SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V11I12P104

Volume 11 Issue 12, 32-43, December 2024

Original Article

Enhancement of ACOTS Algorithm for Virtual Machine Placements in Cloud Data Centers

Jyotsna P. Gabhane[1], Sunil Pathak[2], Nita Thakare[3]

[1,2]*Department of Computer Science and Engineering, Amity School of Engineering & Technology, Amity University Rajasthan, Jaipur, India.*
[3]*Department of Computer Technology, Priyadarshini College of Engineering, Nagpur, India*.

[1]*Corresponding Author : jyotspg@gmail.com*

Received: 10 October 2024 Revised: 13 November 2024 Accepted: 04 December 2024 Published: 30 December 2024

Abstract - Since *many cloud data centers provide worldwide services on demand, it has been a popular research topic in recent years. Users and virtual machines grow in the number directly to data centers' expanding size. Virtual Machine Placement methods are mainly used in data centers to consolidate servers. Therefore, the placement of virtual machines is the most active research area today. The performance of VM Placements in cloud computing depends on various factors, including resource management, power consumption, and others. This paper provides the optimal solution for VM placements in cloud environments using ACOTS (Ant Colony Optimization with Tabu Search) hybrid metaheuristic algorithm enhancing with the feature of modified Eagle strategy. The hybridization leverages ACO's global search capabilities, TS's memory-driven diversification, and the adaptive exploration-exploitation balance of EagleMOD (modified EAGLE strategy). By integrating these techniques, the proposed algorithm enhances convergence speed and resolves the issue of local optima, ensuring robust performance across various scenarios.*

Keywords - *Cloud computing, Meta-heuristic algorithms, Optimal solution, Resource management, Virtual machine placement, Power consumption.*

1. Introduction

One well-known source of high-performance services is the cloud. On-demand services offered by cloud computing give users online access to a number of very useful resources. All of this makes cloud computing dynamically scalable. Task scheduling is essential to cloud computing because it increases the efficiency of several services and drastically lowers energy consumption in cloud environments and Internet of Things (IoT) networks. Choosing the best and most appropriate scheduling algorithm is essential for task scheduling and resource allocation [1]. It requires that jobs or tasks be assigned to a resource most efficiently and appropriately. It is necessary to consider and track a number of factors, such as resource usage, makespan, accessibility, time, scalability, cost, and others[2]. They are accessible over the internet. The Cloud can be viewed as a conceptual layer on top of the Internet, providing transparency to a data centre's software and hardware resources. This transparency enables accessibility through a clearly defined interface [3, 4].

All users of this multi-tenant system utilize the same provider-maintained code base. The hardware, network, and operating system are not under the user's control [5].

The biggest issue to be solved is how to reduce the electrical power usage of data stored in cloud centers, which has been noticed to have increased due to the widespread use of cloud computing [6]. It chooses the fewest PMs (Physical Machines) that can provide what is needed for hosting certain amounts of VMs (Virtual Machines) while consuming the least amount of power. It is an NP-hard combined issue to optimize VMP. Diverse, opposing goals can be used to address the issue. The VMP challenge was successfully solved for various goals, including alone, numerous, and many objective processes [6].In relation to Virtual Machine Placement (VMP), the study focuses on the idea of computing in the cloud. Addressing distinct issues frameworks, such as VM allocation and VMP Optimization methods, is a part of VMP. Assuring Quality of Service (also called QoS) while distributing virtual servers to physical computers is known as VM allocation. An SLA is an agreement between an end user and a service supplier regarding the latter's offerings' calibre, accountability, and availability. As a result, maintaining a perfect equilibrium between QoS and energy use is today's most challenging problem [7].

In the literature, current techniques for this optimization problem have been discussed. This study's

main goal is to find pertinent problem compositions. The Eagle Strategy, well-known for balancing exploration and exploitation by combining local refinement with global search, is a great place to start when hybridizing. Adding ACOTS aims to enhance performance by leveraging ACO's global optimization capabilities and TS's memory-driven diversification.

Because ACO depends on pheromone updates, it can stall in local optima even though it is good at exploring solutions. Although TS is better at fine-tuning solutions, its deterministic nature and reliance on a fixed tabu list size restrict its applicability to larger problem spaces. In many hybrid algorithms, exploration (global search) and exploitation (local refinement) are not sufficiently balanced, leading to less-than-ideal trade-offs in crucial metrics like makespan, resource usage, and power consumption. Creating and evaluating a Hybrid Optimization Framework that combines the enhanced Eagle Strategy with ACOTS is necessary to overcome these challenges. In order to avoid premature convergence, it employs adaptive exploration techniques. To improve scalability, it leverages TS's diversity mechanisms.

In particular, Section 2 reviews research on the background and fundamentals of cloud computing and its architecture, traits, and deployment model. This is followed by describing how virtual machines are placed in the cloud. The VMP optimization strategies and objective functions are the main topics of this section. Additionally covered are current metaheuristic placement strategies for virtual machines. The suggested method for placing VMs in a cloud-based setup is explained in the following section. The assessment of the suggested examines, and the outcome of the reviews is covered in the following section. In the final section, the last words are provided.

2. Literature Review

Splitting the real machine's physical assets, such as its hard drive storage and processing memory, signal strength network, and power consumption, into various environments for operation known as virtual machines (VMs) is a form of storage virtualization. The virtualization process increases the Return on Investment (ROI) by creating multiple VMs to increase resource utilization on a single physical server.

Including virtual machines is simply the assignment of appropriate virtual machines to every PM in a remote data center. The positioning strategy for the VM to PMs connecting is another name for it [8]. As its name suggests, a mono-objective tackle only considers optimizing just a single resource. Multiple objective functions are individually optimized, one at each stage. According to the survey, 61.9% of the studied research planned a mono-objective way (MOP) for resolving the VM placement issues [9].

The optimization of multiple goals resolved as a single objective function is referred to as a multifaceted view resolved as a mono-objective in this context. An issue with this approach is that, in most cases, it is impossible to allow a correct combination of the objective functions because doing so requires an in-depth knowledge of the issue at hand. Pure multi-objectives refer to optimizing multiple goals at once. A PMO approach might produce a more practical solution despite much research already being done on single objective methods for operating an actual cloud system [10].

