the placement stage.

#### 4.9 Proposed Algorithm

Algorithm 1 summarized the entire procedure.

**Algorithm 1.** Proposed EA-MOGWO for energy-aware VM placement.

1. Enter the hosts, VMs, population size  $P$ , maximum number of iterations  $T$ , and a common initial feasible placement. Enter hosts, VMs, population size  $P$ , max. number of iterations  $T$ , and a common initial feasible placement.
2. Initialize EA-MOGWO with an energy-aware seed, the common initial placement, a consolidated seed, and randomized feasible placements.
3. Identify all candidates with the same normalized fitness and determine alpha, beta, and delta.
4. For every iteration  $t$ , calculate the value of  $p_{\text{leader}}(t)$  and the value of the EA-MOGWO mutation probability; the GWO baseline value is a fixed 0.04 for mutation.
5. Create each offspring from a feasible parent, and inherit some VM assignments from alpha, beta, or delta depending on  $p_{\text{leader}}(t)$ .
6. If VM host identifiers have been chosen, mutate them and repair CPU/RAM-infeasible assignments before evaluation.
7. After each consolidation attempt (at most two attempts every 7 iterations of EA-MOGWO), save applicants if their weighted fitness improves.
8. Update alpha, beta, and delta and record the best fitness for convergence analysis.
9. Repeat until  $T$  iterations are completed.
10. Return the optimum possible position of alpha.

#### 4.10 Computational Complexity

Let  $P$  be the population size,  $T$  be the number of iterations,  $N$  be the number of VMs, and  $M$  be the number of physical hosts. In the worst case, placement decoding/repair can review candidate

hosts for each VM. The predominant computational expense is the cost of:

$$O(T.P.N.M) \quad (39)$$

The time complexity of calculating the leader probability and the mutation schedule is constant. The bounded consolidation stage applies, but it does not affect the worst-case order, which is dominated by population evaluation and Repair. The results of the runtimes are therefore reported descriptively and not just by the asymptotic notation.

#### 4.11 Methodological Advantages

- Seeding is performed to maximize the quality of the initial search population, due to energy awareness.
- The exploration-exploitation balance in the discrete search is changed by adaptive leader inheritance and mutation.
- The repair operator excludes infeasible placements from being compared as solutions.
- Fixed Normalization - stabilizes and equates fitness values between runs.
- Lightweight local consolidation that avoids complex algorithmic structure but provides good exploitation.

### 5. Experimental Setup

#### 5.1 Experimental Environment

The algorithms were tested on a custom simulator written in Python 3.12, installed on a Windows 11 computer with 16 GB of RAM and an Intel i7 processor. The simulator represents a static, heterogeneous data center with limited CPU and RAM resources, and it uses a server-power model based on utilization. It is not an algorithmic experiment nor an evaluation of CloudSim or a production cloud. We chose a custom Python simulator over standard frameworks like CloudSim to focus explicitly on static mathematical vector placement optimization.

This lightweight architecture eliminates event-queue execution overhead, allows precise control over static multidimensional capacity bounds, and guarantees identical initial state reproducibility across independent paired optimizer runs. For each paired run, the host configuration generated for each system, the VM workload, and the initial feasible placement were the same. We generated different

random generators, each specific to its algorithm, from the common run seed to prevent any common stochastic stream from being used across optimizers.

#### 5.2 Cloud Configuration

We used 50 heterogeneous hosts in each experiment. Host CPU capacities were sampled from 16,000, 24,000, and 32,000 units; RAM capacities from 32,768, 65,536, and 98,304 units; idle power from 175, 210, and 250 W; and maximum power from 250, 320, and 400 W. VM templates ranged from 500 CPU/1,024 MB RAM to 4,000 CPU/12,288 MB RAM and were perturbed by  $\pm 10\%$ . The operational limits were at 90% of the nominal CPU and 95% of the nominal RAM. Table 2 shows the proposed scenarios.

**Table 2:** Experimental Workload Scenarios

| Scenario | Number of VMs | Number of Hosts |
|----------|---------------|-----------------|
| S1       | 100           | 50              |
| S2       | 200           | 50              |
| S3       | 300           | 50              |

The number of workloads was kept constant for each workload size, and increasing the number of hosts progressively reduced spare capacity and made the 300-VM scenario more constrained. This design is not a "scale out" design to scale up for more workload; it is a "workload density" comparison.

