ISSN: 0067-2904

Load Balancing on Virtual Machines Using Levy Flight Combined with Gray Wolf Optimization Algorithm

Mohammed Khawwam Ahmed[1 *], Salah Awad Aliesawi [2] and Omar Younis[3] Abdulhammed

[1]*Technical Institute of Baqubah, Middle Technical University, Iraq*
[2] *College of Computer Sciences and Information Technology, University of Anbar, Iraq*
[3] *College of Science, University of Garmian, Iraq*

Received: 29/4/2024 Accepted: 24/9/2024 Published: 30/10/2025

Abstract

In a cloud computing environment, virtual machines must be allocated wisely among queued resources waiting for a specific service. In this case, the optimal load balancing on those virtual machines must be done using optimization methods. Because the distribution of processes on the VMs is considered one of the NP-Hard optimization problems, it is necessary to use optimization methods. This paper proposes a hybridized Gray Wolf Optimizer (GWO) and Levy Flight (LF) search approach. The proposed method is called Levy Flight-GWO-Virtual Machine Load Balancing (LF-GWO-VM-LB). Despite the success of the standard GWO alone as a load-balancing method, it needs to improve its explorative capability and premature convergence. Levy flight is an ideal way to increase the exploration capability of the GWO as it introduces randomness and unpredictability, enhancing the diversity of the search process. This increased diversity can prevent the algorithm from getting stuck in suboptimal solutions. The algorithm's performance was measured using the makespan and throughput. However, the last is only used as one of the objectives used in the optimization process. Both makespan and throughput results showed the outperformance of the LF-GWO-VM-LB compared to the other state-of-the-art methods, i.e., QMPSO, MPSO, and the Q-learning methods.

Keywords: Cloud Computing, Load Balancing, Levy Flight, Makespan, Optimization, Virtual Machines,

موازنة الأحمال على الأجهزة الافتراضية باستعمال طيران ليفي مدمجة مع خوارزمية التحسين الذئب الرمادي

محمد خوام احمد[1]*, صلاح عواد العيساوي[2], عمر يونس عبدالحميد[3]

[1] المعهد التقني بعقوبة, الجامعة التقنية الوسطى, ديالى, العراق
[2]كلية علوم الحاسوب وتكنلوجيا المعلومات, جامعة الانبار, الانبار, العراق
[3] كلية العلوم, جامعة كارميان, سليمانية, العراق

الخلاصه

في بيئة الحوسبة السحابية، يجب تخصيص الأجهزة الافتراضية بحكمة بين الموارد الموضوعة في قائمة الانتظار في انتظار خدمة معينة. في هذه الحالة، يجب إجراء موازنة التحميل الأمثل على تلك الأجهزة الافتراضية

*Email: moh20c1005@uoanbar.edu.iq

باستعمال طرق التحسين. نظرًا لأن توزيع العمليات على الأجهزة الافتراضية يعتبر أحد مشكلات تحسين NP-Hard، فمن الضروري استعمال طرق التحسين. تقترح هذه الورقة طريقة هجينة للبحث عن Gray Wolf Optimizer (GWO) و Levy Flight (LF). الطريقة المقترحة تسمى Hybrid GWO Load Balancing (LF-GWO-VM-LB). على الرغم من نجاح GWO القياسي وحده كطريقة لموازنة التحميل، إلا أنه يحتاج إلى تحسين قدرته الاستكشافية والتقارب المبكر. تعد رحلة Levy طريقة مثالية لزيادة قدرة الاستكشاف في GWO لأنها تقدم العشوائية وعدم القدرة على التنبؤ، مما يعزز تنوع عملية البحث. هذا التنوع المتزايد يمكن أن يمنع الخوارزمية من التعثر في الحلول دون المستوى الأمثل. تم قياس أداء الخوارزمية باستعمال المدى والإنتاجية واستهلاك الطاقة. ومع ذلك، يتم توظيف الأخير فقط كأحد الأهداف التي تم استعمالها في عملية التحسين. أظهرت كل من نتائج التصنيع والإنتاجية تفوق أداء LF-GWO-VM-LB بالمقارنة مع الطرق الحديثة الأخرى، مثل QMPSO، وMPSO، وطرق Q-learning

1. Introduction

When customers' access to diverse administrations and assets can be transformed using cloud computing, clients can rapidly extend their capabilities using the cloud without investing in licensing or foundational hardware resources [1]. Over time, cloud computing has become popular among clients seeking administration. Several task management strategies have been proposed to enhance system performance by optimizing resource utilization. However, symmetry-based strategies have yet to receive much consideration. Given the widespread adoption of cloud computing technology, it is anticipated that symmetry will be employed to enhance cloud computing performance. Cloud computing has the advantages of flexibility, constant quality, and lower maintenance because the outside organization maintains the administration and products. These are the hypotheses that explain why the cloud is so prominent. Simplified terms for these cloud administrations include IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service) [2–4].

Cloud computing is a virtual environment that provides users with services that exceed the physically available resources by virtualizing such assets using specific algorithms. Thus, the clients can utilize the cloud environment to get rapid processing without investing in foundations or licensing [1]. The cloud environment allows for the simultaneous execution of multiple tasks in the background. Besides board costs, server farms might be retained to check during the innovative utilization of board automation and virtualization. Making an asset reservation in a cloud environment goes against the cloud's virtualization concept, resulting in an environment that appears limited. Ad-lobbing the resource usage rates has the purpose of arriving at a complete reservation calculation. The problem of diversified resource allocation can be considered a combinatorial optimization problem, which can also be represented as an NP-Hard complete issue. Many grid-and-cloud techniques have been designed to resolve this problem [2].

Meta-heuristics and optimization have long been successful in solving such types of NP-hard problems. Many possible solutions make it easier for the traditional search methods to find the best solution among all other solutions. Looking for the optimality in matching. It will likely accomplish better results by utilizing metaheuristics and optimization methods that can extend the search space area [5-7]. As for the single-solution-based optimizations, these methods can intensify the optimization of a single solution to reach optimality. However, using population-based metaheuristic methods for identifying the optimal solution requires less effort than single-based methods. Thus, such methods are more time-saving due to their parallel nature, which optimizes many solutions simultaneously. The metaheuristics capability of discovering the optimal regions within the search space made it more desirable with NP-Hard optimization problems such as the load balancing problem. In this area, many researchers have investigated

the utilization of iterative algorithms that aim to optimize tasks on virtual machines. Gray Wolf Optimization and Levy Flight Optimization are two examples of these methods.