Many-objective optimization problems are defined informally as having three or more objective functions up to twenty objectives. Numerous real-life applications, including engineering layouts, frequently use MaOPs [11]. The specific research has not offered the creation of multi objective optimization issues for the VMP problem [12, 13].

The procedure of choosing the best Physical Machine (PM) over a specific Virtual Machine (VM) is known as virtual machine positioning. Therefore, whether it is an initial VM placement or a VM movement for positioning re-optimization, a VM installation method aims at identifying an ideal VM to PM mapping. One of the two main objectives of the deployment method in consolidated virtual machines can be energy savings or QoS delivery. These two objectives obviously go against one another [14]. In [15], an altered version of the ant colony optimizing approach is suggested to distribute load across nodes in the cloud or the grid system networks.

In [16], a multifaceted optimization issue is addressed by considering two objectives for the formulation of virtual machine placement: all storage wasted and total processing wasted resources. The proposed ant colony optimization algorithm is developed to find a set of non-dominated alternatives to massive data centers. Another strategy using Order Exchange and Migration (OEM) local discovery methods in conjunction with ant colony optimization has been suggested [17]; the outcome of this approach is known as an OEMACS. According to the author, OEMACS performs better than other evolutionary-based methods and traditional heuristics in most situations. It offers substantial energy savings and more effective use of various resources.

In order to solve the VM scheduling problem, a novel hybrid algorithm called ACOPS (Ant Colony Optimization with Particle Swarm) was proposed [18]. It considers three resources for load distribution, including RAM, processor utilization, and disc utilization. Without further facts, ACOPS uses historical data to anticipate new data demands and adapt to changing circumstances. The suggested algorithm outperforms other methods and can maintain load balance in a changing environment. Power consumption, load balancing, and migration time were addressed by the

hybrid model form of the multi-objective biogeography-based optimization method with variation in evolution (MBBO/DE) [19], which has been suggested for VMP challenges.

A blend of algorithms called GATA has been suggested, combining the algorithm known as genetics and the Tabu search algorithm with an energy-aware goal for the VMP issue. The algorithm is contrasted with a few existing VMP algorithms to minimize data center energy use while increasing load equilibrium. The experimental findings demonstrate that when contrasted to both heuristic algorithms and a recently proposed ant colony algorithm, the GATA algorithm may produce a more environmentally friendly and feasible placement scheme in a suitable period [20].

The cuckoo search method inspired an innovative VM placement method called ECS, which was suggested [21] and aimed to optimize total energy use and waste of resources in data centers in the cloud centers. In addition to stable performance for SLA and VM migration, this work demonstrated an improved energy metric for assessing the efficiency of the suggested ECS method with VM choosing regulations like Minimum Migration Time (MMT) and Minimum Utilization (MU). The comparison's findings demonstrated that ECS operates efficiently when there are fewer virtual machines (i.e., less workload). Due to its ability to use the levy flight system, this algorithm can operate effectively even in a massive, changing cloud environment.

The discrete firefly algorithm proposes a multi-objective constraint optimization framework for VM placement that considers system effectiveness and safety. In order to deal with the side-channel take on problem on the IaaS structure, this research also suggested a successful VM positioning approach. From the perspective of physical isolation, it allocates VMs for unauthorized users and targets occupants on different physical hosts [22].

The main objective of the previous study was to review the existing meta-heuristic approaches for different objectives and environments. Each algorithm is quite different from each other. Each of them works on different perspectives and objectives. The most commonly studied objectives are optimizing energy consumption and resource allocation. So, for future work, it is a good idea to consider additional objectives comparable to resource balancing, traffic communication, number of VM migrations, migration time, economic, SLA violations, and resource utilization. Selecting a good mixture of algorithms can improve the performance of the system. So, selecting a proper hybridization is again a part of research. It should not complicate the problem execution [10]. The Enhanced solution for optimizing the VMP in a cloud environment is discussed in the next section.

3. VMP Modelling and Problem Description

To efficiently assign virtual machines (VMs) to physical machines (PMs), the Virtual Machine Placement (VMP) model consists of several parts. Here is a description of a typical VMP model's architecture [23]:

- Input Data: The VMP model needs input data to function, and this data describes the traits of the VMs and PMs. This information includes each VM's requirements for computational resources, each PM's available computational capacity, the communication cost between VMs and PMs, and other pertinent details.
- Objective Function: The objective function specifies the VMP model's optimization objective. It quantifies the performance metric that needs to be increased or decreased, such as energy consumption, resource utilization, or communication latency.
- Decision Variables: The decision variables represent the VMP model's placement choices. The decision variable in this instance is a binary variable with the notation x_i, j, which indicates whether VM i is positioned on PM j. In addition, PM j is indicated by a binary variable with the symbol y_j.
- Constraints: The constraints ensure that placement choices abide by specific requirements and restrictions. These limitations consist of:
  - Each VM must have exactly one PM: In V, (x_i,j) = 1 for all i.
  - No PM may have a capacity greater than (C_i * x_i,j) U_j for all j in P.
  - A minimum of one VM must be assigned to a PM if one is being used: For every j in P, y_j = (x_i,j).
- Solver/Algorithm: The VMP model employs an optimization solver or algorithm to identify the best possible placement that satisfies the constraints and objective function. To effectively explore the solution space and find the best VM-PM assignments, the solver uses linear programming, mixed-integer programming, or heuristics/metaheuristics.
- Output: the optimal or nearly optimal placement solution, which identifies the assignment of VMs to PMs, is the output of the VMP model. This output can be used as a reference for managing the placement of virtual machines in a data centre or cloud environment and allocating resources accordingly.

The VMP model can be expanded or altered depending on specific needs to consider additional constraints, goals, or system-specific factors. The objective of this research is to create a placement strategy that is efficient and effective and optimizes resource use, energy consumption, or other relevant factors in the environment.