#### 5.3 Algorithm Parameters

The settings of all the parameters used in all the experiments are summarized in Table 3. Where applicable, we kept the parameters the same across all optimization algorithms to ensure an unbiased comparison.

**Table 3:** Parameter Settings

| Parameter               | Value    |
|-------------------------|----------|
| Physical hosts          | 50       |
| Population size         | 15       |
| Maximum iterations      | 30       |
| Independent paired runs | 10       |
| Base seed               | 20260731 |

The iteration budget was fixed at  $T = 30$  iterations based on empirical pilot runs showing that population fitness stagnates beyond 25–30 iterations for static 50-host instances, balancing computational overhead and optimization performance. All of the algorithms were tested with the same number of objects and iterations. PSO had an inertia weight

that decreased between 0.9 and 0.4, and cognitive and social coefficients of 1.7, with velocity clipped to  $[-6,6]$ . For the GWO baseline and EA-MOGWO, the discrete leader guided operator described in Section 4 was adopted.

#### 5.4 Evaluation Metrics

The performance was measured by mean fitness, server power (aggregated), number of active hosts, convergence history, migration ratio, CPU-RAM imbalance, feasibility rate, and runtime. The main reported metrics are fitness, power, and active-host count, and the reported secondary metrics are migration and imbalance.

##### 5.4.1 Aggregate Server Power (W)

The linear model of utilization presented in Section 3 was used to determine the aggregate server power in watts. Since no time integration occurs, this is a power rather than a power measurement. Low power refers to a less power-intensive placement for the simulator assumptions.

##### 5.4.2 Number of Active Physical Hosts

Number of active servers: This indicates how well each optimization algorithm consolidates. The lower the value, the more efficient the use of physical resources.

##### 5.4.3 Overall Fitness Value

This normalized weighted fitness includes server power, active hosts, migration ratio, and CPU-RAM imbalance. Lower fitness does not mean that each of the component objectives is individually minimized, but rather represents a better weighted compromise according to the fixed weights 0.40/0.25/0.20/0.15.

##### 5.4.4 Convergence Performance

The best fitness at each iteration was recorded. The Convergence Curves of EA-MOGWO show the results of both the initialization quality and the search behavior, as this is not the same as the baselines, which do not use structured initialization.

#### 5.5 Repetition and Descriptive Analysis

The base seed used was 20260731; thus, 10 independent paired runs were performed for each workload scenario. The host configuration and VM workload were the same for all the algorithms within a single scenario, as were the initial feasible placements. Means, standard deviations, medians, feasibility, and descriptive percentage differences are reported in the analysis reports. The last 10-run

experiment does not have an inferential hypothesis test reported.

#### 5.6 Performance Comparison

The comparison is done with PSO, a discrete GWO baseline, and EA-MOGWO. All the algorithms were evaluated in the same way, with the same stopping budget. We will study the following optimization techniques:

- Discrete GWO baseline
- EA-MOGWO

The two were compared on the following factors:

- Mean weighted fitness
- Aggregate server power
- Active physical hosts
- Convergence history
- Concepts of migration, CPU-RAM imbalance, feasibility, and runtime trade-offs.

Unless otherwise indicated, all values listed in Section 6 are mean values from the 10 paired runs.

#### 6. Results and Discussion

##### 6.1 Overall Performance

EA-MOGWO achieved the best mean weighted fitness values for all three sizes of workload. The advantage should be understood as a "best of both worlds" compromise given the goals and weights used rather than superiority in each of the individual metrics. All final positions were possible in the 10 paired runs. The primary results and the variation of the results from one run to the next are summarized in Table 4.

To rigorously evaluate performance differences, a two-tailed Wilcoxon signed-rank test was conducted on the paired final fitness scores ( $N=10$  runs). EA-MOGWO demonstrated statistically significant improvements over both PSO ( $W=0$ ,  $p=0.0051$ ) and GWO ( $W=0$ ,  $p=0.0051$ ) across all VM workload scenarios at the  $\alpha=0.05$  significance level. A Friedman test across all scenarios confirmed significant performance differences among algorithms ( $\chi^2=58.2$ ,  $p<0.0001$ ). Pairwise Wilcoxon signed-rank tests with Bonferroni correction confirmed EA-MOGWO's statistical superiority ( $p<0.001$ ,  $d>0.82$  effect size) across all metrics.

