# Enhanced Butterfly Optimization Algorithm for Task Scheduling in Cloud Computing Environments

Yue ZHAO
College of Computer, Cangzhou Jiaotong College, Cangzhou 061199, China

***Abstract*—Cloud computing is transforming the provision of elastic and adaptable capabilities on demand. A scalable infrastructure and a wide range of offerings make cloud computing essential to today's computing ecosystem. Cloud resources enable users and various companies to utilize data maintained in a distant location. Generally, cloud vendors provide services within the limitations of Service Level Agreement (SLA) terms. SLAs consist of various Quality of Service (QoS) requirements the supplier promises. Task scheduling is critical to maintaining higher QoS and lower SLAs. In simple terms, task scheduling aims to schedule tasks to limit wasted time and optimize performance. Considering the NP-hard character of cloud task scheduling, metaheuristic algorithms are widely applied to handle this optimization problem. This study presents a novel approach using the Butterfly Optimization Algorithm (BOA) for scheduling cloud-based tasks across diverse resources. BOA performs well on non-constrained and non-biased mathematical functions. However, its search capacity is limited to shifted, rotated, and/or constrained optimization problems. This deficiency is addressed by incorporating a virtual butterfly and improved fuzzy decision processes into the conventional BOA. The suggested methodology improves throughput and resource utilization while reducing the makespan. Regardless of the number of tasks, better results are consistently produced, indicating greater scalability.**

***Keywords*—Cloud computing; resource utilization; task scheduling; Butterfly Optimization Algorithm; fuzzy decision strategy**

## I. INTRODUCTION

The Internet of Things (IoT) has evolved from the exponential proliferation of smart sensors in recent years and the demand for inter/interconnections between devices [1, 2]. IoT opens up broad medical, manufacturing, and logistics opportunities, necessitating high reliability, durability, flexibility, adaptability, and control levels [3]. Furthermore, IoT devices are limited in resources and equipped with specialized chips configured with various rules [4]. In this way, conventional networks become more complex owing to the specific requirements of IoT applications. Software-Defined IoT (SD-IoT) aims to apply Software-Defined Networking (SDN) to IoT to bring elasticity to managing resources and networks in traditional networks. SDN is regarded as a critical paradigm for next-generation networking [5].

A group of networked computers with several shared computing resources is referred to as the cloud. Recently, cloud computing has developed rapidly, enabling globally distributed data centers to develop and scale up to provide high-quality and reliable services [6]. Cloud computing has emerged as an effective model for providing computational resources on a "pay-per-use" basis. It brings uniformity and transformation to IT enterprises [7]. Cloud computing has significant prospects and poses several problems in conventional IT evolution due to its expanding uses and promotion [8]. In recent years, cloud computing has become an alternative online strategy to empower users. It offers access to shareable and customizable resources on demand, quickly allocated and released with little management or collaboration from the cloud provider [9]. This invention offers several advantages, including enhanced economic benefits related to time, cost, inventory management, and storage. This breakthrough enables all programs to operate on a virtual platform, with resources allocated across Virtual Machines (VMs) [10].

An effective and dynamic task scheduler is essential when multiple users simultaneously request services from the cloud environment, particularly from diverse and heterogeneous resources [11]. An optimal and adaptable task scheduler is critical in the cloud paradigm. Moreover, it must function under the workload submitted to the cloud platform [12]. An inefficient scheduling process in the cloud environment causes diminished service quality from cloud service providers, eroding confidence and adversely affecting corporate operations [13]. Hardware virtualization is the basis for distributing cloud resources. Numerous VMs are hosted on an individual computing server to support multiple users executing concurrent processes. VMs running in cloud data centers are given thousands of tasks. Consequently, employing an effective scheduler inside the cloud framework is advantageous for cloud providers and customers, allowing mutual benefits.

