A Novel Hybrid Algorithm Based on Butterfly and Flower Pollination Algorithms for Scheduling Independent Tasks on Cloud Computing

Huiying SHAO
Hebei Vocational University of Technology and Engineering, Xingtai 054000, China

***Abstract*—Cloud computing is an Internet-based computing paradigm where virtual servers or workstations are offered as platforms, software, infrastructure, and resources. Task scheduling is considered one of the major NP-hard problems in cloud environments, posing several challenges to efficient resource allocation. Many metaheuristic algorithms have been extensively employed to address these task-scheduling problems as discrete optimization problems and have given rise to some proposals. However, these algorithms have inherent limitations due to local optima and convergence to poor results. This paper suggests a hybrid strategy for organizing independent tasks in heterogeneous cloud resources by incorporating the Butterfly Optimization Algorithm (BOA) and Flower Pollination Algorithm (FPA). Although BOA suffers from local optima and loss of diversity, which may cause an early convergence of the swarm, our hybrid approach outperforms such weaknesses by exploiting a mutualism-based mechanism. Indeed, the proposed hybrid algorithm outperforms existing methods while considering different task quantities with better scalability. Experiments are conducted within the CloudSim simulation framework with many task instances. Statistical analysis is performed to test the significance of the obtained results, which confirms that the suggested algorithm is effective at solving cloud-based task scheduling issues. The study findings indicate that the hybrid metaheuristic algorithm could be a promising approach to improving resource utilization and optimizing cloud task scheduling.**

Keywords*—*Task scheduling; cloud computing; butterfly optimization algorithm; flower pollination algorithm; mutualism

I. INTRODUCTION

Cloud computing is an Internet-based approach that enables elastic, easy-to-scale access to a broad set of resources, including storage, computing, and networking applications delivered via the Internet [1]. In contrast to traditional systems, dependent on locally stored resources, cloud computing allows flexible and scalable access to resources from any location. There are three main service configurations: Platform as a Service (PaaS), Infrastructure as a Service (IaaS), and Software as a Service (SaaS) [2]. IaaS involves virtualized assets accessible through the internet, offering elementary amenities like servers, storage facilities, and networking, empowering companies to expand slanted utopias without trusting physical devices [3].

While IaaS provides virtualized computing resources, PaaS is an extension that offers developers a platform complete with tools and frameworks [4]. This allows the developer to create, evaluate, and launch applications without explicitly managing the underlying infrastructure. SaaS directly delivers ready-to-consumer applications to end-users, including email, CRM, and collaborative software, accessed via web browsers [5]. Together, these models catalyze innovation and cost-efficiency in sectors by allowing companies to lessen their IT overhead, hasten product deployment, and respond dynamically to market demands.

Scheduling tasks in cloud computing belongs to fundamental NP-hard problems that need to be solved to ensure better efficiency in resource allocation within virtual environments [6]. This problem falls under the combinatorial optimization class wherein multiple heterogeneous tasks must be assigned to available resources for maximum efficiency [7]. As finding an optimal resource allocation problem in scheduling tasks with different requirements is often combinatorial, more advanced strategies provide an alternative to conventional approaches. Common objectives in task scheduling include reducing execution time (or makespan) to ensure tasks are completed as quickly as possible, which enhances user satisfaction and system performance [8].

The other objective is to perform load balancing, in which tasks should be allocated to resources to avoid bottleneck situations and overutilization of particular servers, providing better system resiliency [9]. Last but not least, efficient usage of resources will prevent the idleness of resources and minimize operational costs by utilizing the maximum availability of infrastructure [10]. Therefore, efficient scheduling strategies are crucial for cloud environments, where dynamic scaling of resources relies on accurate and adaptive scheduling to accommodate the diversified requirements of end-users and applications.

Simultaneously, advancements in mobile robotics, particularly in navigation and mapping, provide valuable insights into addressing dynamic resource allocation challenges in cloud environments. Techniques such as reinforcement learning have demonstrated the potential to enhance decision-making and adaptability in complex scenarios [11]. These insights could inspire novel approaches to optimizing task scheduling in cloud computing, where dynamic and unpredictable demands necessitate intelligent and resilient solutions.

Metaheuristic algorithms, including the Flower Pollination Algorithm (FPA) and Butterfly Optimization Algorithm (BOA), are employed in cloud task scheduling because of their flexibility in handling complicated optimization challenges [12]. BOA is inspired by butterflies' sensory communication through fragrance. The fragrance guides each solution or member chemically to an optimal solution, mirroring the prey's natural process. This mechanism will help explore potential solutions within the search space and zoom into promising, high-quality areas [13]. On the other hand, FPA draws inspiration from flowers' pollination behavior, combining local and global pollination processes to examine the solution domain effectively. The global pollination phase facilitates the exploration of diverse solutions, while local pollination fine-tunes promising areas [14].

While BOA and FPA have emerged as promising optimization techniques, they face significant limitations when applied to task scheduling. BOA often experiences early convergence and can become trapped in local optima due to its limited ability to fully utilize the optimal solution. Additionally, BOA’s phase-switching mechanism may become disoriented, deviating from the best global solutions. Similarly, FPA, despite its strength in balancing exploration and exploitation, can suffer from reduced diversity over time, limiting its capacity to explore novel solutions. To overcome these challenges, this study introduces a novel hybrid algorithm that integrates BOA and FPA through a mutualism mechanism inspired by ecological interactions.