Swarm intelligence (SI) is an extensive set of optimization algorithms that prioritize the social links and interactions among swarm members as they seek and pursue food sources [8]. Multiple SI algorithms have been developed and proposed in recent years. Gray Wolf Optimization (GWO) is a popular and commonly used approach. The GWO algorithm was inspired by the gray wolf's instinctive hunting behavior, which focuses on finding the most efficient method to chase prey. This resulted in a favorable equilibrium between exploration and exploitation. One drawback is its low performance in global search [9]. An enhanced version of gray wolf optimization, improved gray wolf optimization (IGWO), was introduced by including the Dimensional Learning-based Hunting (DLH) search technique to overcome this restriction. GWO excels at multidimensional functions, but it could be more effective at unidimensional functions. Therefore, an approach called Levy flight-based enhanced gray wolf optimization (LF-IGWO) is proposed in [4] to address this one-dimensional problem.

This current research aims to contribute to optimal task load balancing using an LF-GWO-VM-LB method. The focal point of this method is the amalgamation of the Levy Flight Optimizer (LFO) with the Gray Wolf Optimizer (GWO), which presents a synergistic strategy to improve task allocation on virtual machines in a computing environment. The primary motivation for this hybridization stems from recognizing the complementary strengths of the Levy Flight Optimizer and the Gray Wolf Optimizer. The Levy Flight Optimizer is renowned for its capacity to facilitate long-range exploration in solution spaces, leveraging Levy flights that exhibit scale-free characteristics. On the other hand, the Gray Wolf Optimizer excels at exploiting promising areas, drawing inspiration from the social hierarchy observed in gray wolf packs.

The rest of this paper will be arranged as follows: In Section two, the literature review is explained, while in Section three, the methodology will be highlighted in terms of spotting the light on the main algorithmic components of the proposed task distribution methods. Section four explains the test results of the proposed method, while Section five provides conclusions and recommendations for future enhancements.

2. Related Work

The literature presents the most recent academic work in this area. In [8], the authors proposed a Whale Optimization Algorithm (WOA) for optimizing the load balancing on the virtual machines. The whale optimization technique could prevent unexpected traffic and the regular operations of Internet websites by providing a suitable plan for load balancing of the user requests bouncing between the server machines and minimizing the total feedback time. Thus, the authors could prevent DDoS attacks by applying this optimization technique. It is necessary to be advised that the utilization of HA-Proxy for preventing DDoS is insufficient, and depending on the attack type, many layers of hardware and software security measures are prerequisites to be considered.

[9] presented a task scheduling method for the cloud computing environment motivated by the ACO-based load balancing (LBACO) algorithm. This method reduces the makespan of tasks by balancing the load of the whole system. The main contribution of our work is to balance the entire system load while minimizing the makespan of a given task set. The new scheduling strategy was simulated using the CloudSim toolkit package. The experiment results showed that the suggested LBACO algorithm exceeded FCFS (First Come, First Serve) and the primary ACO (Ant Colony Optimization). The authors presented a job scheduler for scheduling the

tasks in the cloud environment. This method was based on the Ant Colony Optimization (ACO), which is named a Loads Balancing ACO (LB-ACO). The technique minimizes the makespan by looking for the best task distribution on the entire system. The authors aimed to achieve system balance with a minimal makespan. The CloudSim toolkit was used for the simulation of this method. The experimental results showed the superiority of the LB-ACO method when compared to the traditional ACO and the first-come, first-served FCFS methods.

In [10], the authors suggested a task distribution method that fairly spread the tasks among the virtual machines. The fuzzy logic has been used to reduce energy and costs in Geo-Distribute Multiple Data Centers. The utilization of fuzzy reasoning made this process to be more efficient. The suggested system revealed good results when compared to the other methods. The authors noticed that this fuzzy-based method had minimized the costs dramatically with no previous knowledge about the existence of renewable upcoming power prices and loads burdens.

In [11], this paper encompasses two pivotal components: firstly, the establishment and integration of an innovative healthcare system based on the Internet of Things (IoT), and secondly, the adaptation resolution of load balancing in cloud computing via the application of the Sparrow Search Algorithm (SSA). The SSA, an intelligent algorithm, plays a central role in the dynamic selection of virtual machines (VMs) by assessing their fitness values. This process guarantees the optimal allocation of diversity of tasks, where priority ranks are determined by Instruction Millions (IM) associated with each task. High IM processes are strategically mapped with high-valued fitness VMs, improving the system's efficiency. The results show the method's capability to minimize latency and packet losses while maximizing throughput within healthcare systems.

Moreover, the robustness, efficiency, and success of the SSA appear when compared to other methods. The method reduces both makespan time and total processing time while ensuring a well-balanced distribution of workloads among VMs. This dual-faceted approach addresses the intricacies of healthcare system optimization and showcases the adaptability and efficacy of the SSA in the broader context of cloud computing load balancing.

The authors of [12] have suggested a novel edition of Genetic Algorithm (GA) in which individualized chromosomes have a set of labels referring to each VM.
Hybridized optimization methods have the potential to combine the advantages of two methods to gain a double benefit in cloud computing. For instance, in [13], the improved Wild Horse Optimization, also known as IWHO, is utilized to address high-cost consumption, task scheduling time management, and high virtual machine loads in cloud environment tasks. In this context, the primary factors influencing cloud computing task scheduling and the distribution model include virtual machines, time, and cost. Second, feasible planning for each member maps to cloud computing task scheduling is to discover the best member that the best possible plan; the authors used the inertial weighting technique for the IWHO optimization method for improving the local search ability and efficiently preventing the algorithm from stagnation and converge pre-maturely and to find better the best member for more improvement of the effectiveness of jobs scheduling in the cloud environment. The Improved Wild Horse Optimization was combined with the Levy Flight Algorithm for Task Scheduling in Cloud Computing (IWHOLF-TSC) approach. As a result, the levy flight improved the traditional WHO algorithm's performance.