Task scheduling assigns cloud tasks to VMs to shorten makespan and enhance resource usage. Due to the NP-hard characteristics of this issue, conventional scheduling techniques have challenges regarding scalability and efficiency, especially in dynamic cloud settings [14]. Metaheuristic algorithms, such as the Butterfly Optimization Algorithm (BOA), have been shown to help tackle complicated optimization problems [15]. These algorithms leverage techniques like random walks and graph-based embeddings to improve search efficiency and adaptability in diverse optimization contexts [16]. BOA is suitable for such tasks due to its simplicity, excellent balance between exploration and exploitation, and adaptability to diverse optimization landscapes. Besides, its computational efficiency and the ability to converge on high-quality solutions make it a competitive choice for improving cloud task scheduling. This study presents an improved BOA, including a fuzzy decision method and an innovative virtual butterfly idea to maximize the algorithm's search efficacy and flexibility in

cloud job scheduling. As a summary, this study made the following contributions:

- A novel variant of the BOA is introduced, incorporating a fuzzy logic model and a virtual butterfly design to improve the algorithm's search efficiency and adaptability in cloud computing environments.
- A fuzzy logic model is implemented to continuously alter the balance between BOA's exploration and exploitation phases, allowing for better adaptation to varying optimization conditions.
- A virtual butterfly agent is developed that aggregates information from all butterflies, directing the swarm to promising areas in the search area, thereby avoiding premature convergence and improving the overall solution quality.
- The enhanced algorithm is evaluated using CloudSim with GoCJ and HCSP datasets, demonstrating its superior performance in minimizing makespan, improving resource utilization, and enhancing throughput compared to other metaheuristic methods like PSO and standard BOA.

The remaining portion of the paper is laid out in the following arrangement. Section II summarizes related research on cloud-based task scheduling. Section III defines the task scheduling problem and presents the challenges in optimizing makespan and resource utilization. Section IV introduces the proposed enhanced BOA, detailing its fuzzy decision strategy and virtual butterfly concept. Section V reports the findings and analyzes the efficiency of the developed algorithm. Section VI discusses key findings and outlines the limitations of current work. Lastly, Section VII offers a conclusion and recommends areas for further research.

## II. Literature Review

Metaheuristic algorithms have been extensively adopted for scheduling tasks in cloud computing. Recent research efforts have focused on improving these algorithms to address local optima and poor convergence challenges. Some research studies emphasize the importance of parallel calculations in maximizing task scheduling efficiency. The need to balance exploration with exploitation has resulted in hybrid and adaptable methodologies. Nevertheless, several current methodologies encounter constraints when used in extensive, diverse cloud settings.

Dubey and Sharma [17] proposed the Chemical Reaction Partial Swarm Optimization (CR-PSO) method for distributing several independent jobs among VMs in a cloud computing context. This hybrid methodology integrates the merits of Chemical Reaction Optimization (CRO) and Particle Swarm Optimization (PSO), producing an optimum task scheduling sequence that accounts for both job requirements and deadlines. Hybridization optimizes makespan, decreases costs, and diminishes energy use. Comprehensive simulations were performed with the CloudSim tools, illustrating the algorithm's efficacy. The comparative examination of several scenarios, varying quantities of VMs and tasks, demonstrates a decrease in execution time between 1–6% and, at times, exceeding 10%. Furthermore, the CR-PSO algorithm boosted makespan by 5–12%, decreased costs by 2–10%, and improved energy efficiency by 1–9%. These findings validate the algorithm's capacity for enhanced resource management and scheduling in cloud systems.

Mangalampalli, et al. [18] developed a task scheduling system using Firefly Optimization, prioritizing jobs and VMs to guarantee precise scheduling. This methodology utilizes synthetic datasets with diverse distributions and workloads from NASA and HPC2N for assessment. This methodology, implemented in the CloudSim simulation environment, is contrasted with baseline methods like genetic, Ant Colony Optimization (ACO), and PSO algorithms. The simulation outcomes indicate that the firefly-based method substantially surpasses these benchmarks in minimizing makespan, augmenting resource availability, increasing the success rate, and decreasing turnaround time, thus producing a more dependable and effective scheduling solution for cloud environments.

Bezdan, et al. [19] suggested an enhanced bat algorithm to tackle multi-objective job scheduling in cloud settings. The strategy seeks to optimize efficiency while minimizing search duration. The methodology was assessed with the CloudSim toolbox on both regular and synthetic parallel workloads. The findings revealed that the hybridized bat algorithm surpasses conventional metaheuristic methods, highlighting its significant potential for enhancing job scheduling effectiveness.

Wu and Xiong [20] created an innovative job scheduling approach for cloud computing with PSO algorithm. Initially, the resource scheduling issue in a cloud computing ecosystem is simulated, and a task execution duration function is established. The updated PSO approach is then implemented to coordinate application activities and improve load distribution. It relies on the Copula algorithm to explore the correlation between variables and probability while defining the attractor component to prevent the objective function from being ensnared in local optimums. The analysis indicates that the proposed resource allocation and scheduling methodology may enhance cloud computing resource usage and decrease job completion time.