In this context, the strengths of one algorithm offset the weaknesses of the other, creating a synergistic optimization process. Furthermore, we propose an adaptive switching probability mechanism, a key innovation of this study, which dynamically adjusts the balance between the exploitation and exploration phases. This unique combination enhances the search process, improves convergence, and significantly optimizes cloud-based task scheduling, marking a substantial contribution to cloud computing optimization.

The remainder of the paper is organized as follows: The state-of-the-art review is outlined in Section II, about different existing cloud task-scheduling approaches as well as different meta-heuristic algorithms. This is followed by describing, in Section III, the problem statement, which includes the challenges and objectives that characterize cloud task scheduling. Section IV outlines our hybrid novel algorithm, illustrating its various components, including the mechanism behind the mutualism and switching probability adaptation process. Section V describes the experimental setup and discusses the results of the simulation. The implications of the findings are discussed in detail in Section VI. Finally, the paper concludes by summarizing the contributions in Section VII and presenting possible further research.

II. Related Work

This section summarizes recent advancements in cloud task scheduling algorithms, as summarized in Table I. Various hybrid and metaheuristic approaches are highlighted, focusing on optimizing makespan, resource utilization, and load balancing to handle scheduling challenges in cloud computing environments.

TABLE I. SUMMARY OF RELATED WORKS ON CLOUD TASK SCHEDULING ALGORITHMS

Research	Description	Performance metrics	Key Findings
[15]	Genetic algorithm and multi-verse optimization are integrated to optimize task scheduling, focusing on bandwidth, virtualization, task counts, and sizes in cloud environments.	Time minimization and task transfer efficiency	Shows promising results in minimizing time for massive tasks by optimizing resource allocation.
[16]	Combination of genetic algorithm and thermodynamic simulated annealing, with crossover operator and thermodynamic mechanisms for balanced exploration and exploitation.	Effectiveness, speedup, schedule duration, and makespan	Effective in balancing exploration and exploitation and reducing makespan compared to other approaches.
[17]	Multiple objective task scheduling using grey wolf optimization, prioritizing tasks based on resource status and demand using HPC2N and NASA workload archives.	Makespan and resource allocation efficiency	Achieves significant improvements in scheduling parameters and adapts well to workload variability.
[18]	A novel method merging particle swarm optimization and genetic algorithms using phagocytosis-inspired merging for population diversity with a feedback mechanism.	Task completion time and convergence accuracy	Enhances task completion time and accuracy by guiding population movement toward optimal solutions.
[19]	Hybrid grey wolf optimization and genetic algorithm	Makespan, cost, and energy consumption	Outperforms GWO, GA, and PSO in minimizing makespan, energy use, and cost for large scheduling tasks.
[20]	The chameleon and remora search optimization algorithm integrates CSA and RSOA with a greedy approach focusing on MIPS and network bandwidth.	Makespan, load balancing, and cost	Effectively minimizes completion time and balances VM load, outperforming baseline approaches.
[21]	Uses dense spatial clustering to schedule tasks, aiming to optimize execution time and enhance the quality of service for user tasks.	Execution time, average start time, and completion time	Achieved a 13% reduction in execution time and a 49% improvement in start and completion times over ACO and PSO algorithms.

Abualigah and Alkhrabsheh [15] presented MVO-GA, a hybrid multi-verse optimizer and genetic algorithm to optimize task scheduling. In such a way, this approach enhances task transfer efficiency in a cloud system by investigating various aspects of cloud assets, including bandwidth, virtualization, task counts, and task sizes. The technique has shown promising results in minimizing the time used for massive cloud tasks.

Tanha, et al. [16] developed a combined meta-heuristic algorithm using the thermodynamic simulated annealing and genetic algorithms to resolve the cloud task scheduling issue. The performance of the algorithm is improved by a crossover operator and thermodynamic simulated annealing. In this approach, there is a reasonable equilibrium between exploration and exploitation.

Mangalampalli, et al. [17] suggested the multi-objective task scheduling grey wolf optimization algorithm, MOTSGWO, in which tasks are prioritized based on cloud resource status and workload demand. This approach is implemented in the Cloudsim toolkit with workloads generated from the HPC2N and NASA parallel workload archives. The experiments show the outstanding performance of MOTSGWO.

Fu, et al. [18] created a novel methodology using phagocytosis combined with particle swarm optimization and genetic algorithms. The method divides particles, adjusts their positions, and merges subpopulations for diversity. It uses a feedback mechanism to ensure the population moves towards the optimal solution. Simulations show it enhances cloud task completion time and convergence accuracy.

Behera and Sobhanayak [19] developed an algorithm that combines the Grey Wolf Optimization Algorithm (GWO) and Genetic Algorithm (GA) to improve cloud computing task scheduling. It aims to minimize cost, makespan, and energy usage while leveraging the GA-driven GWO algorithm's accelerated convergence in significant scheduling challenges.

Pabitha, et al. [20] developed a Chameleon and Remora Search Optimization Algorithm (CRSOA) to optimize cloud task scheduling by considering MIPS and network bandwidth. The CRSOA model, a multi-objective model, integrates the strengths of the Chameleon Search Algorithm (CSA) and Remora Search Optimization Algorithm (RSOA) through a greedy strategy. Simulation results showed that the CRSOA approach minimizes completion time and effectively handles load balancing between available VMs. The experimental investigation confirmed its effectiveness in reducing makespan, cost, and imbalance levels over baseline approaches.