Thus, the levy search is one of the best optimization methods that can enhance the performance of another optimization algorithm. In [14], the authors proposed a hybrid meta-heuristic technique based on particle swarm optimization (PSO) for promptly solving the problem of polynomials. The suggested method, known as LMPSO, successfully solved the service cache problem. This proposed method added lévy flight movements for particle update to improve the diversification of particles and avoid the PSO being stuck in the local optima. In [15], the authors use multi-tenant and load-balancing technologies in cloud-based digital resource sharing, specifically for academic and digital libraries. A suggestion is presented to enhance the current user service model by implementing private cloud storage in many sectors, such as the medical and financial fields, as a novel approach to digital resource sharing. This paper provides an overview of cloud computing, its potential uses, and optimizing digital data for web-based services.

The authors of [16] proposed a fog-based spider web algorithm (FSWA), a metaheuristic technique that aims to reduce the delay time (DT). It also enhances the response time (RT) during the workflow of tasks that are distributed across various edges within the cloud.
In [17], the authors studied the concepts of the environment and the architecture of fog computing, besides the evaluation measures that impact the asset assignments in the environment.

In [18], the authors summarized cloud computing and the aspects that this technology is used in. Thus, this research gives the latest advancement in the area of cloud technology, where the most recent inventions are reviewed and compared.

3. Standard GWO

The basic GWO consists of three main motions: encircle, chase, and attack. The GWO has a number of wolves whose best represent the leaders of the group, as can be represented in Figure 1, where the best three wolves are represented according to their hierarchical levels [19]. Those three movements can be represented as follows:

3.1 Encircle

During the hunting process, gray wolves will encircle their target. The encircling can be modeled by updating the locations of the wolves that are circling the goal, which is the prey. This shows the ideal solution that is discovered through this process. When seen from a mathematical perspective, the model can be expressed as illustrated in the equations that follow:

$$\vec{D} = \left|\vec{C}.\vec{X}_P(t) - \vec{X}(t)\right| \quad (1)$$

$$\vec{X}(t+1) = \vec{X}_P(t) - \vec{A}.\vec{D} \quad (2)$$

Here, X_P is the position of the prey, $X_{(t)}$ is the position of the gray wolf, $\vec{A}$ and $\vec{C}$ are coefficient vectors, and $\vec{D}$ is the distance between the wolf and the prey.
Both vectors $\vec{C}$, $\vec{A}$ are timely charged as shown in the following equations [18-20]:

$$\vec{C} = 2 \times r_2 \quad (3)$$

$$\vec{A} = 2\vec{a}.r_1 - \vec{a} \quad (4)$$

C, which is a coefficient that lies within the range [0,2], while r_2 is the randomized number in the interval of 0 and 1. In GWO, C is a coefficient vector that is used to control the influence of the prey's position on the movement of the wolves. This range is essential for maintaining the balance between exploration (searching new areas) and exploitation (focusing on promising areas).

$$a = 2 - 2 * \frac{t}{T} \quad (5)$$

Where a is a constant that balances the convergence, which exists between 2 and 0, which iteratively decreases overtime. The random number r_1 can be produced in between [0,1], similarly to r_2. As for t it represents the current iteration, while T is the maximum number of iterations. Here, a is a coefficient that decreases linearly from 2 to 0 over the course of the iterations. This decrease controls the balance between exploration and exploitation. At the beginning of the optimization, a is large, allowing for more exploration (searching a wide area). As a decrease, the focus shifts towards exploitation, allowing the wolves to fine-tune their positions around the best solutions found.

3.2 Chase

The act of the wolf running after its prey is referred to as "chasing." The wolf's location will be changed according to the whereabouts of the three best wolves, which are the alpha, beta, and delta. It is, therefore, the responsibility of the three most capable wolves in the group to provide direction to the other wolves. The chasing mechanism allows for the possibility of approaching optimality, which is the point at which convergence occurs.

3.3 Attack

The attack ensues when the gray wolves approach the prey within a certain proximity. The distance between the wolf and the prey will determine the convergence. During the search, the magnitude of vector A decreases, resulting in fewer oscillations and a more precise convergence towards the prey, which represents the optimal solution. The optimal solution can achieve the best fitness value. This stage signifies the process of exploitation, during which the algorithm concentrates on enhancing and perfecting the solutions.

During the iterations, the value of a declines in a linear manner from 2 to 0, while r represents a random vector within the range of 0 to 1. When the absolute value of A is greater than 1, the wolves go in the opposite direction from their prey. This behavior imitates exploration, enabling the algorithm to explore novel solutions in various parts of the search space. This aids in circumventing local minima and efficiently exploring the global search space.

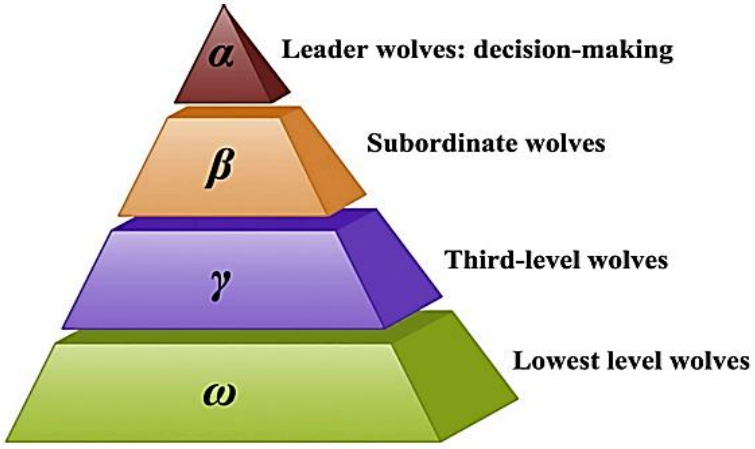

Figure 1: Hierarchical levels of gray wolves [19]

The algorithm of the standard GWO is explained in algorithm 1.

Algorithm 1. Pseudo code of Standard Gray Wolf Optimization
Input: population of gray wolves Xi (i = 1, 2, ... , n), a, A, and C Begin **Step 1:** Initialize the population of gray wolves Xi (i = 1, 2, ... , n)

Step 2: Initialize a, A, and C //three best wolves in the group
Step 3: Calculate the fitness values of search agents and grade them. (Xα= the best solution in the search agent, Xβ= the second-best solution in the search agent, and Xδ= the third best solution in the search agent)
Step 4: t= 0
Step 5: While (t < Max number of iterations)
For each search agent
Update the position of the current search agent by Equation (4)
End for
Update a, A, and C
Calculate the fitness values of all search agents and grade them
Update the positions of Xα, Xβ, and Xδ
t= t+1
End while
End

4. Proposed Method

The method begins by initializing the Gary Wolf individuals in terms, followed by modeling the attack, encircle, and chase motions using equations from 1 to 5. Table 1 summarizes all of the symbols used in the proposed method's equations.