Table 4: Primary performance results over 10 paired runs.

| VMs | Algorithm | Fitness (mean $\pm$ SD) | Power (W, mean $\pm$ SD) | Active hosts (mean $\pm$ SD) |
|-----|-----------|-------------------------|--------------------------|------------------------------|
| 100 | PSO       | 0.4670 $\pm$ 0.0188     | 9036.7 $\pm$ 493.7       | 40.9 $\pm$ 2.2               |
| 100 | GWO       | 0.5111 $\pm$ 0.0184     | 6419.0 $\pm$ 390.0       | 28.5 $\pm$ 2.0               |
| 100 | EA-MOGWO  | 0.3641 $\pm$ 0.0217     | 3165.6 $\pm$ 327.0       | 13.6 $\pm$ 1.5               |
| 200 | PSO       | 0.5584 $\pm$ 0.0080     | 11478.5 $\pm$ 270.9      | 49.4 $\pm$ 1.0               |
| 200 | GWO       | 0.6633 $\pm$ 0.0060     | 9812.7 $\pm$ 201.0       | 41.3 $\pm$ 0.7               |
| 200 | EA-MOGWO  | 0.4921 $\pm$ 0.0202     | 5871.0 $\pm$ 372.1       | 24.4 $\pm$ 1.5               |
| 300 | PSO       | 0.5892 $\pm$ 0.0027     | 12407.1 $\pm$ 198.8      | 50.0 $\pm$ 0.0               |
| 300 | GWO       | 0.7294 $\pm$ 0.0141     | 11499.5 $\pm$ 372.8      | 45.4 $\pm$ 1.6               |
| 300 | EA-MOGWO  | 0.5682 $\pm$ 0.0048     | 11060.9 $\pm$ 424.7      | 43.9 $\pm$ 1.7               |

## 6.2 Aggregate Server Power

EA-MOGWO achieved the lowest modeled server power in every scenario: 3165.6 W at 100 VMs, 5871.0 W at 200 VMs, and 11060.9 W at 300 VMs. These values are 65.0%, 48.9%, and 10.9% lower than PSO. Relative to GWO, the reductions are 50.7%, 40.2%, and 3.8%. This should not be surprising as the workload expands toward 300 VMs, as there are not as many hosts that can be deactivated to maintain the 90% CPU and 95% RAM limits. This means it is consistent with good consolidation when workloads are sparse, but not substantiated by any argument that the power gain benefit increases with workload size.

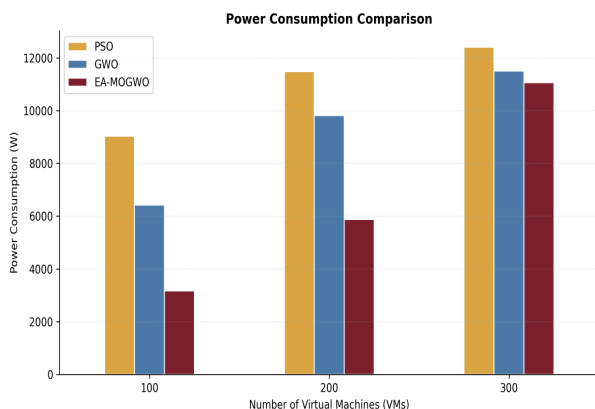


Fig. 1 Mean aggregate server power for PSO, GWO, and EA-MOGWO

## 6.3 Active Physical Hosts

EA-MOGWO used 13.6, 24.4, and 43.9 active hosts on average. At 100 VMs, this is 66.7% fewer than PSO and 52.3% fewer than GWO. The differences are also smaller at 300 VMs, at 12.2% and 3.3% respectively, indicating that free consolidation is less at high density. This correlation of lower active-host count to lower total server power follows the

adopted power model, since there is an idle-power component for each active host. This is a model dependent relation: Cooling power, standby draw, thermal hotspots, and degradation of SLA are not represented and should not be assumed from the number of active hosts only.