Mustapha and Gupta [21] designed an approach based on DBSCAN (Density-Based Spatial Clustering) for task scheduling to ensure optimal effectiveness. DBSCAN-based task scheduling methodology enhances user satisfaction and optimises execution times, average start times, and end times. The experimental results reveal that the suggested model surpasses the current PSO and ACO, demonstrating 15% better execution times and 48% better start and completion times.

III. Problem Statement

The cloud task scheduling problem revolves around efficiently assigning multiple tasks to Virtual Machines (VMs) within a Cloud System (CS) to achieve the shortest possible execution time [22]. An overview of the proposed task scheduling system for a CS is shown in Fig. 1. The task manager component in this system collects tasks from different users. Upon receiving user tasks, the task manager arranges and forwards them to the scheduler component. The task manager also knows the basis for VMs' information. As a result, it supplies the task scheduler component with information about the status of VMs and task requests. In this context, the CS is represented by multiple Physical Machines (PMs), each housing several VMs [23], expressed as follows:

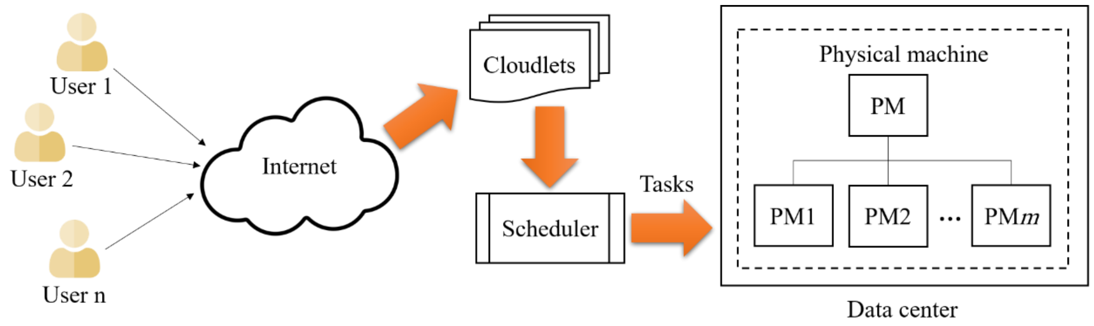

Fig. 1. System model for cloud task scheduling.

$$CS = \{PM_1, PM_2, \dots, PM_i, \dots, PM_m\} \tag{1}$$

Where m reflects the number of PMs in the system, and each PM i comprises a set of VMs as follows:

$$PM = \{VM_1, VM_2, \dots, VM_k, \dots, VM_n\} \tag{2}$$

Where n denotes the number of VMs within a particular PM. Each VM_k is defined by its processing speed $MIPS_k$ (measured in millions of instructions per second) and unique identifier SID_{VMk} [24]. The tasks to be scheduled in the cloud are detailed as follows:

$$T = \{T_1, T_2, \dots, T_l, \dots, T_z\} \tag{3}$$

Where z stands for the total number of tasks, and each task T_l is described by an identifier SID_{Tl}, its length in terms of instructions (task_length), the expected completion time ECT_l, and priority PI. The expected completion time for a task T_l on VM_k is calculated using Eq. (4) [25].

$$ECT_{lk} = \frac{T\_length_l}{MIPS_k} \tag{4}$$

This scheduling problem is, therefore, formulated as an optimization problem. The objective is to distribute tasks across VMs to minimize total execution time, thereby maximizing resource utilization. The objective function to shorten the overall makespan can be expressed as:

$$fit = \min\left(\max_{k=1,2,\ldots,n} \sum_{l=1}^{z} ECT_{lk}\right) \tag{5}$$

This approach aims to balance the load across virtual machines and optimize resource usage within the cloud infrastructure.

IV. Methodology

A. Butterfly Optimization Algorithm

BOA is a metaheuristic algorithm inspired by butterflies' cooperative foraging behavior. In the BOA, butterflies can find optimal solutions based on a fragrance function, influenced by parameters such as power exponent (a) and sensory modality (c) [26]. This fragrance, which represents the butterfly's fitness, is defined by Eq. (6).

$$f(t) = c \cdot I(t)^a \tag{6}$$

Where $I(t)$ denotes the stimulus intensity at a given step t, controlled by the sensory modality and power exponent. The fragrance emitted by each butterfly attracts others and guides their movement through the solution space.

BOA operates in three primary phases: initialization, iteration, and finalization [27]. During the initialization process, the objective function and the solution area are defined, generating a population of butterflies. Each butterfly's position is set, and fitness and fragrance scores are computed. In the iteration stage, BOA alternates between global and local key search strategies. Based on Eq. (7), each butterfly is guided toward the fittest solution $*$ in the global search.

$$x_i^{t+1} = x_i^t + \left(r^2 \cdot (g^* - x_i^t)\right) \cdot f_i \tag{7}$$

Where x_i^t represents the position of the i^{th} butterfly at iteration t, $g*$ denotes the best current solution, f_i is the fragrance of the i^{th} butterfly, and r is a random number between 0 and 1. In the local search mode, butterflies move based on the influence of nearby individuals. The position update is given by:

$$x_i^{t+1} = x_i^t + \left(r^2 \cdot \left(x_j^t - x_k^t\right)\right) \cdot f_i \tag{8}$$

Where x_j^t and x_k^t are positions of two randomly selected butterflies in the population. This local interaction allows BOA to explore diverse regions within the solution space.

A switch probability (p) controls the balance between global and local searches, enabling the algorithm to dynamically shift from broad exploration to intense local refinement. This adaptive strategy helps BOA avoid premature convergence and enhances its ability to find optimal solutions effectively, making it suitable for complex optimization tasks such as cloud scheduling.