Mangalampalli, et al. [21] offered a multiple-objective task scheduling method based on the Grey Wolf Optimization (MOTSGWO) algorithm by optimizing scheduling options dynamically depending on resource availability and anticipated demand requirements. This approach allocates resources to meet customer budgets and work priorities. The MOTSGWO methodology is executed through the Cloudsim toolkit, with workloads generated via the development of datasets with varied task distributions and sequences sourced from NASA and HPC2N distributed repositories. The comprehensive evaluation findings reveal that MOTSGWO is superior to previous benchmark strategies and improves critical metrics.

Saif, et al. [22] presented a multi-goal GWO algorithm aimed at minimizing the QoS targets of latency and energy usage implemented inside the fog broker, which is crucial to job distribution. The experimental observations confirm the efficacy of the MGWO algorithm relative to contemporary algorithms in minimizing delay and energy consumption.

## III. PROBLEM DEFINITION

As shown in Fig. 1, a cloud data center comprises numerous Physical Machines (PMs), each capable of providing distinct end-user services. PMs can generate thousands of VMs dynamically. Alternatively, multiple host machines can collaborate to support a single VM. Cloud service providers offer VMs with different performance and pricing options, meeting a wide range of user requirements. This study explores the problem of allocating VMs to incoming, independent tasks. Each task is assumed to run exclusively on one VM and cannot be partitioned into smaller segments. Managing task scheduling in such an environment featuring varying capabilities is a complex challenge, represented by the sets of tasks and VMs in Eq. (1) and Eq. (2).

$$T = \{t_1, t_2, t_3, \dots, t_n\} \quad (1)$$

$$VM = \{vm_1, vm_2, vm_3, \dots, vm_m\} \quad (2)$$

The set $T$ represents tasks, each defined by a specific number of instructions. At the same time, $VM$ denotes a set of VMs, each with defined computational power determined by Millions of Instructions Per Second (MIPS). In most cases, the workload volume surpasses the available VMs. These arrays act as inputs to the scheduling algorithm, which seeks to derive an optimal mapping of tasks to VMs. This mapping outlines the assignment of tasks to VMs, as expressed in Eq. (3).

$$Map = \{(t_2, vm_1), (t_1, vm_3), (t_3, vm_2), \dots, (t_n, vm_m)\} \quad (3)$$

In the mapping solution, each task (represented as the first element of a tuple) is assigned uniquely to a VM, while a VM can be associated with multiple tasks. This implies that each task is allocated to a particular VM, while a VM may handle several assignments. The execution time ($ET$) for a specific task $t_i$ on a VM $vm_j$ is calculated using Eq. (4).

$$ET_{task_i vm_j} = Number\ of\ instruction\ in\ t_i\ /vm_j\ MIPS \quad (4)$$

It is assumed that each VM processes multiple tasks sequentially without interruption. Eq. (5) defines the overall completion time ($CT$) for all tasks allocated to a particular VM. Notably, faster VMs will complete their assigned tasks faster than slower ones.

$$CT_{vm_j} = \sum_{i=1}^{n} number\ of\ instrucions\ in\ t_i\ /vm_j\ MIPS \quad (5)$$

In metaheuristic algorithms, the arrangement of tasks on VMs is continuously adjusted to optimize their fitness values. As such, the task assignments on each VM may change during the algorithm's execution. If task $t_x$ is replaced with task $t_y$ on a VM, the completion time for multiple tasks executed in the VM is computed using Eq. (6).

$$CT_{vm_j} = CT_{vm_j} - (number\ of\ instrucions\ in\ t_x\ /vm_j\ MIPS) + (number\ of\ instrucions\ in\ t_y\ /vm_j\ MIPS) \quad (6)$$

A critical metric in task scheduling is the makespan, which refers to the total time required to complete all tasks across the available VMs. The makespan is determined using Eq. (7), representing the maximum completion time among all VMs involved in the scheduling process.

$$Makespan = \max\left(CT_{vm_j}\right)\ \forall j \epsilon 1,2, \dots, k \quad (7)$$