Table 1: description of symbols and notations mentioned in the equations of the proposed method.

Notation	Description
step	The Levy flight distribution is utilized to calculate the step size, which is crucial in determining the magnitude of the move in the solution space.
u	A normal distribution is a random variable used in the Levy flight calculation.
v	The Levy flight calculation utilizes a random variable drawn from a normal distribution.
β (beta)	The Levy distribution parameter, typically in the range of 0-2, influences the step size distribution, with a smaller β resulting in larger steps.
step-size	The calculated step is scaled to adjust the solution's movement.
s	The current position of the solution in the search space.
rand(size(s))	A random number generator generates values within the current position's size.
Flagub	A binary flag indicates if the current position surpasses the upper bound (*ub*) of the search space.
Flaglb	A binary flag indicates if the current position is below the lower bound (*lb*) of the search space.
ub	The upper bound of the search space.
lb	The lower bound of the search space.
position	The solution's final updated position in the search space, considering boundary conditions.
μ	Represents the location parameter, which is the distribution's average
σ	Represents the scaling parameter, which is the distribution's standard deviation.
g	The tail index parameter that controls the distribution shape.
C, D	represent the distance vectors between the i^{th} wolf and the j^{th} wolf.
$x_i(t)$	The current location of the i^{th} member (wolf) at time t.
$x_i(t+1)$	The updated location of the i^{th} member (wolf) at the next time unit t+1.

Presume $\propto, \beta, and\ \delta$ are the best solutions in the algorithm because of the knowledge of the optimal location because of the strength in the pack of them. Thus, other wolves in the group attempt to modify their locations as follows:

$$x(t+1) = \frac{1}{3}X_1 + \frac{1}{3}X_2 + \frac{1}{3}X_3 \quad (6)$$

Where x_1, x_2 and x_3 can be calculated as in the following linear system:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_\propto(t) \\ x_\beta(t) \\ x_\delta(t) \end{pmatrix} - \begin{pmatrix} A_1.D_\propto \\ A_2.D_\beta \\ A_3.D_\delta \end{pmatrix} \quad (7)$$

In the GWO algorithm:
• α is the leader of the pack, representing the best solution found so far.
• β is the second-best solution and assists α in leading the group.
• δ is the third-best solution and also supports α and β.
Where, D_α, D_β, and D_δ are computed as shown in the following:

$$|x1.x_\alpha - x| \quad (8)$$
$$|x2.x_\alpha - x| \quad (9)$$
$$|x3.x_\alpha - x| \quad (10)$$

After setting up the wolves' population, it becomes necessary to enhance the algorithm's performance using levy optimization, characterized by its predictability and randomness. As a kind of unplanned walk, LF is a step-size-based method; it is considered a probability proportional to the heavy tail. Thus, the distribution has a higher probability of generating broad jumps. The broad jumps appear more frequently than they are in a normal distribution [16]. Figure 2 represents the levy movement. The Levy distribution power law tail and its probability density function can be written as follows:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\gamma}{2\sigma^2}|x-\mu|\right) \frac{\gamma}{|x-\mu|^{1+\gamma}} \quad (11)$$

The tail index coefficient g controls the shape of the distributions. The distribution has infinite variance when g is between 0 and 2 [17]. This leads to the fact that distribution variance is undefined. When g is greater than 2, the distribution has a finite variance; when g is less than or equal to 1, the distribution has an infinite mean. The random positions are selected and updated to enhance the exploration aspect. The equation used to find the wolf's location is expressed as below:

$$step = \frac{u.}{abs(v)^{\frac{1}{\beta}}} \quad (12)$$
$$step - size = 1 \times 10^{-3} \times step \times s \quad (13)$$
$$s = s + step_{size} \times rand(size(s)) \quad (14)$$
$$position = s \times \left(\sim(Flagub + Flaglb)\right) + ub \times (Flagub + lb) \times Flaglb \quad (15)$$

Using equations from 12-15, the updated wolf's position is computed and assigned to the wolf individuals. The update of the alpha wolf locations can be calculated as seen in equation (16) [21-25].

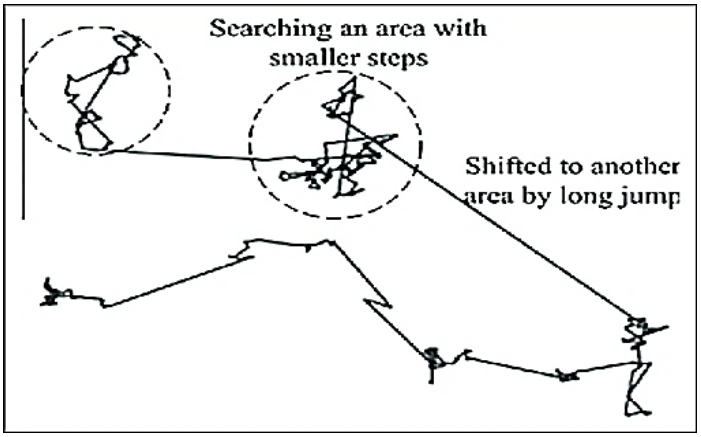

Figure 2: levy flight search [26]

By utilizing the varying distance vector (D) of the $\omega, \delta, and\ \beta$ for the following position. In this context, $x_i(t)$is the present location of the ith member at the time unit (t), while $x_i(t+1)$ is the modified location at the time unit (t+1). Regarding, $A_j D_j$, the first represents the scaling coefficient for the jth member, while the second is the distance vector running from the ith and jth wolves. A_j Can be modified as in equation (17).

$$A_j = \frac{2\times(1-u)\times a-u}{2} \quad (16)$$

Where u is a uniformly distributed random number ranging within the interval [0, 1], a is the iteration counter, and j refers to the $\omega, \delta, and\ \beta$ wolves

D_j Can be calculated as can be seen in equation (17).

$$D_j = \left|C_j \times x_j - x_i\right| \quad (17)$$

Every iteration generates a random vector where Cj represents each wolf within the [0, 1] range. The algorithm concludes when a stopping condition is satisfied, which could be reaching a maximum iteration count or attaining a minimal improvement level in the objective function [19].

Notably, the proposed method is explained in the flowchart in Figure. 3.