Generally, BOA involves five steps. The first step is initializing all BOA and problem parameters. BOA has five parameters: population size (N), number of iterations (Itr), c, a, and p. In the second step, the BOA generates all solutions randomly. The solutions take the form of length vectors with the same dimension as the problem dimension d. A matrix containing all the solutions creates the population as follows.

The BOA typically consists of five sequential steps. The initial step involves setting up all BOA-related parameters and problem-specific variables. BOA utilizes five key parameters: population size (N), number of iterations (Itr), and the constants c, a, and p. During the second step, BOA generates all solutions randomly. These solutions are represented as vectors of equal length, corresponding to the dimensionality of the problem (d). The collection of these solution vectors forms a population matrix, structured as follows.

$$Population = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_d^1 \\ x_1^2 & x_2^2 & \cdots & x_d^2 \\ \vdots & \vdots & \cdots & \vdots \\ x_1^N & x_2^N & \cdots & x_d^N \end{bmatrix} \tag{9}$$

The optimization problem's objective function serves to assess all potential solutions during the third step. Subsequently, the best-performing solution is designated as $g*$. In the fourth step, all solutions are revised according to the fitness values determined in the previous phase. To guide the search process locally or globally, a random number r r is generated and compared to the threshold p. If r is smaller than p, the butterfly executes a global movement following Equation 7; otherwise, it performs a local movement based on Equation 8. If the newly generated solution outperforms the previous one, it replaces the earlier solution. The value of $g*$ is then updated accordingly. Finally, the termination condition is evaluated. The pseudo-code outlining the general steps of the BOA is presented in Fig. 2.

```
Step 1:
Set up the problem-specific parameters.
Initialize BOA parameters, including the maximum iterations $Itr$, population size $N$, sensory modality $c$, power exponent α, and switch probability $p$.
Step 2:
Create an initial population matrix for butterfly positions.
Step 3:
Begin the main loop: repeat until the maximum number of iterations is reached.
For each solution in the population:
    • Calculate its fitness value.
    • Identify the current best solution $g*$.
Step 4:
For each butterfly in the population:
    • Generate a random number $r$ within the range [0, 1].
    • If $r<p$:
        ○ Update the butterfly's position using Eq. 7.
    • Otherwise:
        ○ Update the butterfly's position using Eq. 8.
    • If the updated solution improves, update the population with this new solution.
    • Adjust the sensory modality $c$ if necessary.
Step 5:
Check if the iteration count has reached the maximum limit:
If not, increment the iteration counter by one and continue the loop.
End the loop when $Itr$ is reached.
Return the best solution $g*$ found by the algorithm.
```

Fig. 2. Pseudo-code of BOA.

B. *Flower Pollination Algorithm*

FPA is a metaheuristic technique designed to solve complex optimization problems mimicking natural pollination. FPA follows the principles of two types of pollination found in nature: local and global pollination. Global pollination promotes exploration, allowing the algorithm to escape local optima, while local pollination emphasizes exploitation, speeding up convergence. The algorithm performs exploration or exploitation for each iteration based on a switching probability *p*, ensuring optimal solutions are found efficiently.

The algorithm searches for the global most attractive flower for each candidate solution in FPA, representing a flower in a *d*-dimensional space. This search is carried out to minimize the fitness function and locate the flower with the lowest fitness score, corresponding to the optimal solution. Four main steps are involved in the FPA's operation: initialization, fitness evaluation, pollination process, and selection. At first, the population of flowers *F* is defined using Eq. (10). Then, each solution X_{ij} within the defined search bounds is initialized using Eq. (11).

$$F = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,d} \end{pmatrix} \tag{10}$$

$$X_{ij} = x_{min} + (x_{max} - x_{min}).\mu \tag{11}$$

Where μ varies between 0 and 1. Eq. (12) refers to the objective function to evaluate fitness.

$$\mathrm{f(x)}, \quad \mathrm{X} = (\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_\mathrm{d}) \tag{12}$$

The fitness of each flower is determined, and the current optimal solution $g*$ is identified, which has the lowest fitness value among all flowers. A random number rand is determined by a uniform distribution between (0,1) for each flower. If *rand>p*, global pollination is accomplished as follows:

$$\mathrm{X}_\mathrm{i}^{\mathrm{t+1}} = \mathrm{X}_\mathrm{i}^\mathrm{t} + \mathrm{L} \cdot (\mathrm{X}_\mathrm{i}^\mathrm{t} - \mathrm{g}^*) \tag{13}$$

Where *L* follows a Lévy flight distribution to simulate long-distance pollination, represented as follows:

$$L(\lambda) \sim \frac{\lambda\Gamma(\lambda)\sin(\pi\lambda/2)}{\pi} \cdot \frac{1}{s^{1+\lambda}} \tag{14}$$

With $\lambda=1.5$ and $s>0$ as the step size. If $rand \leq p$, local pollination occurs, and the flower's position is updated based on the positions of two randomly selected solutions as follows:

$$X_i^{t+1} = X_i^t + \epsilon \cdot \left(X_j^t - X_k^t\right) \tag{15}$$

Where ϵ comes from the [0,1] range, and X_j and X_k correspond to randomly chosen flowers. Each flower is rounded up to the closest valid position. The new positions are evaluated for fitness and each flower is updated if fitness has improved. The best solution $g*$ is also updated if a better solution is found.