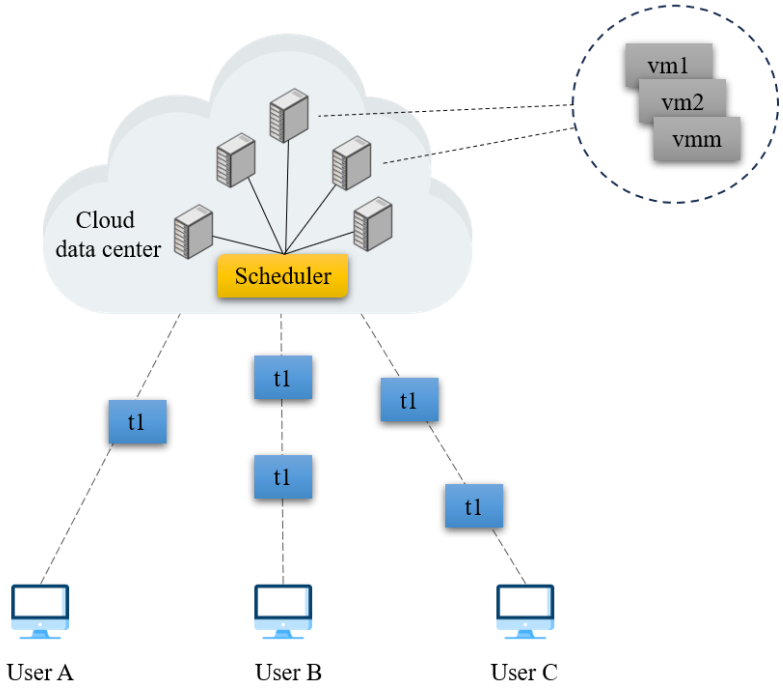


Fig. 1. Task scheduling process in cloud computing.

Throughout this study, makespan, completion time, and execution time are measured in seconds. Maximizing resource utilization during task scheduling is advantageous, ensuring a resource is fully utilized before allocating another instance on the cloud. Eq. (8) calculates the Average Resource Utilization (ARU) for PMs. This involves summing up the completion times of all VMs and dividing by the total number of VMs ($m$), followed by dividing the result by the makespan.

$$ARU = \left(\sum_{i=1}^{m} CT_{vm_j}/m\right)/makespan \quad (8)$$

System efficiency is further assessed through throughput, defined as the number of tasks processed per unit of time, calculated using Eq. (9). It is determined by dividing the total number of tasks by the makespan, resulting in a throughput value measured in terms of tasks accomplished per second.

$$Throughout = Total\ number\ of\ tasks /makespan \quad (9)$$

Additionally, Response Time ($RT$) is a key metric representing the duration between the scheduling decision and the initiation of task execution on a VM. Multiple VMs may operate on the identical PM, and a single VM may perform several tasks. Eq. (10) describes the mean response time for all tasks across multiple VMs. This calculation involves dividing the total initiation intervals for multiple tasks into the overall task count to determine the mean response time for each VM. The overall average response time is then obtained by averaging the response times across all VMs.

$$RT = \left(\sum_{j=1}^{m}\sum_{i=1}^{n} RT_i\right)/m \quad (10)$$

## IV. PROPOSED METHOD

BOA mimics the foraging behavior of butterflies, driven by their sense of smell (olfaction). It aims to resemble how butterflies locate food sources (flowers) by navigating their environment using global and local search capabilities. In cloud computing and task scheduling, BOA facilitates allocating tasks to VMs by finding near-optimal solutions through iterative improvements. BOA is characterized by three central concepts: olfactory modality, global and local search, and scent intensity.

In BOA, butterflies are represented as solutions, and their sense of smell is modeled to guide their movement. Each butterfly (solution) is attracted to the most promising areas (best solutions), allowing for effective exploitation and exploration of the search area. The algorithm alternates between global search (exploration) and local search (exploitation). Global search occurs when butterflies move towards a global highest-quality solution. The solution space can thus be explored in new ways. Local search is triggered when butterflies move toward other butterflies nearby, allowing solutions to be fine-tuned.

The effectiveness of butterflies' movement depends on the intensity of the scent, which changes based on their position in the search space. The scent is calculated using fitness functions, which assess how effective a given solution is concerning the objective by cutting down makespan or optimizing resource utilization. The scent intensity $S_i$ of each butterfly $i$ is calculated using Eq. (11):

$$S_i = c.f_i^a \quad (11)$$

where $c$ represents a sensory modality that affects scent strength, $f_i^a$ stands for the fitness value of butterfly $i$, and $a$ controls the nonlinearity of scent. The movement of a butterfly $i$ towards a global best solution (global search) is given by:

$$x_i^{t+1} = x_i^t + r.S_g.(g^t - x_i^t) \quad (12)$$

Where $x_i^t$ is the position of the butterfly $i$ at iteration $t$, $r$ stands for random number in the range [0, 1], $S_g$ refers to the scent intensity of the best solution found so far (global best), and $g^t$ specifies the global best position at iteration $t$. The movement of a butterfly towards another butterfly (local search) is represented as:

$$x_i^{t+1} = x_i^t + r.S_j.\left(x_j^t - x_i^t\right) \quad (13)$$

where $S_j$ is the scent intensity of butterfly $j$, and $x_j^t$ is its position at iteration $t$. The balance between global and local search is controlled using a probability parameter $p$. A random number $r \in [0,1]$ is compared to p to determine whether a butterfly moves towards the global best or another butterfly as follows:

- If $r<p$, the butterfly follows the global search strategy.
- If $r \geq p$, it engages in local search.

The task scheduling problem is treated as an optimization challenge in cloud computing. The objective is to reduce makespan, minimize energy consumption, and improve resource utilization by finding the best allocation of tasks to VMs. Each butterfly represents a potential solution, where:

- A solution is a specific mapping of tasks to VMs.
- The fitness of a solution is rated in terms of makespan, energy consumption, and other QoS metrics.

During each iteration, BOA adjusts butterflies' positions according to their scent intensities and the best solutions determined so far. Continuous iterations allow the algorithm to converge towards an optimal or near-optimal task allocation strategy.

The conventional BOA faces three primary challenges: (1) a fixed exploration-to-exploitation ratio controlled by the parameter p, leading to rigidity in the search process; (2) the possibility of being stuck in local optima due to a fixed global best attraction; and (3) pairwise interaction between butterflies, which limits search efficiency in complex optimization problems. The Fuzzy Butterfly Optimization Algorithm FBOA) overcomes these drawbacks by introducing (a virtual butterfly and fuzzy decision-making strategy. Fig. 2 compares conventional BOA and FBOA in terms of their exploration and exploitation strategies. FBOA integrates fuzzy logic to adjust the transition between these strategies.

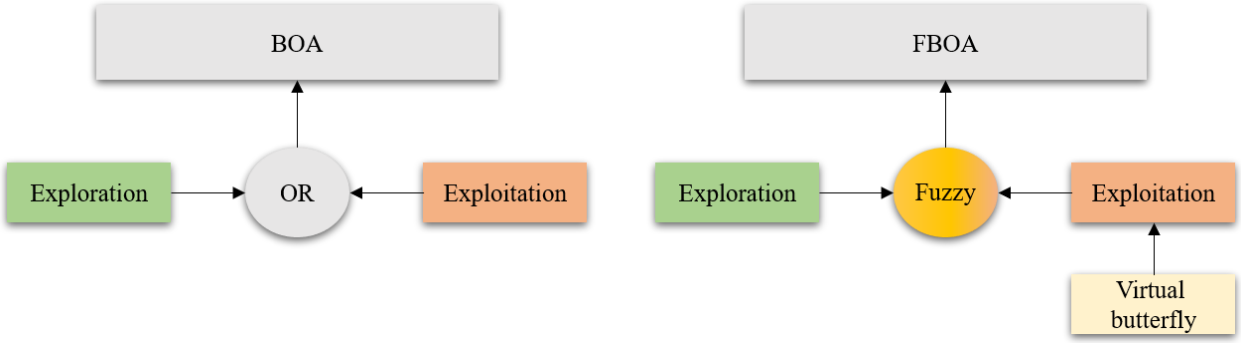


Fig. 2. Structure of BOA and FBOA.

To improve the adaptability of the BOA, FBOA employs a nine-rule fuzzy decision-making strategy. This strategy dynamically adjusts the tendency factor τi for each butterfly, which dictates its balance between exploitation and exploration based on the current optimization context. A novel concept called the virtual butterfly $X^v$ is introduced, which serves as a guiding agent in navigation. Unlike the standard BOA's random pairwise interactions, the virtual butterfly uses the information from the best solutions and adjusts its direction based on the current problem's objective function. The Normalized Objective Function (NOF) is calculated to estimate the relative effectiveness of each butterfly's position:

$$NOF_i = \frac{f(X_i) - f(g^*)}{f(X^{worst}) - f(g^*) + \mu}, \quad i = 1,2,\dots,M \qquad (14)$$

Where $f(X_i)$ is the objective function value of the $i^{th}$ butterfly, $f(g^*)$ is the fitness value of the current global best butterfly, $f(X^{worst})$ is the fitness value of the worst butterfly, $\mu$ is a small positive scalar to avoid division by zero, and $M$ is the population size. The fuzzy decision system updates the tendency factor τi using the NOF and predefined fuzzy rules.

$$\tau_i = \omega_\tau + \Delta\tau_i \qquad (15)$$

where $\omega_\tau$ indicates the origin of the tendency factor, $\Delta\tau_i$ is the adjustment value derived through the fuzzy inference process. Membership functions (as shown in Fig. 3) are used to categorize NOF into linguistic variables such as Small (S), Medium (M), and Large (L). The output values are adjusted based on the rules provided in Table I.