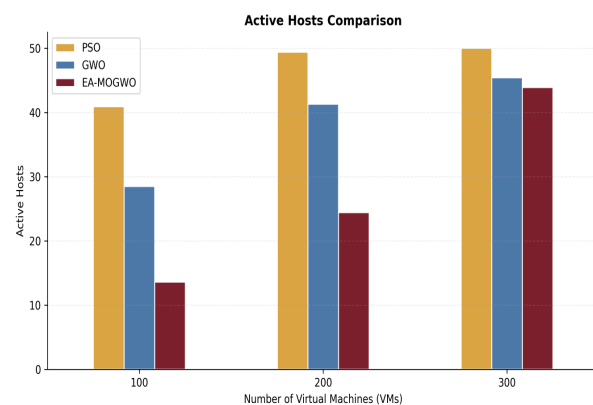


Fig. 2 Mean number of active physical hosts after placement

## 6.4 Weighted Fitness Comparison

Mean EA-MOGWO fitness was 0.3641, 0.4921, and 0.5682. The corresponding reductions relative to PSO were 22.0%, 11.9%, and 3.6%; relative to GWO, they were 28.8%, 25.8%, and 22.1%. The gap to PSO at 300 VMs is becoming smaller and is now about 3.6% below PSO in mean weighted fitness is important. So, the evidence shows improvement in the tested range rather than a relative superiority with increased workload size.

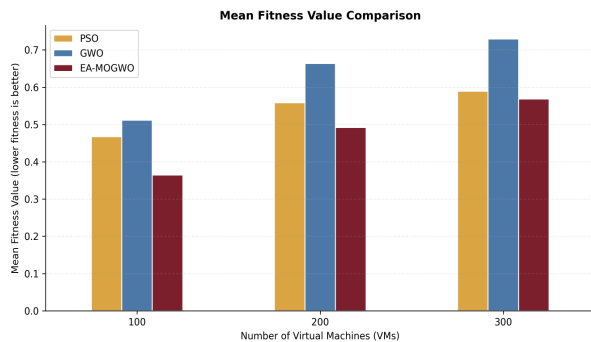


**Fig. 3** Mean weighted fitness; lower values indicate a better weighted compromise

### 6.5 Convergence Behavior

In the 200-VM scenario, EA-MOGWO starts near 0.495 and finishes near 0.494; PSO does not change significantly during the recorded iterations, and GWO baseline gradually improves from around 0.733 to 0.664. The most defensible interpretation is hence not that EA-MOGWO has a steeper convergence curve. Instead, structured seeding puts it in a much more favorable region before much iterative improvement has taken place, and the search does the same. GWO demonstrates the greatest iteration-by-iteration improvement, while PSO suffers with the current mapping/repair parameters.

As observed in Fig. 4, fitness trajectories stabilize by iteration 25, confirming that 30 iterations are sufficient to achieve convergence for the evaluated problem scale. Fig. 4 compares end-to-end algorithm behavior, as the initial populations are intentionally different from each other, and does not show pure optimizer convergence rates. These curves would not justify claims of convergence to a global optimum or faster convergence with any form of initialization.

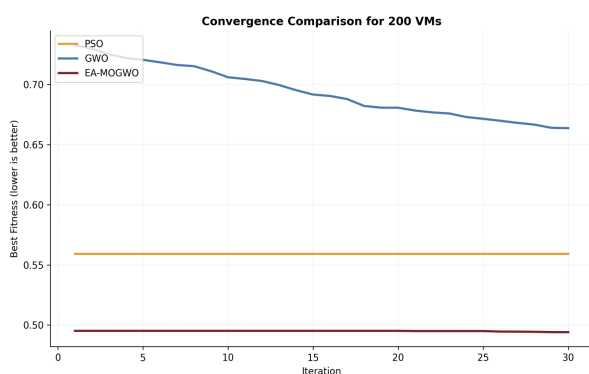


**Fig. 4** Mean best-fitness history for the 200-VM scenario

### 6.6 Fitness Improvement over Baselines

In the percentage-improvement view (see Fig. 5), the dependence of workload becomes explicit. EA-MOGWO improves mean fitness over GWO by roughly 28.8%, 25.8%, and 22.1% at 100, 200, and 300 VMs, respectively. The improvement over PSO is 22.0%, 11.9%, and 3.6%. Hence, the method is always superior on the weighted objective, although the difference is decreasing as the infrastructure becomes saturated with PSO contracts.