C. *Mutualism-based Hybrid Approach*

MHA aims to enhance BOA's exploration and exploitation abilities by combining this algorithm with the FPA. Previous studies, such as BOA/DE and BOA/ABC, have shown that hybridizing BOA with other algorithms can balance exploration (exploring the solution space broadly) and exploitation (improving solutions locally). However, these approaches still need more diversity, as they can become overly focused on high-performing solutions early on, leading to premature convergence.

The effectiveness of metaheuristic algorithms is determined by their capacity to maintain harmony throughout exploration and exploitation. Exploration refers to the search for solutions across the entire space while exploitation fine-tunes solutions around promising areas. Our approach introduces mutualism, a cooperative interaction between BOA and FPA to address this balance. This interaction allows the two algorithms to complement each other, with BOA providing global search capability and FPA enhancing local search.

MHA divides the population into two subgroups: butterflies and flowers. Each subgroup evolves independently, benefiting from BOA and FPA search properties. Dynamic switching probability is applied to determine when individuals should alternate between global and local searches, adapting based on the current optimization stage.

Mutualism in this context refers to the mutual benefit observed between butterflies and flowers in ecosystems. For instance, butterflies aid in pollination, while flowers provide nectar, benefiting both species. The hybrid algorithm simulates this mutualism using the Symbiotic Organisms Search (SOS) algorithm, which models ecosystem cooperation through mutualism, communalism, and parasitism. The mutualism stage, in particular, facilitates collaboration between two solutions by averaging their traits as follows:

$$Mutual_{agent} = \frac{X_i^t + X_j^t}{2} \tag{16}$$

The positions of the two interacting solutions X_i and X_j are then updated as follows:

$$X_i^{t+1} = X_i^t + \mathrm{rand}[0,1] \times \left(g^* - Mutual_{agent} \times BF1\right) \tag{17}$$

$$X_j^{t+1} = X_i^t + \mathrm{rand}[0,1] \times \left(g^* - Mutual_{agent} \times BF2\right) \tag{18}$$

Where $g*$ represents the most optimal solution in the population, *BF1* and *BF2* refer to attraction variables, and *rand[0,1]* gives a random value to introduce variability. In addition, a dynamic switching probability *p* regulates the equilibrium between exploration and exploitation, calculated by Eq. (19).

$$p = 0.8 - 0.1 \times \frac{(Max_{iter} - iter)}{Max_{iter}} \tag{19}$$

Where *Max_iter* indicates the total number of iterations and *iter* denotes the ongoing iteration. This probability decreases over time, favoring exploration early in the search and shifting towards exploitation as the algorithm progresses.

As shown in Fig. 3, through mutualism and adaptive switching, this hybrid strategy effectively incorporates the best features of both BOA and FPA, enhancing the diversity and convergence rate of the solution population. This hybrid method improves task scheduling performance by ensuring a comprehensive search in the solution space and optimizing the allocation of resources in cloud computing.

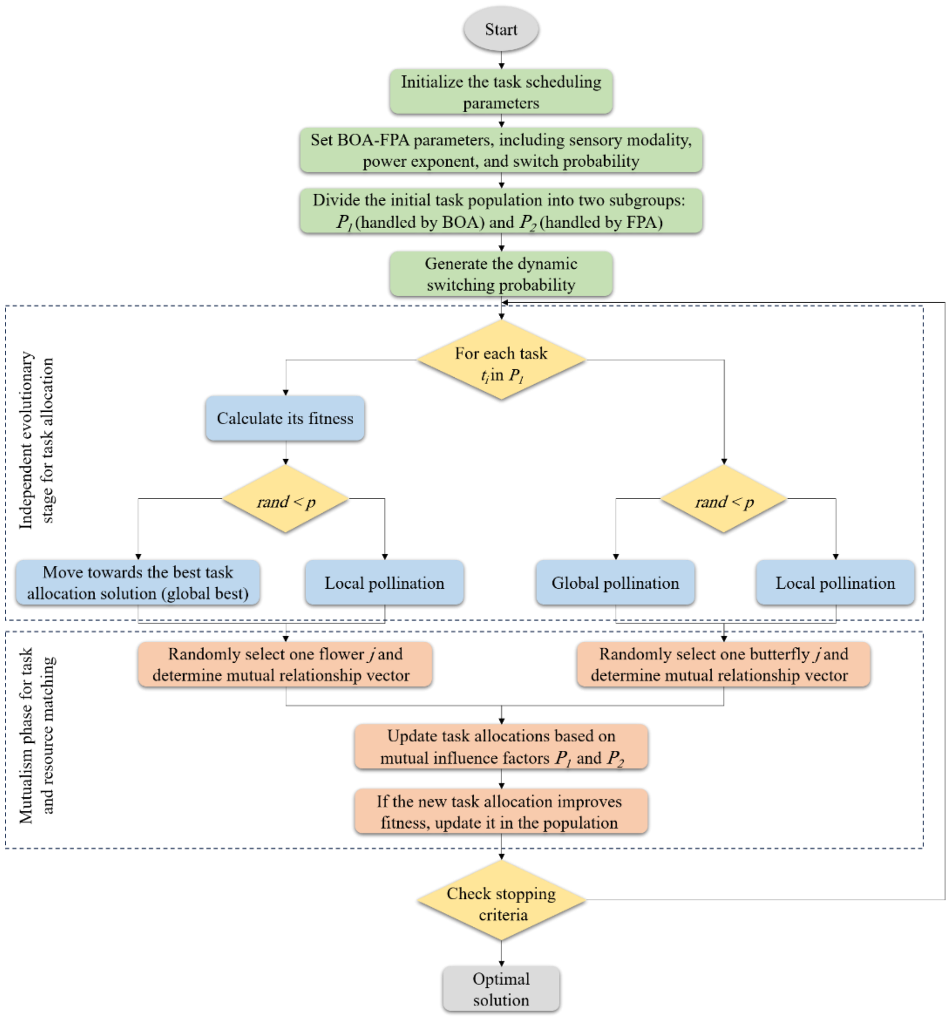
Start
Initialize the task scheduling parameters
Set BOA-FPA parameters, including sensory modality, power exponent, and switch probability
Divide the initial task population into two subgroups: P_1 (handled by BOA) and P_2 (handled by FPA)
Generate the dynamic switching probability
Independent evolutionary stage for task allocation
For each task t_i in P_i
Calculate its fitness
rand < p
Move towards the best task allocation solution (global best)
Local pollination
Global pollination
Local pollination
Mutualism phase for task and resource matching
Randomly select one flower j and determine mutual relationship vector
Randomly select one butterfly j and determine mutual relationship vector
Update task allocations based on mutual influence factors P_1 and P_2
If the new task allocation improves fitness, update it in the population
Check stopping criteria
Optimal solution