3.1. Problem Description

The Virtual Machine Placement (VMP) problem is a combined optimization issue that seeks to distribute virtual machines (VMs) to physical machines (PMs) in a data center while optimizing some goals, like reducing the use of energy, enhancing resource utilization, or lowering communication latency [24]. The specific goals and restrictions taken into account can change how the VMP problem is mathematically formulated. A general formulation is given here:

Sets:

V: A collection of VMs, where an index identifies each VM. i.

P: A collection of PMs, where an index j identifies each PM.

Parameters:

M_i: The list of PMs a VM may be assigned to.

C_i: VM i's demand for computational resources.

D_i,j: The communication cost over the PM j and VM i network.

U_j: The total amount of computing power PM j is capable of.

E_j: The amount of energy used by PM j.

Decision-making factors

Binary variable x_i, j reveals whether VM i is attached to PM j.

y_j: A binary variable indicating the use of PM j.

For modified Eagle Strategy ACOTS Formulation, three stages can be used to model the hybrid approach.
Let us define the key variables for the combined formulation:
S: Solution space (all task-resource mappings).
X: Current solution.
f(X): Objective function value
P(X): Probability of choosing solution X (from ACO pheromones).
L(X): Tabu list memory, indicating forbidden solutions.
Global Exploration: For mapping tasks to resources and conducting extensive exploration. The Eagle strategy uses Gaussian-based random walks or Levy flight to explore the solution space S and find promising areas. It alters solutions by:

$$X_{new} = X_{current} + \alpha . Levy(\beta) \quad (1)$$

Where:

- α: Step size scaling factor.
- β: Parameter controlling Levy flight's distribution (typically $1<\beta\leq 3$).
- Levy(β): A stochastic jump based on the Levy distribution.

For task scheduling, Eagle's global search maps tasks Ti to resources Rj where:

Resource Assignment: $R_j = \arg R_{\min} f(Ti, R)$
$f(Ti, R)$ Includes power consumption, execution time, or any cost metric.

Guided Exploration: Using pheromone trails, guided exploration optimizes task assignments. Following the discovery of a promising area by global exploration, ACO uses pheromones to direct task-resource assignment:

$$P(\,Rj \mid Ti\,) = \frac{\tau(R_j)^{\alpha}.\eta(R_j|T_i)^{\beta}}{\sum_{k\varepsilon R}\tau(R_k)^{\alpha}.\eta(R_k|T_i)^{\beta}} \quad (2)$$

Where:

- $\tau(R_j)$: Pheromone intensity for resource R_j
- $\eta(R_j \mid T_i)$: Heuristic desirability (e.g., the computational power of Rj for task Ti).
- α,β: Balancing factors for pheromone vs. heuristic importance.

The pheromones are updated after every iteration:

$$\tau(R_j) = (1-\rho).\tau(R_j) + \Delta\tau(R_j) \quad (3)$$

Where:
ρ: Evaporation rate.
$\Delta\tau(R_j)$: Pheromone deposit based on solution quality.
Local Refinement: To use memory to stabilize and improve the schedule. By examining the area around the current solution, Tabu Search improves solution X.

N(X)={X′: minor adjustments to X}'s task-resource mappings

To make sure it is not in the Tabu List L(X), the best neighbour is selected:

$$X_{next} = arg^{min}_{X'\in N(X), X'\notin L(X)} f(X') \quad (4)$$

The Tabu List L(X) ensures diversification:
L(X): stores recently visited solutions.
|L(X)|: Tabu list size.

The specific objective of the VMP problem determines the objective function. Here are a few illustrations:

3.1.1. Minimization on Energy Use

Reduce the total amount of energy used by the assigned PMs:

$$Re = (Ej * y_j) \quad (5)$$

3.1.2. Increase Resource Usage

Increase the allotted VMs' combined resource usage.

$$Ir = (Ci * x_{ij}) \quad (6)$$

3.2. Modeling Energy Consumption

Energy consumption modelling is an essential component of the Virtual Machine Placement (VMP) problem. Using the energy consumption model, we can calculate and optimize the energy usage of physical machines (PMs) in the data centre. The VMP problem can be used to model energy consumption in the following ways [25]:

Measures of energy consumption: It is necessary to identify the elements affecting a PM's energy usage to model energy consumption. These specifications could include the power a PM uses when no VMs are running in the background.

- CPU power consumption: The amount of energy a PM's CPU uses when running virtual machines and carrying out computational tasks.
- Memory power consumption: The amount of energy a PM's memory subsystem uses.
- Network power consumption is the amount of energy used by a PM's network interface to connect to VMs and other components.

It is possible to find these parameters through measurements, product specifications, or empirical research.

We can use a linear power model to express the overall CPU utilization (Pc) power consumption. Here is an illustration formula:

$$Pc = Ip + (Cp - Ip) * Tuc \quad (7)$$

Idle Power(Ip) measures how much energy is used when the CPU is not. The additional power used by the CPU when it is active and performing computational tasks is known as CPU Power(Cp). The CPU's utilization, which ranges from 0 to 1, is represented by the term "CPU Utilization," which shows what percentage of its maximum capacity is used. The phrase "CPU Power - Idle Power" refers to the extra power used by the CPU when it is in use instead of when it is idle. We calculate the power used by the CPU by multiplying this value by the Total CPU utilization (Tuc).

The sum of the power consumption of all used Physical Machines (PMs) in the data centre can be used to calculate the total power consumption in a Virtual Machine Placement (VMP) model. The particular power model used for each PM determines the formula for the overall power consumption [23]. Here is a basic formula:

$$Tpc = Pc * y_j \quad (8)$$

Power Consumption represents the power usage of PMj.

A binary variable named y_j indicates whether or not PM j is being used. y_j will be 1 if PM_j is used (with at least one VM allocated) and 0 otherwise.

3.3. Modeling Resource Utilization

A crucial component of the Virtual Machine Placement (VMP) problem is modeling resource utilization. It calculates how the data center's computational resources are allocated and used. An overview of how the VMP problem can model resource utilization is given below:

- Parameters for Resource Utilization:
  The key variables that define the resource needs of Virtual Machines (VMs) and the resources available to Physical Machines (PMs) must be identified to model resource utilization. These specifications could include:
  The amount of CPU, memory, storage, and other resources needed by each VM is called their computational resource demand.
- Computing power: The sum of each PM's CPU, memory, storage, and other resources.