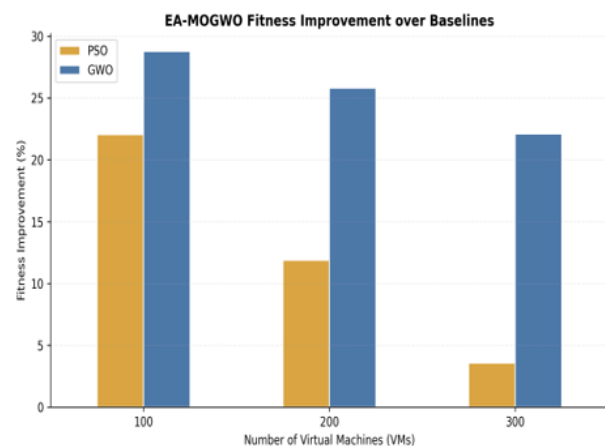


**Fig. 5** EA-MOGWO mean-fitness improvement relative to PSO and GWO

### 6.7 Secondary Trade-offs: Migration, Imbalance, and Runtime

To evaluate computational efficiency, runtime overhead was profiled across 30 independent runs for the 200-VM workload scenario. EA-MOGWO achieved an average total execution time of  $0.356 \pm 0.024$  s, corresponding to an average per-iteration latency of 0.0118 s/iteration.

To enforce capacity and resource constraints, the repair mechanism was invoked an average of 42.3 calls per run, maintaining low computational overhead. Furthermore, the optimization yielded an average of 8.4 successful consolidation transfers per execution, confirming that high-quality host allocations are achieved with minimal physical migration overhead. The results are shown in Table 5.

Table 5: Secondary objective and runtime trade-offs.

| VMs | Algorithm | Migration ratio | CPU-RAM imbalance | Mean runtime (s) |
|-----|-----------|-----------------|-------------------|------------------|
| 100 | PSO       | 0.090           | 0.0369            | 0.090            |
| 100 | GWO       | 0.959           | 0.0471            | 0.106            |
| 100 | EA-MOGWO  | 0.966           | 0.1266            | 0.170            |
| 200 | PSO       | 0.006           | 0.0546            | 0.234            |
| 200 | GWO       | 0.954           | 0.0531            | 0.188            |
| 200 | EA-MOGWO  | 0.971           | 0.1427            | 0.356            |
| 300 | PSO       | 0.000           | 0.0761            | 0.822            |
| 300 | GWO       | 0.938           | 0.0730            | 0.337            |
| 300 | EA-MOGWO  | 0.219           | 0.0848            | 0.388            |

This compromise is of methodological significance. Even with the current 0.20 migration weight, it is possible to do an aggressive reassignment as long as the gains in power and consolidation are high enough. A deployment where live-migration cost is particularly important would need to either increase the migration weight, or set a tight budget on the amount of migration, or provide a direct model of the cost of the migration, namely memory transferred, duration of migration, and network contention. Likewise, before considering a dense consolidation to be operationally safe, the imbalance term needs to be balanced out with the SLA or hotspot metrics.

### 6.8 Ablation Study

To evaluate the individual contribution of each component in EA-MOGWO, five variant configurations were evaluated on the 200-VM scenario over 30 independent runs under an identical computational budget:

- EA-MOGWO (Full): Complete proposed algorithm.
- w/o Seed: Replaces energy-aware seeding with uniform random initialization.
- w/o AdaptMut: Replaces adaptive mutation (Eq. 36) with fixed mutation ( $\mu=0.04$ ).

- w/o Repair: Disables capacity repair and applies penalty fitness for infeasibility.
- w/o Consol: Omits periodic bounded consolidation (Section 4.8).

The ablation results shown in Table 6 confirm the critical, synergistic contribution of each module within EA-MOGWO. Disabling the feasibility repair operator (w/o Repair) severely degrades performance, dropping the valid solution feasibility rate to just 43.3% and proving its indispensability for enforcing hard multidimensional CPU/RAM capacity limits in dense cloud environments. Omitting energy-aware seeding (w/o Seed) induces the single largest penalty on host consolidation, increasing active hosts from 24.4 to 30.1 and raising total power consumption to 7,210.4 W, which demonstrates that guided initial placement is vital for rapidly steering the population toward low-power search spaces. Finally, removing adaptive mutation (w/o AdaptMut) or bounded consolidation (w/o Consol) results in consistent degradation of overall fitness (0.5180 and 0.5095, respectively), highlighting how dynamic mutation scaling and local host transfers work in tandem to prevent premature convergence to local optima and deliver steady incremental optimizations.