Fig. 3. Flowchart for proposed hybrid algorithm.

V. Performance Evaluation

To assess the performance of the algorithm, MHA, for cloud environments, simulations were performed on a synthetic dataset using MATLAB2018b on a PC powered by an Intel Core i5 3.5GHz CPU and 8GB RAM, running Windows 10. The experimental configuration, including the range of parameter values, is detailed in Table II. MHA was benchmarked against other algorithms, such as the Whale Optimization Algorithm (WOA), BOA, and FBA, using several key metrics: Makespan, Resource Utilization (RU), and imbalance degree.

TABLE II. Experimental Configuration and Parameter Ranges

Parameter	Value range
Bandwidth	500 Mbps
VM Memory (RAM)	1 GB
VM CPU Speed (MIPS)	3,000 to 5,000
No. of VMs	20
Task Size (Million instructions)	1,000 to 20,000
No. of tasks	100 to 1,000

Makespan, the interval between task starts and endpoints, measures scheduling efficiency. Fig. 4 compares average makespan values across different algorithms. For 100 tasks, MHA achieved an average makespan of approximately 15.2, outperforming comparative algorithms. MHA maintained lower makespan values as task sizes increased to 500 and 1000, with 71.4 and 140.2, respectively. Regarding large-scale cloud scheduling, MHA is significantly better at handling larger task sets. Imbalance degree measures the stability and balance of workload distribution across VMs. A lower value indicates better balance, reducing overload risk. This metric is calculated as follows:

$$\mathrm{ID} = \frac{ET_{max} - ET_{min}}{ET_{avg}} \quad (20)$$

Where ET_{min} and ET_{max} are the minimum and maximum execution times across VMs, and ET_{avg} is the average execution time. Fig. 5 shows ID comparisons for different algorithms. For 100 tasks, MHA achieved an imbalance degree of 0.71, lower than other algorithms. This trend of lower imbalance degree persisted as the number of tasks increased, demonstrating MHA's ability to maintain balanced workloads across VMs. RU measures the extent of VM utilization during task scheduling and is given by:

$$RU = \frac{\sum_{j=1}^{n} ET_j}{\text{makespan} \times m} \quad (21)$$

Where ET_j refers to the execution time of each VM and m represents the number of VMs. Fig. 6 illustrates that MHA achieved the highest RU values, indicating better resource usage.

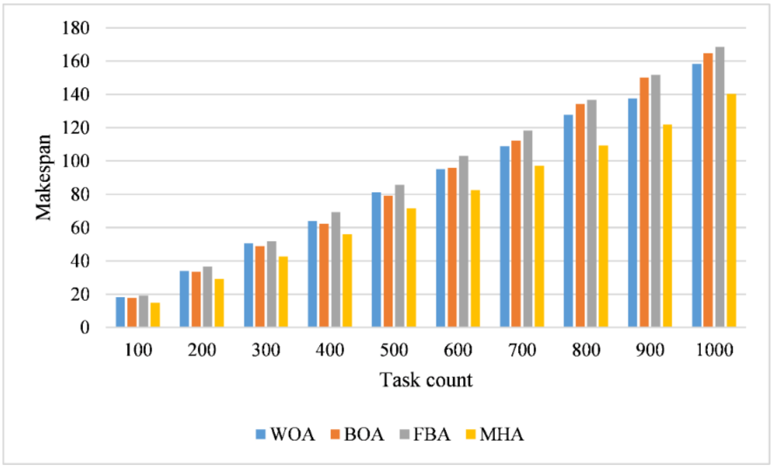

Fig. 4. Makespan results.

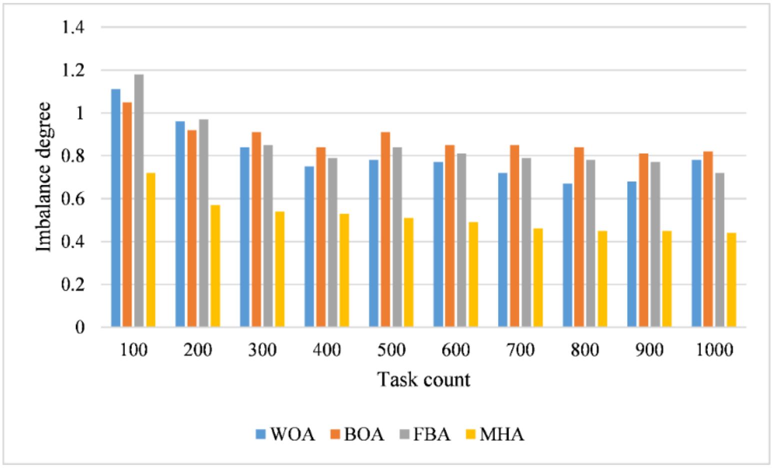

Fig. 5. Imbalance degree results.