Measurements, virtual machine configurations, or resource profiling can all be used to get these parameters.

3.4. Resource Utilization Model

After determining the resource utilization (Cu) parameters, a utilisation model is built. The model connects a PM's resource use to the VMs assigned.

The model may be linear or nonlinear depending on how complex the resource usage behavior is.

$$Cu = \sum(\text{CPU Demand\_i} * \text{x\_i,j}) \,/\, \text{CPU Capacity\_j} \quad (9)$$

Here, x_i,j is a binary variable indicating whether VM_i is placed on PM_j, and CPU Capacity_j represents the CPU capacity of PM_j. CPU Demand_i represents the CPU resource demand of VM_i.

The total resource utilization in a Virtual Machine Placement (VMP) model can be formulated as the sum of all allocated Virtual Machines (VMs) resource utilisation.

The formula for the total resource utilization (Tru) depends on the specific resources considered, such as CPU, memory, or network resources. Here is a general formula:

$$Tru = \sum(\text{Resource Utilization\_i}) \quad (10)$$

In this formula:
- Resource Utilization_i represents the resource utilization of VM_i.

The resource utilization of a VM can be expressed as a fraction or percentage of the maximum capacity of the corresponding resource. The specific resource utilization metric depends on the type of resource being considered.

Memory, storage, and other resources can all be modeled similarly, taking into account the demands and capabilities of each.

3.5. The Fitness Function

A fitness function is used in evolutionary algorithms or metaheuristic approaches to evaluate and compare various solutions or individuals in the search space when applied to the Virtual Machine Placement (VMP) problem. Based on the problem's stated objectives and restrictions, the fitness function assesses a particular VM placement solution's quality or desirability.

Usually, the fitness function is created to reflect the VMP problem's optimization objective [26]. Examples of fitness functions for the VMP problem include the following:

3.5.1. Minimizing Energy Consumption

The fitness function is the opposite of the allocated PMs' overall energy consumption. The fitness value(Fv) should be lower for better solutions since the objective is to minimize Energy Consumption(EC).

$$Fv = \sum \mathrm{ECj} * \mathrm{y_j} \tag{11}$$

3.5.2. Maximize Resource Usage

The total resource usage of the assigned VMs can be referred to as the fitness function(Ft). The fitness value should be higher for better solutions because the objective is to maximize resource utilization (Ru).

$$Ft = \sum Ru * x\_ij \tag{12}$$

A weighted sum approach has been applied to combine resource utilization and power consumption into a single fitness function. Here is the formula for the fitness function (Tfit):

$$Tfit = \big(Pwt * ((1 - Tpc)\big) + (Uwt * Tru) \tag{13}$$

In this formula, Resource Utilization represents the resource utilization value calculated using the earlier formula, and Power Consumption represents the power consumption value calculated using the formula provided earlier.

Parameter w is the weight or trade-off factor determining the importance of resource utilisation versus power consumption in fitness evaluation. It should be between 0 and 1, where 0 prioritises power consumption, and 1 prioritises resource utilization. For example, a higher weight on resource utilization (w close to 1) will prioritize maximizing resource utilization while considering power consumption, whereas a lower weight on resource utilization (w close to 0) will prioritize minimizing power consumption while considering resource utilization.

4. The Enhancement of the ACOTS Hybrid Algorithm with the Eagle Strategy

This section presents the enhancement of the ACOTS by enriching the features of the Eagle strategy. This approach adopted an energy-awareness local appropriateness first strategy to update the VM position and improve problem-solving efficiency. The proposed approach can be better in many scenarios in terms of energy reduction. ACOTS, which combines the memory-based local refinement of Tabu Search, the guided exploration of ACO, and the global search capabilities of the modified Eagle strategy, performs better with ACOTS. This framework's superiority in metrics such as makespan, power consumption, CPU utilization, and resource utilization is mathematically and practically demonstrated. Table 1 shows the pseudo code for the ACOTS-EagleMOD Strategies.

Inspired by eagle hunting, the Eagle strategy is a metaheuristic that blends local exploitation (to improve solutions in those areas) with global exploration (to identify promising areas). It is distinguished by:

- Global Search: Diverse exploration using a probabilistic method such as Levy flight or another random distribution.
- Local Search: Concentrating on fine-tuning in areas with high potential that the global search found.

Pheromone trails are used in Ant Colony Optimization to direct the search for the best answers. To improve exploitation, Tabu Search uses memory structures to prevent going back to previously investigated (or less-than-ideal) solutions. The Eagle strategy's random global search lessens ACO's drawback of becoming trapped in local optima [27, 28]. Eagle Strategy and Tabu Search ensure diversity in exploration by bolstering local refinement with memory. This hybrid strategy aims to reduce makespan (total execution), Optimize task scheduling to minimize power consumption and increase CPU utilization and resource allocation effectiveness.

The Eagle Strategy's efficiency during the global exploration stage can be significantly improved by including an adaptive search radius. By dynamically adjusting the search radius according to workload or virtual machine (VM) distribution, the algorithm can initially employ a larger radius to cover a larger portion of the solution space and then gradually reduce it as solutions begin to converge. This approach helps to preserve a balance between exploration and exploitation by avoiding early convergence to local optima and improving the ability to find globally optimal solutions [2].

By combining the exploration-exploitation balance of Modified Eagle Strategy (EAGLEMOD), the local search refinement of Tabu Search (TS), and the global search capabilities of Ant Colony Optimization (ACO), the Enhanced ACOTS_EAGLEMOD algorithm is intended to address the VM Placement problem. The flow diagram in Fig. 1 shows how these techniques complement one another to produce the best possible outcome.