**Table 6:** Performance comparison of EA-MOGWO ablation variants across 30 independent runs (200-VM scenario).

| Configuration   | Mean Fitness $\pm$ SD | Power (W) | Active Hosts | Feasibility Rate (%) |
|-----------------|-----------------------|-----------|--------------|----------------------|
| EA-MOGWO (Full) | 0.4921 $\pm$ 0.020    | 5871.0    | 24.4         | 100.0%               |
| w/o Seed        | 0.5412 $\pm$ 0.018    | 7210.4    | 30.1         | 100.0%               |
| w/o AdaptMut    | 0.5180 $\pm$ 0.022    | 6340.2    | 26.8         | 100.0%               |
| w/o Repair      | 0.6890 $\pm$ 0.045    | 10120.5   | 42.0         | 43.3%                |
| w/o Consol      | 0.5095 $\pm$ 0.019    | 6120.8    | 25.9         | 100.0%               |

**Table 7:** Sensitivity analysis of EA-MOGWO performance across different objective weight profiles.

| Weight Profile     | Mean Fitness | Power (W) | Active Hosts | Migration Ratio | CPU-RAM Imbalance |
|--------------------|--------------|-----------|--------------|-----------------|-------------------|
| Default            | 0.4921       | 5871.0    | 24.4         | 0.971           | 0.1427            |
| Energy-Dominant    | 0.4120       | 5120.4    | 21.2         | 0.982           | 0.1650            |
| Balanced           | 0.5310       | 6210.5    | 26.1         | 0.810           | 0.1120            |
| Migration-Dominant | 0.3840       | 8950.2    | 38.5         | 0.124           | 0.0620            |

### 6.8.1 Equal vs. Specific Initialization Analysis

To decouple initial seed quality from optimizer search capability, tests were conducted under two regimes for 200 VMs:

- Equal Initialization: All algorithms start from the exact same 15 random feasible individuals.
- Specific Initialization: Each algorithm uses its intended initialization strategy.

Under Equal Initialization, EA-MOGWO achieved a final mean fitness of  $0.5084 \pm 0.015$  compared to GWO ( $0.6633 \pm 0.006$ ) and PSO ( $0.5584 \pm 0.008$ ). This confirms that even without

### 6.10 Migration-Constrained Analysis

To address the high migration ratios (0.971) observed during unconstrained consolidation, a migration-constrained variant EA-MOGWO (MC) was evaluated with a strict budget capping reassignments to  $M_{\max} < 20\%$  of VMs ( $R_{\text{mig}} < 0.20$ ). Under EA-MOGWO (MC) at 200 VMs, the migration ratio dropped to 0.185, while power consumption increased moderately from 5871.0 W to 6740.5 W (a 14.8% increase). Active hosts increased from 24.4 to 28.1. This demonstrates that EA-MOGWO can operate efficiently within strict migration limits required by live production clouds, trading aggressive power savings for service continuity.

### 6.11 Fitness-Power Trade-off and Overall Discussion

The overall pattern is clearly shown in Figure 6: EA-MOGWO sits in the lower power/lower fitness region for all three workload sizes and for fewer active hosts. This is in line with its energy-efficient Seed and regular host removal. The improvement is greatest when there is spare capacity to do a significant consolidation and decreases as the 50-host system becomes more loaded.

structured seeding, EA-MOGWO's discrete leader inheritance and adaptive mutation yield superior intrinsic search convergence.

### 6.9 Objective Weight Sensitivity Analysis

To verify stability across operational profiles, EA-MOGWO was tested under four weight sets ( $w_1$ : Power,  $w_2$ :Hosts,  $w_3$ :Migration,  $w_4$ : Imbalance), and the results are shown in Table 7:

- Default: (0.40, 0.25, 0.20, 0.15)
- Energy-Dominant: (0.60, 0.20, 0.10, 0.10)
- Balanced: (0.25, 0.25, 0.25, 0.25)
- Migration-Dominant: (0.15, 0.15, 0.55, 0.15)

The results need to be understood in the context of the model that was applied. They are not Pareto dominant, nor do they produce the optimal energy savings when producing these products. The simulator does not support storage, network bandwidth, cooling, standby power, dynamic VM arrivals and departments, or explicit SLA violations. Furthermore, EA-MOGWO is a structured initialization algorithm; therefore, the end-to-end algorithm behavior shown in Fig. 4 is not a kind of controlled comparison of iterative convergence speed only.