Algorithm 2. Pseudo code of LF-GWO-VM-LB proposed method

Input: population of gray wolves Xi (i = 1, 2, ... , n), a, A, and C
Begin
Step 1: Initialize the population of gray wolves Xi (i = 1, 2, ... , n)
Step 2: Initialize a, A, and C //three best wolves in the group
Step 3: Calculate the fitness values of search agents and grade them. (Xα= the best solution in the search agent, Xβ= the second-best solution in the search agent, and Xδ= the third best solution in the search agent)
Step 4: t= 0
Step 5: While (t < Max number of iterations)
For each search agent
New_position = Current_Position∗ Levy (Xi)′
for each search agent do
Update the position of the current search individual
by Eq. (9)
Update the position of the current search agent by Eq. (15)
End for
Update a, A, and C
Calculate the fitness values of all search agents and grade them
Update the positions of Xα, Xβ, and Xδ
t= t+1
End while
End

5. Test Results

In this section, we present the test results of the proposed LF-GWO-VM-LB with the modified particle swarm optimization (MPSO), improved Q-learning algorithm named QMPSO, and Q-learning methods available in [27].

5.1 Simulation Environment

The cloud configuration has been carefully constructed using the Python platform, using its flexibility and scalability to produce a customized cloud environment specifically designed for running the intended scripts. This cloud environment is a flexible and adaptable workspace that offers researchers and practitioners the essential computing resources and tools to conduct experiments efficiently and effectively.

Table 1 provides a detailed summary of the simulation environment used in the research. This environment has been meticulously set up to fulfill the precise demands of the research objectives, incorporating a range of vital software components, libraries, and frameworks necessary for carrying out the experimental protocols. Table 1 provides openness and insight into the computational infrastructure supporting the research by presenting essential details such as device specifications, software configurations, and network settings.

The simulation environment includes a variety of resources, including virtual computers, storage solutions, and networking capabilities. These resources are seamlessly integrated to execute complicated simulations and analyses. By utilizing the scalability and flexibility of cloud computing, researchers may easily adjust the amount of resources to match the changing computational needs. This guarantees that the experimental workflow maintains optimal performance and efficiency.

In addition, based on Python, the cloud setup's architecture provides researchers with exceptional flexibility and adaptability. This enables the smooth integration of customized scripts, libraries, and modules specifically designed to meet their research requirements. This flexible and adjustable methodology allows researchers to quickly repeat, investigate new methods, and be creative in their quest for scientific exploration.

The cloud setup based on the Python platform is an advanced infrastructure specifically designed to facilitate cutting-edge research and experimentation. The cloud architecture offers a flexible, interactive, and comprehensive simulation environment. This allows researchers to explore new frontiers of knowledge and make significant progress in their respective domains.

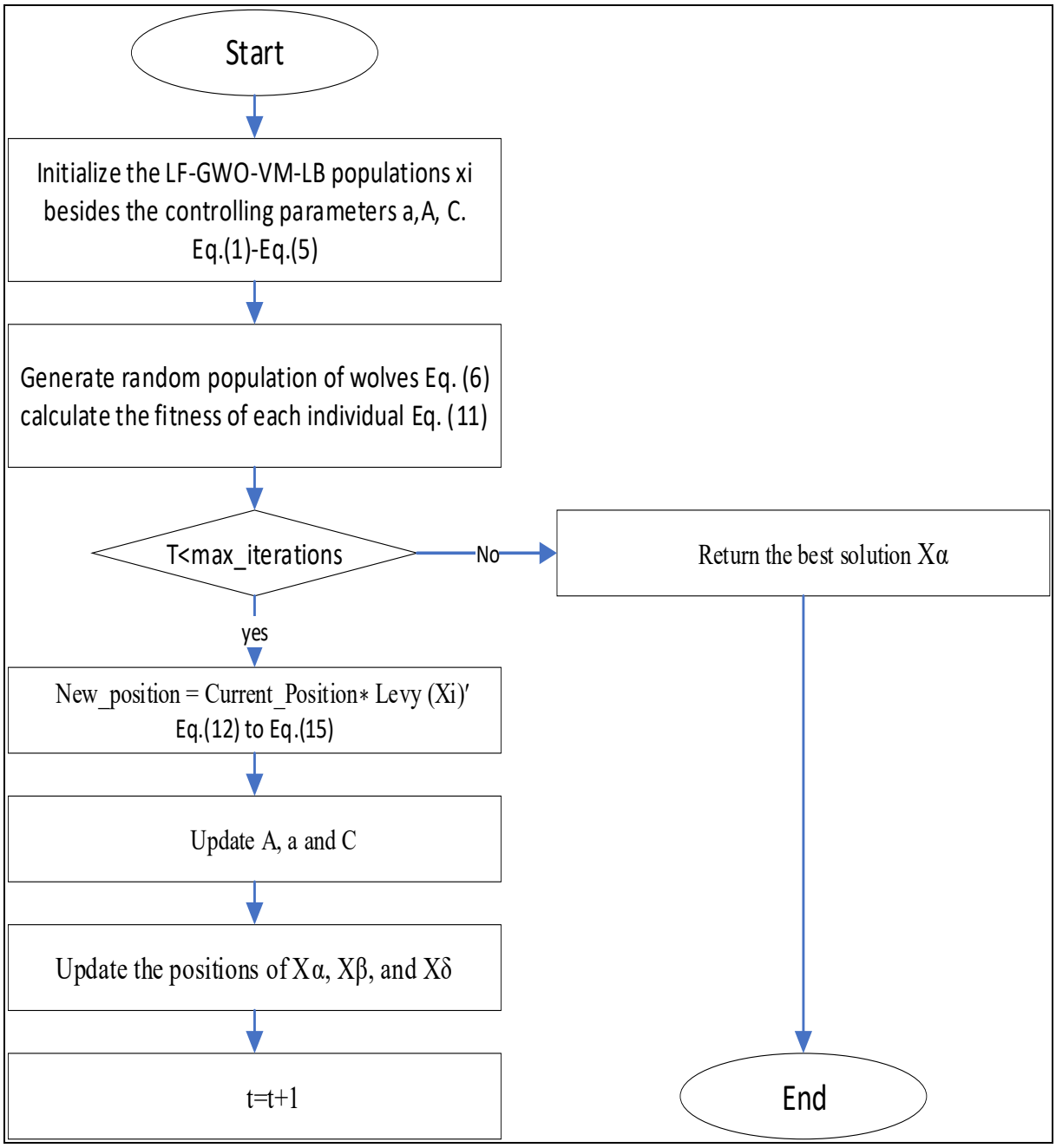

Figure 3: The proposed system flowchart

5.2 Dataset

The supplementary data for the GoCJ [28] dataset (Table 2) is accessible in text and Excel file formats, associated with two dedicated dataset generator items. The generator(s) include an Excel worksheet generator and a Java tool package. Each row provides information about the size of a particular job, quantified in Millions of Instructions (MI) within each text file. The Monte Carlo simulation method [29-30] is applied in crafting the dataset, enabling the generation of any desired number of jobs. This comprehensive approach, encompassing diverse file formats and dataset generation tools, enhances the accessibility and flexibility of the GoCJ dataset for research and analytical purposes.