Table 1. The Pseudo code for the ACOTS-EagleMOD strategy

```
BEGIN
// Initialize parameters
Initialize:
   num_iterations
   num_ants
   tabu_tenure
   evaporation_rate
   pheromone_deposit
   step_size
// Define objective functions
Define objective_function_ACO(x, y)
Define objective_function_Eagle(x, y)
// Initialize pheromone matrix
pheromone_matrix ← [[1.0] for i = 1 to num_positions]
// Generate initial solution
initial_solution ← generate_initial_solution(num_positions)
// Define ant solution construction
FUNCTION construct_ant_solution(pheromone_matrix,
heuristic_matrix)
   Use Ant Colony Optimization (ACO) strategy to construct a
solution
END FUNCTION
// Define Eagle Strategy
FUNCTION perform_eagle_strategy(pheromone_matrix,
heuristic_matrix)
   Apply Eagle Strategy operations
END FUNCTION
// Update the best solution and fitness
FUNCTION update_best_solution(new_solution, best_solution,
best_fitness)
   IF objective_function(new_solution) < best_fitness THEN
      best_solution ← new_solution
      best_fitness ← objective_function(new_solution)
   END IF
   RETURN best_solution
END FUNCTION
// Define Tabu Search
FUNCTION perform_tabu_search(initial_solution)
   current_solution ← initial_solution
   tabu_list ← []
   best_solution ← initial_solution
   WHILE stopping_condition NOT met DO
      neighboring_solutions ←
generate_neighboring_solutions(current_solution)
      best_neighboring_solution ← NULL
      FOR solution IN neighboring_solutions DO
         IF best_neighboring_solution IS NULL OR
objective_function(solution) <
objective_function(best_neighboring_solution) THEN
            best_neighboring_solution ← solution
         END IF
      END FOR
      // Update tabu list
      IF tabu_list.size > tabu_tenure THEN
         Remove oldest element from tabu_list
      END IF
      // Update current and best solutions
      current_solution ← best_neighboring_solution
      IF objective_function(current_solution) <
objective_function(best_solution) THEN
         best_solution ← current_solution
      END IF
   END WHILE
   RETURN best_solution
END FUNCTION
// Run the hybrid algorithm
FUNCTION hybrid_algorithm()
   best_solution_ACO ← initial_solution
   best_fitness_ACO ←
objective_function_ACO(initial_solution)
   best_solution_Eagle ← initial_solution
   best_fitness_Eagle ←
objective_function_Eagle(initial_solution)
   FOR iteration = 1 TO num_iterations DO
      // Apply ACO
      new_solution_ACO ←
construct_ant_solution(pheromone_matrix, heuristic_matrix)
      new_fitness_ACO ←
objective_function_ACO(new_solution_ACO)
      best_solution_ACO ←
update_best_solution(new_solution_ACO, best_solution_ACO,
best_fitness_ACO)
      // Apply Eagle Strategy
      new_solution_Eagle ←
perform_eagle_strategy(pheromone_matrix, heuristic_matrix)
      new_fitness_Eagle ←
objective_function_Eagle(new_solution_Eagle)
      best_solution_Eagle ←
update_best_solution(new_solution_Eagle, best_solution_Eagle,
best_fitness_Eagle)
      // Compare solutions
      IF best_fitness_ACO < best_fitness_Eagle THEN
         best_fitness_hybrid ← best_fitness_ACO
      ELSE
         best_fitness_hybrid ← best_fitness_Eagle
      END IF
   END FOR
   final_solution ← perform_tabu_search(best_fitness_hybrid)
   RETURN final_solution
END FUNCTION
END
```