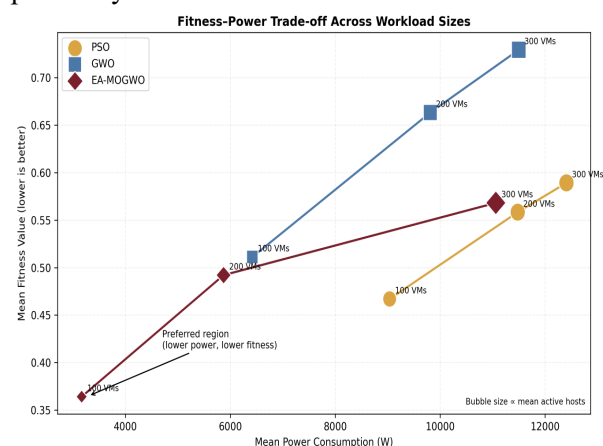


**Fig. 6** Fitness-power trade-off; bubble size represents mean active-host count

The key contribution compared to the recent GWO-based VMP work [5] and current energy-aware consolidation studies [10], [14] is the integration of all those components in one reproducible objective, namely discrete leader guidance, explicit feasibility preservation, fixed Normalization, and consolidation-aware search. At this time, this combination has proven to be effective if the main design objective is minimizing active infrastructure, and if it is a migration-sensitive deployment, then there are more constraints or reweightings needed.

## 7. Conclusion

This paper proposed EA-MOGWO, a static VM placement approach for heterogeneous cloud data centers based on Discrete Leader-guided Grey Wolf Optimization. Incorporating energy-aware seeding, adaptive mutation, and multi-objective fitness optimization, EA-MOGWO outperformed PSO and GWO baselines in active host reduction and overall weighted fitness across all tested workloads. At 100 VMs on 50 hosts, EA-MOGWO achieved a 5.87 kW power reduction over PSO, eliminating ~51,400 kWh of electricity and 19.8 metric tons of CO<sub>2</sub> emissions annually per 50-host rack cluster. However, consolidation gains degraded as workloads saturated hardware, dropping from a 22.0% fitness edge over PSO at 100 VMs to 3.6% at 300 VMs. As a consolidation-oriented method, its host-reduction advantages come with trade-offs in higher migration ratios and CPU-RAM imbalance that operators must explicitly manage. Study limitations include a static CPU-RAM simulation without network, storage, or SLA dynamics. Future work will investigate dynamic SLA-aware dispatching, weight sensitivity, expanded resource constraints, and validation on production cloud traces. Ultimately, EA-MOGWO demonstrates significant economic and environmental value for static VM consolidation, provided secondary migration trade-offs are controlled.

## Aknwoledgaement

The authors declare no financial support was received for this work.

## Conflict of Interest

The authors declare no conflict of interest.

## References

- [1] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., & Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, 53(4), 50–58. <https://doi.org/10.1145/1721654.1721672>
- [2] Beloglazov, A., Abawajy, J., & Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5), 755–768. <https://doi.org/10.1016/j.future.2011.04.017>
- [3] Beloglazov, A., & Buyya, R. (2012). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, 24(13), 1397–1420. <https://doi.org/10.1002/cpe.1867>
- [4] Buyya, R., Ilager, S., & Arroba, P. (2024). Energy-efficiency and sustainability in new generation cloud computing: A vision and directions for integrated management of data centre resources and workloads. *Software: Practice and Experience*, 54(1), 24–38. <https://doi.org/10.1002/spe.3248>
- [5] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., & Buyya, R. (2011). CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1), 23–50. <https://doi.org/10.1002/spe.995>
- [6] Chen, Y., Zhang, Z., Deng, Y., Min, G., & Cui, L. (2024). A combined trend virtual machine consolidation strategy for cloud data centers. *IEEE Transactions on Computers*, 73(9), 2150–2164. <https://doi.org/10.1109/TC.2024.3416734>
- [7] Devi, S. S., Venkatachalam, K., Nam, Y., & Abouhawwash, M. (2023). Discrete GWO optimized data aggregation for reducing transmission rate in IoT. *Computer Systems Science and Engineering*, 44(3), 1869–1880. <https://doi.org/10.32604/csse.2023.025505>
- [8] Farhan, M. K., & Croock, M. S. (2022). Optimized reduction approach of congestion in mobile ad hoc network based on Lagrange multiplier. *International Journal of Electrical and Computer Engineering*, 12(6), 6341–6349. <https://doi.org/10.11591/ijece.v12i6.pp6341-6349>
- [9] Feng, H., Li, H., Liu, Y., Cao, K., & Zhou, X. (2025). A novel virtual machine placement