Table 2: Dataset Description

criteria	description
Subject area	cloud computing
More specific subject area	load balancing
Type of data	10 text files of job size (in MI) are in the dataset, and one text file contains a standard dataset.
Dataset generation	The GoCJ dataset is created using Monte Carlo simulation.

In this test, we run the modified particle swarm optimization (MPSO) and improved Q-learning algorithms, known as QMPSO and Q-Learning, repeatedly with varying task numbers of 20, 30, 40, 60, and 80, while maintaining a virtual machine count of 16. Figure 4 shows the number of tasks on the x-axis besides the algorithm names, while the y-axis shows the performance value achieved by the algorithms in terms of the makespan and throughput measures.

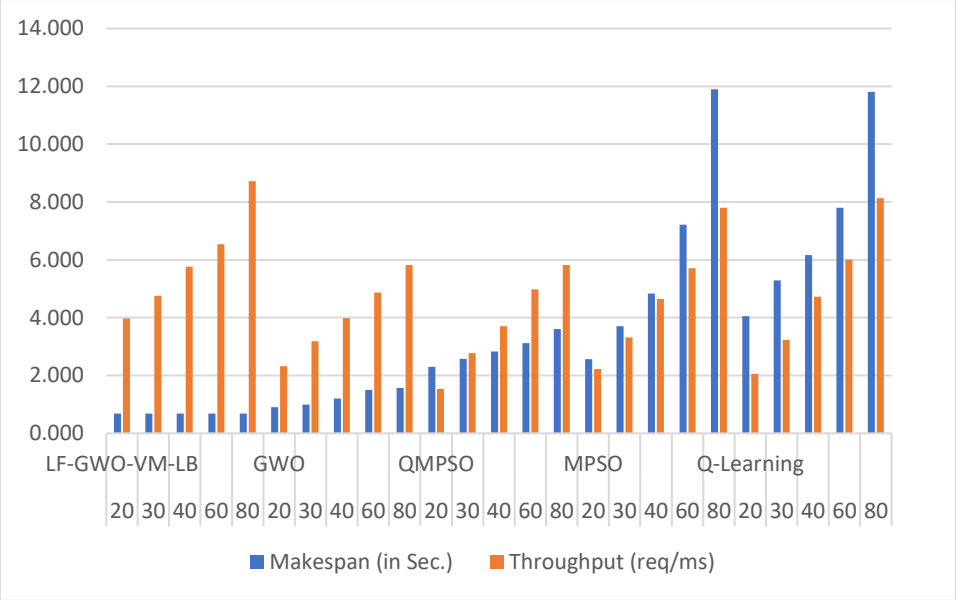

Figure 4: Comparison of the proposed method with other state-of-the-art methods.

The figure shows that the proposed LF-GWO-VM-LB method has achieved a throughput comparable to the QMPSO method. Even though the proposed methods have lower throughput in some cases when compared to the MPSO or the Q-learning technique, they are considered to perform better than those methods. The LF-GWO-VM-LB has consistently outperformed the makespan, achieving a higher throughput in every case. This demonstrates a good balance between execution and task waiting time. Conversely, the performance of the MPSO and Q-learning techniques significantly influences the makespan. In other words, the makespan with all task numbers has a higher percentage than the throughput, indicating a problem with the performance of those methods. Table 3 presents a thorough analysis of the test data, providing in-depth information about the outcomes of the studies. Every row in the table represents a particular test scenario or experimental condition, while the columns separate different metrics, performance indicators, and the associated values collected from the tests. Table 3 provides a detailed explanation of the test results.

Table 3: Test results of (modified Particle swarm optimization (MPSO), improved Q-learning algorithm named QMPSO and Q-Learning

TASK	Algorithm	Makespan (in Sec.)	Throughput (req/ms)
20	LF-GWO-VM-LB	0.686	2.971
30		0.686	2.761
40		0.687	2.761
60		0.684	1.538
80		0.687	2.716
20	QMPSO	2.3	1.53
30		2.58	2.77
40		2.83	3.71
60		3.12	4.98
80		3.61	5.82
20	MPSO	2.57	2.22
30		3.7	3.32
40		4.84	4.65
60		7.22	5.71
80		11.9	7.8
20	Q-Learning	4.05	2.06
30		5.29	3.23
40		6.16	4.72
60		7.8	6.01
80		11.81	8.13

5.3 Makespan vs. Throughput Relationship

The relationship between makespan and throughput is inverse. Any interaction between them degrades the system's performance. Figure 5 illustrates the execution of the LF-GWO-VM-LB method using varying task counts and a minimal makespan percentage, consistently resulting in a gap between the makespan and the throughput. In contrast, the QMPSO method intersects the makespan bar and the throughput line.

5.4 Test of Different VM Numbers

This analysis investigates the impact of increasing the quantity of virtual machines on the test results in terms of makespan and throughput. Upon closer inspection of the table, a trend becomes apparent: the makespan increases according to the number of virtual machines while the throughput decreases simultaneously. Specifically, the table demonstrates a noticeable rise in makespan as the number of virtual machines increases, and a drop in throughput when the task count drops to 200. A deeper look at the table reveals that this tendency continues as the task count is raised to 400 and 800. Both cases show a consistent relationship between the increase in virtual machines and the lengthening of makespan. Table 4 lists the test results of both the makespan and the throughput using different numbers of virtual machines.