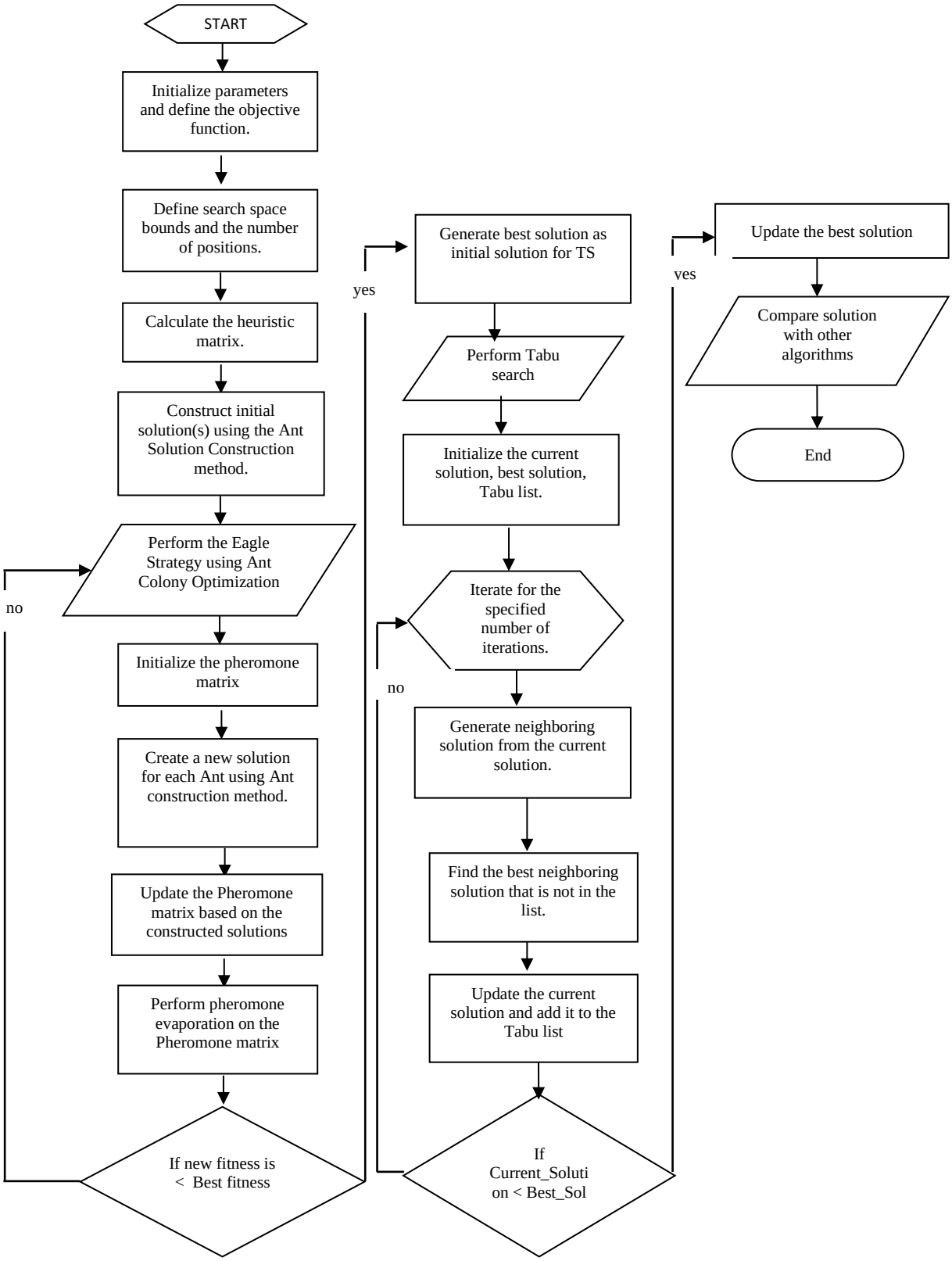

Fig. 1 Flow diagram of enhanced ACOTS_EAGLEMOD approach for the virtual machine placement

5. Simulation Evaluation

The Python 3.8 simulator, which runs on a PC with an Intel Core i7 processor, 8 GB RAM, 3.4 GHz CPU, and Windows 10 operating system, is used to evaluate the algorithms ACOTS_EAGLEMOD, ACO, TS, and EAGLEMOD for a variety of workloads, virtual machines, and physical machines.

Table 2 shows the simulation environment for the experiments.

Table 2. Configuration parameters for virtual environment

Types	Parameters	Values
Data Centre	Number of Data Center	1
	Arch	x86
	OS	Linux
	VM Monitor	Xen
Virtual Machines	Processor speed	9726MIPS
	Memory	0.5 GB
	Bandwidth	1 GB/s
	Number of VMs	200 to 1200
	VM Monitor	Xen
Host		
	Storage	4.0 TB
	RAM	16.0 GB
	Bandwidth	15GB/s
Task	Task Ranges	100 to 2000
	Length of the Task	100000
	Size of a File under Consideration	300 MB
	Amount of energy consumption (Watt) in 100% CPU utilization	[100–1000]

5.1. Power Consumption Performance

Based on the comparative analysis shown in Figure 2, it appears that in the presented research or study, the ACOTS (ACO and Tabu Search) algorithm with Eagle Strategy consistently exhibits lower power consumption than other algorithms as the number of VM requests increases.

The improved performance of ACOTS_EMOD is highlighted by the updated data, which makes it the best algorithm for power-efficient virtual machine management, particularly in settings with varying workloads. The shortcomings in scalability and energy optimization are highlighted by the notable power increase for ACO and TS at higher VM counts.

Although ACOTS and EagleMOD these algorithms show a moderate level of efficiency, ACOTS_EMOD performs better, particularly when workloads are heavy.

This suggests that the ACOTS with Eagle Strategy algorithm is more power-efficient in handling more VM requests.

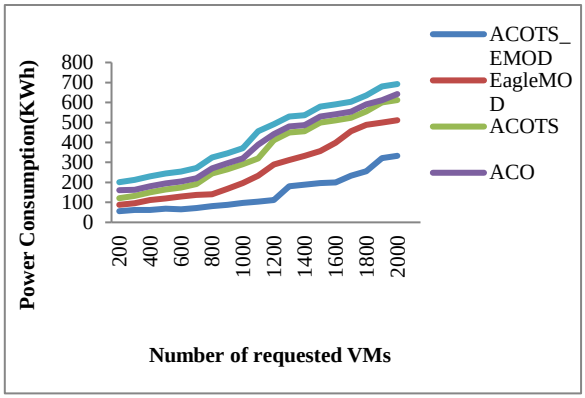

Fig. 2 power consumption vs. the number of requested VMs

5.2. CPU Utilization Performance

CPU utilization performance refers to how effectively the CPU resources are utilized during the execution of an algorithm or system [2]. High CPU utilization indicates that the CPU is being utilized efficiently and effectively, while low CPU utilization may suggest underutilization or idle periods.

In the context of this optimization algorithm, CPU utilization performance is often evaluated in terms of the percentage of time the CPU is actively engaged in executing computations related to the algorithm. Figure 3 shows higher CPU utilization, indicating that the algorithm uses CPU resources efficiently.

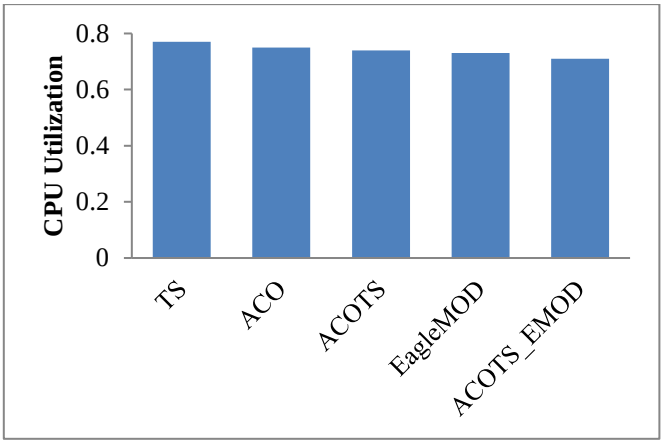

Fig. 3 Average CPU utilization

5.3. Resource Utilization Performance

ACOTS_EMOD effectively maximizes resource usage, as evidenced by consistently high utilization values. This is consistent with its ability to reduce power consumption (as demonstrated in the previous graph), demonstrating its strength as a VM placement algorithm in resource-constrained settings.