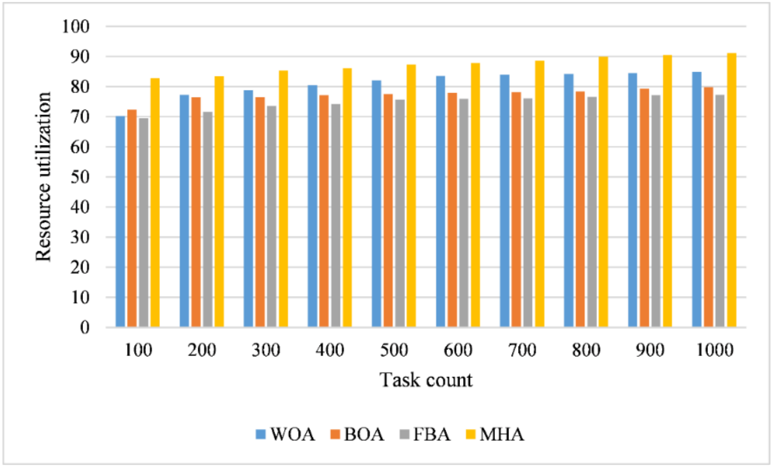

Fig. 6. Resource utilization results.

VI. DISCUSSION

The proposed hybrid algorithm significantly enhanced the challenge of cloud-based task scheduling. This section presents the implications of the results, the strengths of the proposed approach, and future avenues.

The hybridization of BOA and FPA through a mutualism-inspired mechanism addresses the individual limitations of each algorithm. BOA's tendency to converge prematurely is mitigated by FPA's enhanced diversity in exploration, while FPA's limited exploitation capabilities are bolstered by BOA's local search strengths. The introduction of an adaptive switching probability further enhances the balance between exploration and exploitation, allowing the algorithm to dynamically adjust its search strategy based on the progress of the optimization process. This innovative mechanism ensures robust performance, reducing the likelihood of stagnation in local optima and improving convergence speed.

Experimental results showed that the proposed hybrid algorithm achieved much better performance when compared to the benchmark methods on important metrics in this area, namely, makespan, resource utilization, and load balancing. Significance statistical tests are carried out that further establish the proposed algorithm's efficiency in handling diversified and dynamic challenges of task scheduling problems in cloud environments. The algorithm was found to work well with increased loads and much better performance according to scalability tests, hence its applicability to real-world applications.

The ability of the hybrid algorithm to optimize task scheduling has great implications for cloud computing environments. It can reduce operational costs, improve user satisfaction, and enhance system performance by efficiently allocating resources. Besides, it is a promising solution due to its adaptability and scalability in heterogeneous and dynamic cloud infrastructures.

Although the proposed algorithm outperforms the other methods by a great margin, some limitations must be conceded. Its performance is related to some parameters that might be fine-tuned in some situations. Therefore, Future work could address the proposition of automatic parameter-tuning mechanisms to make them more usable. Future work might also explore the effectiveness of the proposed hybrid approach for other optimization problems, such as load balancing in a distributed system or energy-aware scheduling.

VII. CONCLUSION

The paper proposed a new hybrid task scheduling algorithm called MHA, which combines BOA and FPA within a mutualism-based mechanism. MHA can effectively meet the most important challenges during Cloud platforms for effective task scheduling, such as minimizing makespan, maintaining workload balance across virtual machines, maximizing resource utilization, and improving overall scheduling performance. It achieves an excellent balance between exploration and exploitation through effective exploitation of BOA and FPA, with a guaranteed optimal distribution while the scheduling tasks continue to increase in scale and complexity. As reported from simulations, results show the outperformance of the MHA compared to traditional algorithms, proven by comparisons, which always have the best makespan with reduced imbalance degree and resource utilization. More specifically, the rate of performance improvement proves that MHA has considerably improved scheduling efficiency.

The adaptive dynamic switching probability in MHA enables the algorithm to scale up efficiently for large task sizes, presenting a robust approach for real-world cloud computing environments where dynamic and efficient task allocation is paramount. These results reflect that MHA can solve the current cloud scheduling requirements and provide a base for further enhancements in resource allocation strategies. Future research will be done on further hybridization with other metaheuristic algorithms, deep learning usage in the process for predictive scheduling, or even extending MHA to multi-objective optimization frameworks. These will eventually enhance scalability, adaptability, and efficiency in task scheduling in complex cloud computing scenarios.

REFERENCES

[1] V. Hayyolalam, B. Pourghebleh, M. R. Chehrehzad, and A. A. Pourhaji Kazem, "Single-objective service composition methods in cloud manufacturing systems: Recent techniques, classification, and future trends," Concurrency and Computation: Practice and Experience, vol. 34, no. 5, p. e6698, 2022.

[2] B. Pourghebleh, A. Aghaei Anvigh, A. R. Ramtin, and B. Mohammadi, "The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments," Cluster Computing, vol. 24, no. 3, pp. 2673-2696, 2021.

[3] A. Katal, S. Dahiya, and T. Choudhury, "Energy efficiency in cloud computing data centers: a survey on software technologies," Cluster Computing, vol. 26, no. 3, pp. 1845-1875, 2023.