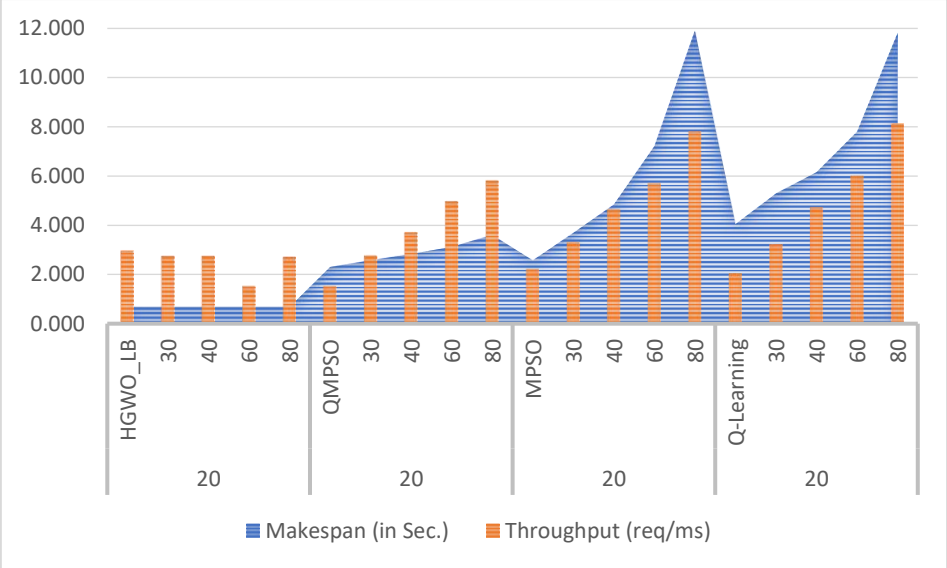

Figure 5: Throughput and makespan relationship.

Table 4: The effect of using different virtual machine numbers on the performance

task	VM	makespan	throughput
200	10	6.748	75.078
	20	7.081	18.848
	30	7.381	9.266
	40	7.552	5.742
	50	7.552	5.742
AVG		7.263	22.935
400	10	6.772	65.809
	20	7.079	20.394
	30	7.456	8.077
	40	7.612	5.235
	50	7.811	5.076
AVG		7.346	20.918
800	10	7.079	20.394
	20	7.097	16.632
	30	7.142	13.954
	40	7.322	10.886
	50	7.609	7.641
AVG		7.25	13.901

6. Conclusion

The paper presented a new metaheuristic algorithm designed to improve job allocation across virtual machines by progressively reducing the search area to get the best results. This strategic approach significantly enhanced the load-balancing process compared to conventional methodologies. The results showed that the suggested algorithm effectively navigated and used the search area, enhancing performance in throughput and makespan. Furthermore, the method efficiently reduced task wait times, improving overall system efficiency.

The technique's main goal was to distribute workloads evenly by assigning jobs to appropriate virtual machines based on their fitness levels. The approach demonstrated its superiority and effectiveness in tackling load-balancing issues compared to state-of-the-art algorithms: QMPSO, Q-learning, and MPSO. The LF-GWO-VM-LB approach showed excellent performance across various workload distributions on virtual computers, highlighting its effectiveness and dependability.

Future research will concentrate on dynamic load balancing to enhance real-time task allocation, especially for interdependent jobs. Integrating the Levy flight strategy into the enhanced gray wolf optimization (IGWO) technique and the current DLH strategy proved challenging for inexperienced researchers due to the algorithm's intricacy. The algorithm's ability to circumvent local optima and alignment with research objectives underscored its significance for advancing load-balancing algorithms. Enhancing and clarifying the algorithm's complexities would make it easier for academics to comprehend and use, thereby aiding in the discussion on improving distributed computing systems.

References

[1] K. Pradeep, L. J. Ali, N. Gobalakrishnan, C. J. Raman, and N. Manikandan, "CWOA: Hybrid Approach for Task Scheduling in Cloud Environment," The Computer Journal, vol. 65, no. 7, pp. 1860–1873, July 2022, https://doi.org/10.1093/comjnl/bxab028

[2] G. Natesan and A. Chokkalingam, "An Improved Gray Wolf Optimization Algorithm Based Task Scheduling in Cloud Computing Environment," The International Arab Journal of Information Technology, vol. 17, no. 1. pp. 73–81, Jan. 2019, doi: https://doi.org/10.34028/iajit/17/1/9.

[3] N. Gobalakrishnan and C. Arun, "A New Multi-Objective Optimal Programming Model for Task Scheduling using Genetic Gray Wolf Optimization in Cloud Computing," The Computer Journal, vol. 61, no. 10, pp. 1523–1536, Mar. 2018, doi: https://doi.org/10.1093/comjnl/bxy009.

[4] I. Casas, J. Taheri, R. Ranjan, L. Wang, and A. Y. Zomaya, "GA-ETI: An enhanced genetic algorithm for the scheduling of scientific workflows in cloud environments," Journal of Computational Science, vol. 26, pp. 318–331, Sep. 2016, doi: https://doi.org/10.1016 /j.jocs.2016 .08.007.

[5] J. Zhou and S. Dong, "Hybrid glowworm swarm optimization for task scheduling in the cloud environment," Engineering Optimization, vol. 50, no. 6, pp. 949–964, Aug. 2017, doi: https://doi.org/10.1080/0305215x.2017.1361418.

[6] Manikandan Nanjappan, Gobalakrishnan Natesan, and Pradeep Krishnadoss, "An Adaptive Neuro-Fuzzy Inference System and Black Widow Optimization Approach for Optimal Resource Utilization and Task Scheduling in a Cloud Environment," Wireless Personal Communications, vol. 121, no. 3, pp. 1891–1916, Aug. 2021, doi: https://doi.org/10.1007/s11277-021-08744-1.

[7] T. Zhou, D. Tang, H. Zhu, and Z. Zhang, "Multi-agent reinforcement learning for online scheduling in smart factories," Robotics and Computer-integrated Manufacturing, vol. 72, pp. 102202–102202, Dec. 2021, doi: https://doi.org/10.1016/j.rcim.2021.102202.

[8] Beni, G., "Swarm Intelligence," In: Sotomayor, M., Pérez-Castrillo, D., Castiglione, F. (eds) Complex Social and Behavioral Systems . Encyclopedia of Complexity and Systems Science Series, pp. 791-818, Springer, New York, NY, 2020. https://doi.org/10.1007/978-1-0716-0368-0_530

[9] M. K. Hussein and M. H. Mousa, "Efficient Task Offloading for IoT-Based Applications in Fog Computing Using Ant Colony Optimization," IEEE Access, vol. 8, pp. 37191–37201, 2020, doi: https://doi.org/10.1109/access.2020.2975741.