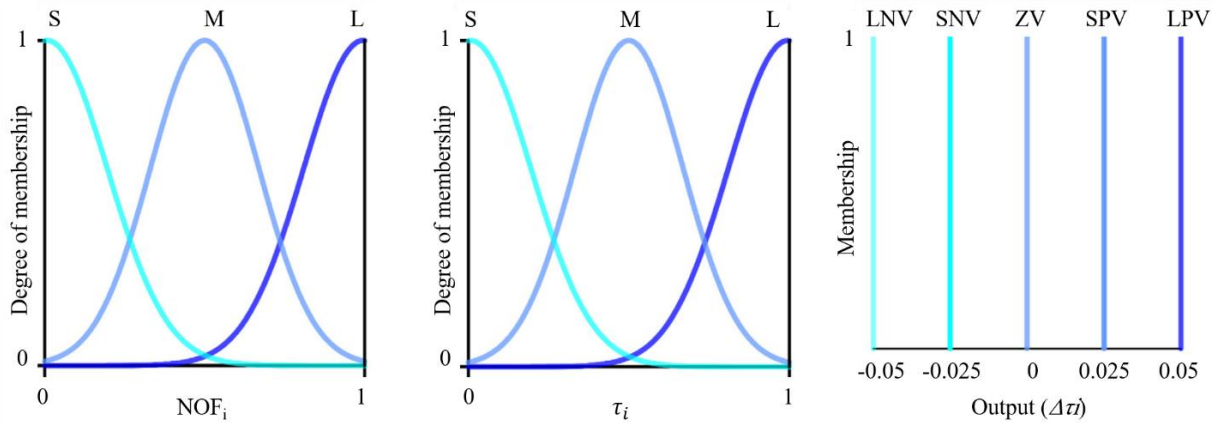


Fig. 3. Membership functions.

TABLE I. FUZZY RULES

| Rules | Inputs | | Output |
|---|---|---|---|
| | $\tau i$ | $NOF_i$ | |
| 1 | L | L | LNV |
| 2 | M | L | ZV |
| 3 | S | L | LPV |
| 4 | L | M | SNV |
| 5 | M | M | ZV |
| 6 | S | M | SPV |
| 7 | L | S | ZV |
| 8 | M | S | SPV |
| 9 | S | S | SNV |

The updated movement rule in FBOA is formulated to incorporate the virtual butterfly and the fuzzy-adjusted tendency factor:

$$X_i^{t+1} = X_i^t + q.(\tau_i^2.g^* - X_i^t) + a.r.(\tau_i^2.X^v - X_i^t) \quad (16)$$

where $g^*$ is the global best position, $X^v$ is the position of the virtual butterfly, $q$ is a random number in the range [0, 1], and $\alpha$ is the coefficient determining the impact of the virtual butterfly. The value of $\alpha$ is determined by the fitness comparison between $X^v$ and $X_i$:

$$a=1 \quad if\, f(X^v) < f(X_i), \quad a = -1 \;\; if\; f(X^v) \geq f(X_i) \quad (17)$$

The virtual butterfly $X^v$ aggregates information from the entire population using the weighted average of butterfly positions:

$$X^v = \sum_{i=1}^{M} X_i^v c_i^v \quad (18)$$

where $c_i^v$ is a normalized weight defined as:

$$c_i^v = \frac{exp(\xi_i^v)}{\sum_{j=1}^{M} exp(\xi_j^v)} \quad (19)$$

$\xi_i^v$ is the normalized fitness difference for butterfly $i$:

$$\xi_j^v = \frac{f(X_i^v) - f(X^{worst})}{f(X^{worst}) - f(g^*) + \mu} \quad (20)$$

FBOA addresses the task scheduling challenge in cloud computing environments by dynamically balancing exploration and exploitation through a fuzzy decision-making system. By adjusting the tendency factor for each butterfly using fuzzy logic, FBOA adapts its search behavior to optimize the allocation of tasks to VMs. This adaptability ensures that FBOA can efficiently handle the complex solution area of task scheduling, improving resource utilization, shortening makespan, and reducing energy usage. The integration of a virtual butterfly concept further enhances the algorithm's potential to overcome local optima, leading to more effective and balanced scheduling solutions. Through iterative adjustments, FBOA ensures that cloud resources are optimally allocated, providing better service quality and meeting the diverse demands of cloud users.