[4] D. Mušić, J. Hribar, and C. Fortuna, "Digital transformation with a lightweight on-premise PaaS," Future Generation Computer Systems, 2024.

[5] M. Saleem, M. Warsi, and S. Islam, "Secure information processing for multimedia forensics using zero-trust security model for large scale data analytics in SaaS cloud computing environment," Journal of Information Security and Applications, vol. 72, p. 103389, 2023.

[6] V. Hayyolalam, B. Pourghebleh, A. A. Pourhaji Kazem, and A. Ghaffari, "Exploring the state-of-the-art service composition approaches in cloud manufacturing systems to enhance upcoming techniques," The International Journal of Advanced Manufacturing Technology, vol. 105, pp. 471-498, 2019.

[7] K. Saidi and D. Bardou, "Task scheduling and VM placement to resource allocation in Cloud computing: challenges and opportunities," Cluster Computing, vol. 26, no. 5, pp. 3069-3087, 2023.

[8] F. S. Prity, M. H. Gazi, and K. A. Uddin, "A review of task scheduling in cloud computing based on nature-inspired optimization algorithm," Cluster computing, vol. 26, no. 5, pp. 3037-3067, 2023.

[9] M.-L. Chiang, H.-C. Hsieh, Y.-H. Cheng, W.-L. Lin, and B.-H. Zeng, "Improvement of tasks scheduling algorithm based on load balancing candidate method under cloud computing environment," Expert Systems with Applications, vol. 212, p. 118714, 2023.

[10] 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, p. 24, 2023.

[11] M. D. Tezerjani, M. Khoshnazar, M. Tangestanizadeh, and Q. Yang, "A Survey on Reinforcement Learning Applications in SLAM," arXiv preprint arXiv:2408.14518, 2024, doi: https://doi.org/10.48550/arXiv.2408.14518.

[12] A. N. Malti, B. Benmammar, and M. Hakem, "Task Scheduling Optimization in Cloud Computing: A Comparative Study Between Flower Pollination and Butterfly Optimization Algorithms," in 2023 5th

International Conference on Pattern Analysis and Intelligent Systems (PAIS), 2023: IEEE, pp. 1-7.

[13] S. Arora and S. Singh, "Butterfly optimization algorithm: a novel approach for global optimization," Soft computing, vol. 23, pp. 715-734, 2019.

[14] X.-S. Yang, M. Karamanoglu, and X. He, "Flower pollination algorithm: a novel approach for multiobjective optimization," Engineering optimization, vol. 46, no. 9, pp. 1222-1237, 2014.

[15] L. Abualigah and M. Alkhrabsheh, "Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing," The Journal of Supercomputing, vol. 78, no. 1, pp. 740-765, 2022.

[16] M. Tanha, M. Hosseini Shirvani, and A. M. Rahmani, "A hybrid meta-heuristic task scheduling algorithm based on genetic and thermodynamic simulated annealing algorithms in cloud computing environments," Neural Computing and Applications, vol. 33, pp. 16951-16984, 2021.

[17] S. Mangalampalli, G. R. Karri, and U. Kose, "Multi Objective Trust aware task scheduling algorithm in cloud computing using Whale Optimization," Journal of King Saud University-Computer and Information Sciences, vol. 35, no. 2, pp. 791-809, 2023.

[18] X. Fu, Y. Sun, H. Wang, and H. Li, "Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm," Cluster Computing, vol. 26, no. 5, pp. 2479-2488, 2023.

[19] I. Behera and S. Sobhanayak, "Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach," Journal of Parallel and Distributed Computing, vol. 183, p. 104766, 2024.

[20] P. Pabitha, K. Nivitha, C. Gunavathi, and B. Panjavarnam, "A chameleon and remora search optimization algorithm for handling task scheduling uncertainty problem in cloud computing," Sustainable Computing: Informatics and Systems, vol. 41, p. 100944, 2024.

[21] S. D. S. Mustapha and P. Gupta, "DBSCAN inspired task scheduling algorithm for cloud infrastructure," Internet of Things and Cyber-Physical Systems, vol. 4, pp. 32-39, 2024.

[22] S. Gupta and S. Tripathi, "A comprehensive survey on cloud computing scheduling techniques," Multimedia Tools and Applications, vol. 83, no. 18, pp. 53581-53634, 2024.

[23] O. L. Abraham, M. A. Ngadi, J. B. M. Sharif, and M. K. M. Sidik, "Multi-Objective Optimization Techniques in Cloud Task Scheduling: A Systematic Literature Review," IEEE Access, 2025.

[24] S. Durairaj and R. Sridhar, "Coherent virtual machine provisioning based on balanced optimization using entropy-based conjectured scheduling in cloud environment," Engineering Applications of Artificial Intelligence, vol. 132, p. 108423, 2024.

[25] S. A. Murad et al., "Priority based job scheduling technique that utilizes gaps to increase the efficiency of job distribution in cloud computing," Sustainable Computing: Informatics and Systems, vol. 41, p. 100942, 2024.

[26] M. Alweshah, S. A. Khalaileh, B. B. Gupta, A. Almomani, A. I. Hammouri, and M. A. Al-Betar, "The monarch butterfly optimization algorithm for solving feature selection problems," Neural Computing and Applications, pp. 1-15, 2022.

[27] P. Chakraborty, S. Sharma, and A. K. Saha, "Convergence analysis of butterfly optimization algorithm," Soft Computing, vol. 27, no. 11, pp. 7245-7257, 2023.