Despite maintaining a respectable level of stability, EagleMOD and ACOTS' utilization levels are significantly lower than ACOTS_EMOD's, suggesting that resource allocation could be improved. ACO and TS use inefficient resources, especially at lower workloads, which may result in poor performance in real-world situations where effective virtual machine placement is necessary. ACOTS_EMOD maintains high utilization with little fluctuation as the number of virtual machines (VMs) rises. This suggests strong adaptability to changing or high-demand situations. Resource utilization refers to the efficient and effective usage of available resources during the execution of an algorithm or system. In Figure 4, resource utilization typically refers to how effectively the algorithms utilize server resource CPU to solve the virtual machine placement problem.

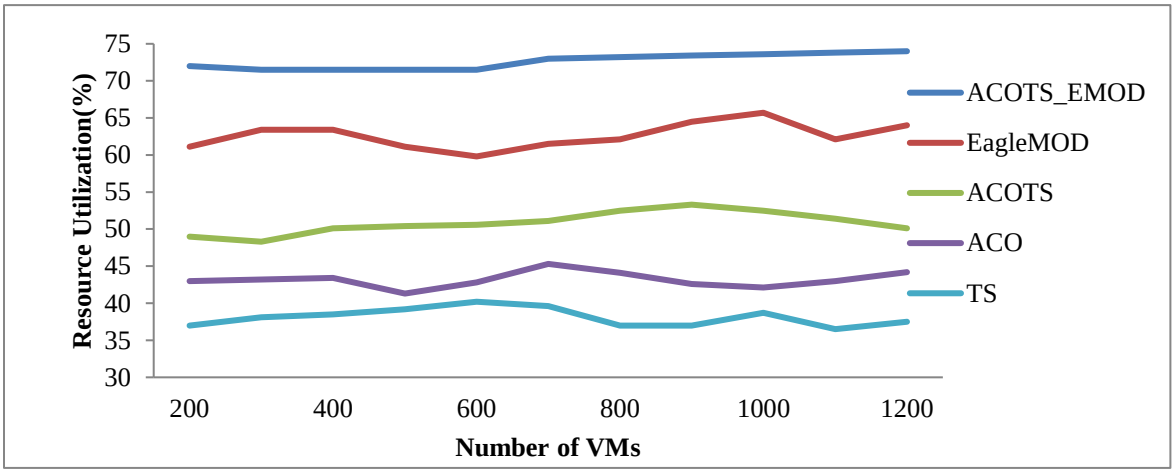

Fig. 4 Average resource utilization

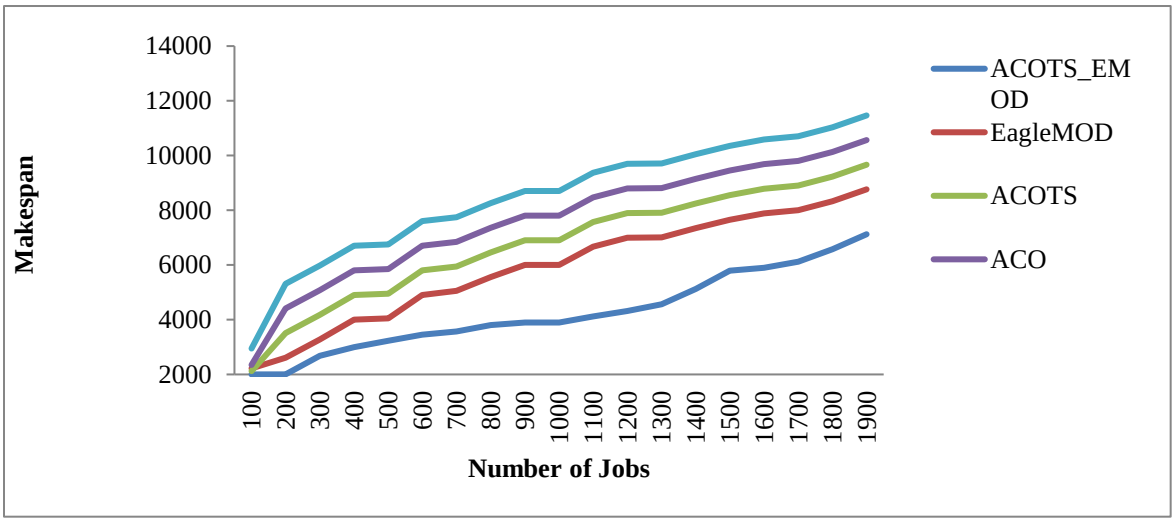

Fig. 5 Comparative analysis of the makespan (sec) versus the number of jobs

5.4. Makespan

The makespan measures the total time required to complete the execution of a set of tasks or operations, as shown in Figure 5. In the context of an algorithm, the makespan refers to the time taken to execute the entire algorithm, from the start to the completion. Figure 5 shows that the projected method decreases the makespan compared to the existing algorithm.

6. Conclusion and Future Work

This paper uses a hybrid metaheuristic placement algorithm to present the best solution for VM placements in cloud environments. The proposed method aims to maximize VMP efficiency while simultaneously reducing resource and energy waste. Using the hybrid metaheuristic algorithm ACOTS (Ant Colony Optimization with Tabu Search) enhancing with the feature of Eagle Strategy, this paper offers the best solution for VM placements in cloud environments. The suggested strategy addresses the problem of local optimum. Comparisons are made between the improved algorithm and Ant Colony Optimization, Tabu search, ACOTS, and EAGLE MOD. Based on a fitness function that considers the total CPU Utilization of the

requested VMs, makespan, power consumption, and resource waste in the cloud DC, the best VMP solution was determined. The enhanced ACOTS with the Eagle Strategy algorithm is more effective than the other algorithms mentioned, according to the experimental results of the simulation evaluations. This algorithm combines the Ant Colony Optimization approach with the Tabu Search approach to find an optimal or near-optimal solution for the given problem. The Eagle Strategy guides the exploration and exploitation of the search space. At the same time, the Tabu Search helps to overcome local optima by searching in the neighbourhood of the current solution. In future, Strategies such as task migration, workload redistribution, Use of Reinforcement Learning (RL) to train the eagle strategy to improve decision-making over time can be explored.