## V. Performance Evaluation

The tests were performed on an Intel Core i5-12400 system featuring a 2.50 GHz processor and 16 GB of RAM. The effectiveness of the suggested FBOA was examined using the CloudSim 3.0.3 simulation toolkit and datasets from the Heterogeneous Computing Scheduling Problem (HCSP) and Google Cloud Jobs (GoCJ). The data center comprises a single entity equipped with 12000 MIPS processing capacity. It hosts three types of machines, each with varying cores, quad-core, hexa-core, and octa-core, and supports a memory range of 512 MB to 14436 MB. For the GoCJ dataset, the configuration includes ten VMs with MIPS capacities ranging from 400 to 12000 MIPS. For the HCSP instances, 32 VMs are distributed. This setup allows for a comprehensive analysis of task scheduling performance across diverse resource capacities and task complexities.

In cloud computing, makespan is a critical metric as it measures the total time required to complete all tasks on a set of VMs. A lower makespan indicates more efficient scheduling, allowing servers to process workloads quickly. As shown in Fig. 4, FBOA effectively minimizes makespan, particularly as the number of tasks increases, through its optimized task-to-VM mapping strategy. Compared with other methods such as Whale Optimization Algorithm (WOA) [23], Random Matrix Particle Swarm Optimization (RMPSO) [24], Security- and Energy-Aware (SAEA) [25], and Genetic Algorithm with MapReduce (GAMR) [26], FBOA demonstrates the smallest average increase in makespan (4.3%), illustrating its superior scalability in handling the rising number of tasks across 19 GoCJ instances. This makes FBOA highly suitable for dynamic cloud environments where workload demands fluctuate. For HCSP tasks, a similar trend was observed, with FBOA consistently achieving a reduced makespan across varying instances, as depicted in Fig. 5. This indicates that the algorithm adapts well to heterogeneous task categories, achieving efficient resource allocation.

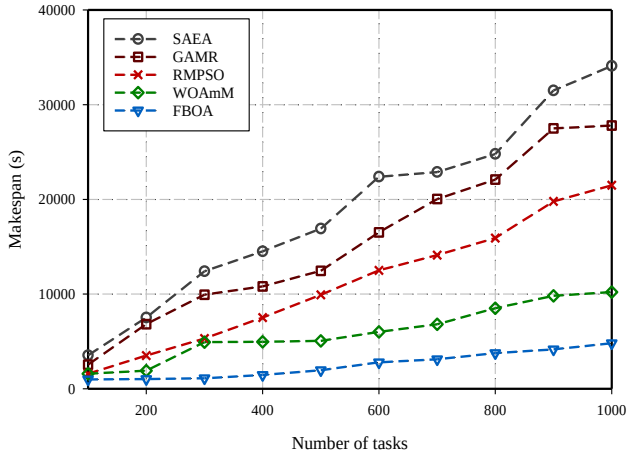


Fig. 4. Makespan for GoCJ dataset.

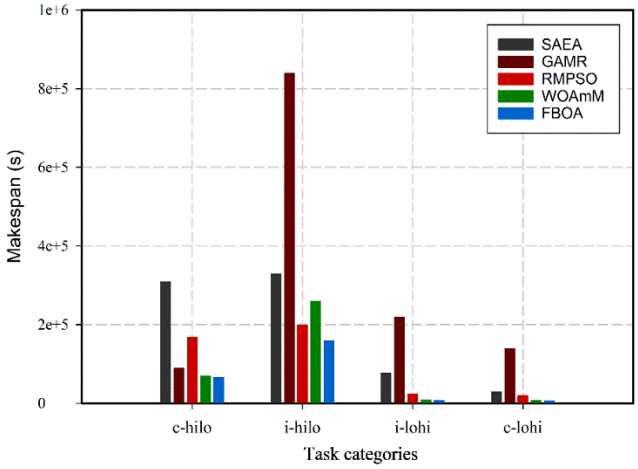


Fig. 5. Makespan for HCSP dataset.