[10] A. N. Toosi and R. Buyya, "A Fuzzy Logic-Based Controller for Cost and Energy Efficient Load Balancing in Geo-distributed Data Centers," 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), Limassol, Cyprus, 2015, pp. 186-194, doi: 10.1109/UCC.2015.35.

[11] O. Y. Abdulhammed, "Load balancing of IoT tasks in the cloud computing by using sparrow search algorithm," The Journal of Supercomputing, vol. 78, pp. 3266–3287, 2022. https://doi.org/10.1007/s11227-021-03989-w

[12] M. Kim and I. -Y. Ko, "An Efficient Resource Allocation Approach Based on a Genetic Algorithm for Composite Services in IoT Environments," 2015 IEEE International Conference on Web Services, New York, NY, USA, 2015, pp. 543-550, doi: 10.1109/ICWS.2015.78.

[13] G. Saravanan, S. Neelakandan, P. Ezhumalai, and S. Maurya, "Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing," Journal of Cloud Computing, vol. 12, no. 1, Feb. 2023, doi: https://doi.org/10.1186/s13677-023-00401-1.

[14] T. Gao, Q. Tang, J. Li, Y. Zhang, Y. Li and J. Zhang, "A Particle Swarm Optimization With Lévy Flight for Service Caching and Task Offloading in Edge-Cloud Computing," in IEEE Access, vol. 10, pp. 76636-76647, 2022, doi: 10.1109/ACCESS.2022.3192846.

[15] Saeed, "Efficient Cloud-Based Resource Sharing Through Multi-Tenancy and Load Balancing: An Exploration of Higher Education and Digital Libraries," Iraqi journal of science, vol. 64, no. 8, pp. 4147–4159, Aug. 2023, doi: 10.24996/ijs.2023.64.8.35.

[16] A. R. DAR, D. Ravindran, and S. Islam, "Fog-based Spider Web Algorithm to Overcome Latency in Cloud Computing," Iraqi Journal of Science, vol. 61, No. 7, pp. 1781-1790, Jul. 2020, https://doi.org/10.24996/ijs.2020.61.7.27.

[17] S. Ali and Raaid Alubady, "A Survey of Fog Computing-Based Resource Allocation Approaches: Overview, Classification, and Opportunity for Research," Iraqi Journal of Science, vol. 65, no. 7, pp. 4008–4029, Jul. 2024, doi: 10.24996/ijs.2024.65.7.37.

[18] Saeed Qasim Al-Khalidi Al-Maliki, "Efficient Cloud-Based Resource Sharing Through Multi-Tenancy and Load Balancing: An Exploration of Higher Education and Digital Libraries," Iraqi journal of science, vol. 64, no. 8, pp. 4147–4159, Aug. 2023, doi: 10.24996/ijs.2023.64.8.35.

[19] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Gray Wolf Optimizer," Advances in Engineering Software, vol. 69, pp. 46–61, Mar. 2014, doi: https://doi.org/10.1016/j.advengsoft.2013.12.007.

[20] M. Banaie-Dezfouli, M. H. Nadimi-Shahraki, and Z. Beheshti, "R-GWO: Representative-based gray wolf optimizer for solving engineering problems," Applied Soft Computing, vol. 106, p. 107328, Jul. 2021, doi: https://doi.org/10.1016/j.asoc.2021.107328.

[21] M. H. Hashem, H. S. Abdullah, and K. I. Ghathwan, "Grey Wolf Optimization Algorithm: A Survey," Iraqi Journal of Science, vol. 64, no. 11, pp. 5964–5984, Nov. 2023, doi: 10.24996/ijs.2023.64.11.40.

[22] F. A. Saif, R. Latip, Z. M. Hanapi and K. Shafinah, "Multi-Objective Gray Wolf Optimizer Algorithm for Task Scheduling in Cloud-Fog Computing," in IEEE Access, vol. 11, pp. 20635-20646, 2023, doi: 10.1109/ACCESS.2023.3241240.

[23] Dina Riadh Abdulrazzaq, Narjis Mezaal Shati, and H. K. Hoomod, "Task Scheduling in a Cloud Environment Based on Meta-Heuristic Approaches: A Survey," Iraqi journal of science, vol. 65, no. 2, pp. 1001–1023, Feb. 2024, doi: 10.24996/ijs.2024.65.2.33.

[24] K. Meidani, A. Hemmasian, S. Mirjalili, and A. Barati Farimani, "Adaptive gray wolf optimizer," Neural Computing and Applications, vol. 34, no. 10, pp. 7711–7731, Jan. 2022, doi: https://doi.org/10.1007/s00521-021-06885-9.

[25] Z. Mustaffa, M. H. Sulaiman, M. F. M. Mohsin, Y. Yusof, F. Ernawan, B. Yusob, and N. M. Noor, "An application of Barnacle Mating Optimizer in Infectious Disease Prediction: A Dengue Outbreak Cases," Iraqi journal of science, vol. 61, no. 8, pp. 2132-2141, Aug. 2020, https://doi.org/10.24996/ijs.2020.61.8.28.

[26] T. Ladhari, I. Khoja, F. Msahli, and A. Sakly, "Parameter identification of a reduced nonlinear model for an activated sludge process based on cuckoo search algorithm," Transactions of the Institute of Measurement and Control, vol. 41, no. 12, pp. 3352–3363, Feb. 2019, doi: 10.1177/0142331218824384.

[27] U. K. Jena, P. K. Das, and M. R. Kabat, "Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment," Journal of King Saud University-Computer and Information Sciences, vol. 34, no. 6, pp. 2332-2342, 2022.

[28] A. Hussain and M. Aleem, "GoCJ: Google cloud jobs dataset for distributed and cloud computing infrastructures,"*Data*, vol. 3, no. 4, p. 38, 2018. https://doi.org/10.3390/data3040038

[29] S. Raychaudhuri, "Introduction to Monte Carlo simulation," *2008 Winter Simulation Conference*, Miami, FL, USA, 2008, pp. 91-100, doi: 10.1109/WSC.2008.4736059.

[30] A. B. Kadhim and S. D. Mahdi, "Determination of the Shape and Dimensions of the Sensitive Volume for Solid State Detectors Using Monte Carlo Computer Technique," Iraqi Journal of Science, vol. 55, no. 4B, pp. 1982–1991, Jul. 2023.