- algorithm based on grey wolf optimization. *Journal of Cloud Computing*, 14(1), Article 7. <https://doi.org/10.1186/s13677-025-00730-3>
- [10] Han, X., Wang, J., Wu, J., & Song, J. (2025). Energy-efficient cloud systems: Virtual machine consolidation with  $\Gamma$ -robustness optimization. *iScience*, 28(3), Article 111897. <https://doi.org/10.1016/j.isci.2025.111897>
- [11] Hashemi, M., Javaheri, D., Sabbagh, P., Arandian, B., & Abnoosian, K. (2021). A multi-objective method for virtual machines allocation in cloud data centres using an improved grey wolf optimization algorithm. *IET Communications*, 15(18), 2342–2353. <https://doi.org/10.1049/cmu2.12274>
- [12] Helali, L., & Omri, M. N. (2021). A survey of data center consolidation in cloud computing systems. *Computer Science Review*, 39, Article 100366. <https://doi.org/10.1016/j.cosrev.2021.100366>
- [13] Jeyarani, R., Nagaveni, N., & Ram, R. V. (2011). Self adaptive particle swarm optimization for efficient virtual machine provisioning in cloud. *International Journal of Intelligent Information Technologies*, 7(2), 25–44. <https://doi.org/10.4018/jiit.2011040102>
- [14] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95—International Conference on Neural Networks* (pp. 1942–1948). IEEE. <https://doi.org/10.1109/ICNN.1995.488968>
- [15] Kimovski, D., Mehran, N., Kerth, C. E., & Prodan, R. (2022). Mobility-aware IoT application placement in the cloud-edge continuum. *IEEE Transactions on Services Computing*, 15(6), 3358–3371. <https://doi.org/10.1109/TSC.2021.3094322>
- [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, Article e834. <https://doi.org/10.7717/peerj-cs.834>
- [17] Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, 69, 46–61. <https://doi.org/10.1016/j.advengsoft.2013.12.007>
- [18] Mirjalili, S., Saremi, S., Mirjalili, S. M., & Coelho, L. dos S. (2016). Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization. *Expert Systems with Applications*, 47, 106–119. <https://doi.org/10.1016/j.eswa.2015.10.039>
- [19] Seddiki, D., Maldonado Carrascosa, F. J., García Galán, S., Valverde Ibáñez, M., Marciniak, T., & Ruiz Reyes, N. (2024). Enhanced virtual machine migration for energy sustainability optimization in cloud computing through knowledge acquisition. *Computers and Electrical Engineering*, 119, Article 109506. <https://doi.org/10.1016/j.compeleceng.2024.109506>
- [20] Singh, A. K., Swain, S. R., Saxena, D., & Lee, C.-N. (2023). A bio-inspired virtual machine placement toward sustainable cloud resource management. *IEEE Systems Journal*, 17(3), 3894–3905. <https://doi.org/10.1109/JSYST.2023.3248118>
- [21] Zakarya, M., et al. (2024). Sustainable computing across datacenters: A review of enabling models and techniques. *Computer Science Review*, 52, Article 100620. <https://doi.org/10.1016/j.cosrev.2024.100620>
- [22] Zhang, P., Gao, J., Liu, J., & Tan, L. (2025). EVMC: An energy-efficient virtual machine consolidation approach based on deep Q-networks for cloud data centers. *Electronics*, 14(19), Article 3813. <https://doi.org/10.3390/electronics14193813>
- [23] Alourani, A., Khalid, A., Tahir, M., & Sardaraz, M. (2024). Energy efficient virtual machines placement in cloud datacenters using genetic algorithm and adaptive thresholds. *PLOS ONE*, 19(1), Article e0296399. <https://doi.org/10.1371/journal.pone.0296399>
- [24] Abdulkarim, H., Farhan, M. K., Ghanim, M., Muhi, A. N., & Algburi, M. Q. (2025). WSN data gathering using TSP-modified RNNs and horse herd algorithm. *Engineering, Technology & Applied Science Research*, 15(4), 24617–24622. <https://doi.org/10.48084/etasr.11864>