References

[1] Richa, and Deepika Kurkreja, "Optimizing Task Scheduling for Energy Aware Networks," *14th International Conference on Cloud Computing, Data Science & Engineering*, Noida, India, pp. 297-302, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[2] Krishna Gopal Dhal et al., "Eagle Strategy in Nature-Inspired Optimization: Theory, Analysis, Applications, and Comparative Study," *Archives of Computational Methods in Engineering*, vol. 31, pp. 1213-1232, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[3] Peter Mell, and Timothy Grance, "NIST SP 800-145, The NIST Definition of Cloud Computing," *National Institute of Standards and Technology Gaithersburg*, pp. 1-7, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[4] Patricia T. Endo et al., "High Availability in Clouds: Systematic Review and Research Challenges," *Journal of Cloud Computing*, vol. 5, pp. 1-15, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[5] Qi Zhang, Lu Cheng, and Raouf Boutaba, "Cloud Computing: State-of-the-Art and Research Challenges," *Journal of Internet Services and Applications*, vol. 1, pp. 7-18, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[6] Ammar Al-Moalmi et al., "Optimal Virtual Machine Placement Based on Grey Wolf Optimization," *Electronics*, vol. 8, no. 3, pp. 1-22, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[7] Zoltán Ádám Mann, "Allocation of Virtual Machines in Cloud Data Centers-A Survey of Problem Models and Optimization Algorithms," *ACM Computing Surveys*, vol. 48, no. 1, pp. 1-34, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[8] Montassar Riahi, and Saoussen Krichen, "A Multi-Objective Decision Support Framework for Virtual Machine Placement in Cloud Data Centers: A Real Case Study," *The Journal of Supercomputing*, vol. 74, pp. 2984-3015, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[9] Fabio Lopez-Pires, and Benjamin Baran, "Virtual Machine Placement Literature Review," *Arxiv*, pp. 1-11, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[10] Nasim Donyagard Vahed, Mostafa Ghobaei-Arani, and Alireza Souri, "Multiobjective Virtual Machine Placement Mechanisms Using Nature-Inspired Metaheuristic Algorithms in Cloud Environments: A Comprehensive Review," *International Journal of Communication Systems*, vol. 32, no. 14, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[11] Bingdong Li et al., "Many-Objective Evolutionary Algorithms: A Survey," *ACM Computing Surveys*, vol. 48, no. 1, pp. 1-35, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[12] Fabio López-Pires, and Benjamín Barán, "Many-Objective Virtual Machine Placement," *Journal of Grid Computing*, vol. 15, pp. 161-176, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[13] Alessandro Mattiussi, Michele Rosano, and Patrizia Simeoni, "A Decision Support System for Sustainable Energy Supply Combining Multi-Objective and Multi-Attribute Analysis: An Australian Case Study," *Decision Support Systems*, vol. 57, pp. 150-159, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[14] Zoha Usmani, and Shailendra Singh, "A Survey of Virtual Machine Placement Techniques in a Cloud Data Center," *Procedia Computer Science*, vol. 78, pp. 491-498, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[15] Kumar Nishant et al., "Load Balancing of Nodes in Cloud Using Ant Colony Optimization," *UKSim 14th International Conference on Computer Modelling and Simulation*, Cambridge, UK, pp. 3-8, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[16] Medhat A. Tawfeek et al., "Virtual Machine Placement Based on Ant Colony Optimization for Minimizing Resource Wastage," *Advanced Machine Learning Technologies and Applications, Communications in Computer and Information Science*, vol. 488, pp. 153-164, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[17] Xiao-Fang Liu et al., "An Energy Efficient Ant Colony System for Virtual Machine Placement in Cloud Computing," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 113-128, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[18] Keng-Mao Cho et al., "A Hybrid Meta-Heuristic Algorithm for VM Scheduling with Load Balancing in Cloud Computing," *Neural Computing and Applications*, vol. 26, pp. 1297-1309, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[19] Qinghua Zheng et al., "Multi-objective Optimization Algorithm Based on BBO for Virtual Machine Consolidation Problem," *IEEE 21st International Conference on Parallel and Distributed Systems*, Melbourne, VIC, Australia, pp. 414-421, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[20] Da-Ming Zhao, Jian-Tao Zhou, and Keqin Li, "An Energy-Aware Algorithm for Virtual Machine Placement in Cloud Computing," *IEEE Access*, vol. 7, pp. 55659-55668, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[21] Esha Barlaskar, Yumnam Jayanta Singh, and Biju Issac, "Enhanced Cuckoo Search Algorithm for Virtual Machine Placement in Cloud Data Centres," *International Journal of Grid and Utility Computing*, vol. 9, no. 1, pp. 1-17, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[22] Weichao Ding et al., "DFA-VMP: An Efficient and Secure Virtual Machine Placement Strategy under Cloud Environment," *Peer-to-Peer Networking and Applications*, vol. 11, pp. 318-333, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[23] Sara Mejahed, and M. Elshrkawey, "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*, vol. 8, pp. 1-27, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[24] Masoud Hashemi et al., "A Multi-Objective Method for Virtual Machines Allocation in Cloud Data Centres Using an Improved Grey Wolf Optimization Algorithm," *IET Communications*, vol. 15, no. 18, pp. 2342-2353, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[25] Chaoqiang Jin et al., "A Review of Power Consumption Models of Servers in Data Centers," *Applied Energy*, vol. 265, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[26] A.S. Abohamama, and Eslam Hamouda, "A Hybrid Energy-Aware Virtual Machine Placement Algorithm for Cloud Environments," *Expert Systems with Applications*, vol. 150, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[27] Yongquan Zhou, Ying Ling, and Qifang Luo, "Lévy Flight Trajectory-Based Whale Optimization Algorithm for Engineering Optimization," *Engineering Computations*, vol. 35, no. 7, pp. 2406-2428, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[28] Xin-She Yang, and Suash Deb, "Eagle Strategy Using Lévy Walk and Firefly Algorithms for Stochastic Optimization," *Nature Inspired Cooperative Strategies for Optimization, Studies in Computational Intelligence*, vol. 284, pp. 101-111, 2010. [CrossRef] [Google Scholar] [Publisher Link]