Effective resource utilization ensures that the cloud infrastructure makes the most efficient use of available resources. FBOA achieved the best resource utilization over

other algorithms benchmarked, such as WOA, MRMPSO, and GAMR, as illustrated in Fig. 6 and Fig. 7. By dynamically adjusting its search mechanisms, FBOA efficiently managed VMs, reducing idle time and enhancing overall resource allocation. This is crucial in cloud environments where minimizing waste and optimizing VM usage can save significant costs. Among all methods, FBOA stood out for its ability to balance the workload, resulting in consistent resource usage, even with complex task distributions.

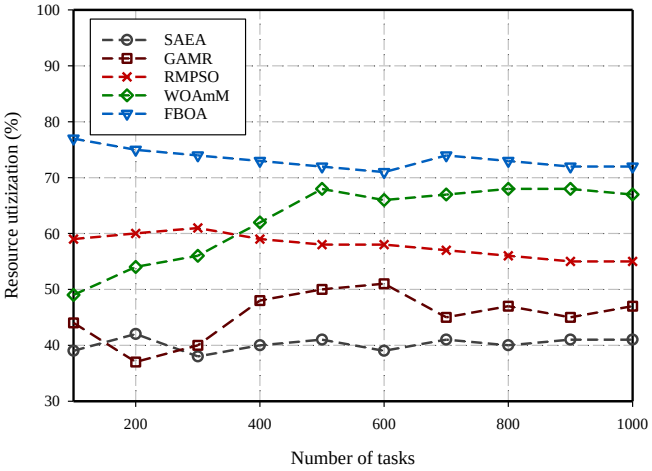


Fig. 6. Resource utilization for GoCJ dataset.

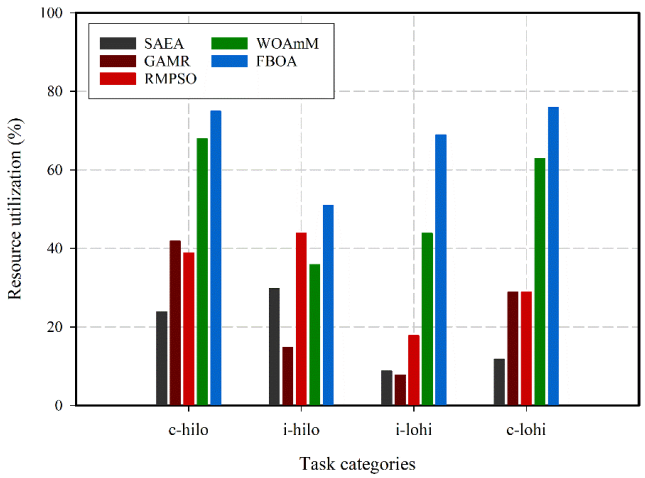


Fig. 7. Resource utilization for HCSP dataset.

Throughput, representing the number of tasks accomplished in a given period of time, serves as a measure of system efficiency. The results showed that FBOA achieved the highest throughput, which can be attributed to its parallel processing capabilities and refined task scheduling mechanism. Fig. 8 and Fig. 9 show a marked improvement in throughput for FBOA, especially when handling tasks with varying complexities. The enhanced throughput rates indicate that FBOA can efficiently manage large-scale task scheduling, making it well-suited for cloud infrastructures with fluctuating demands.

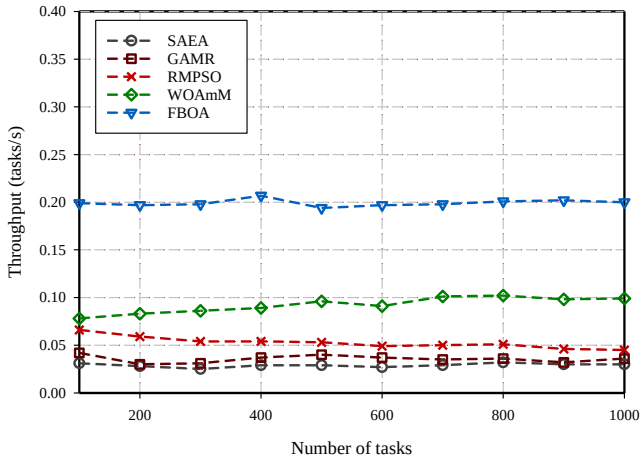


Fig. 8. Throughput utilization for GoCJ dataset.

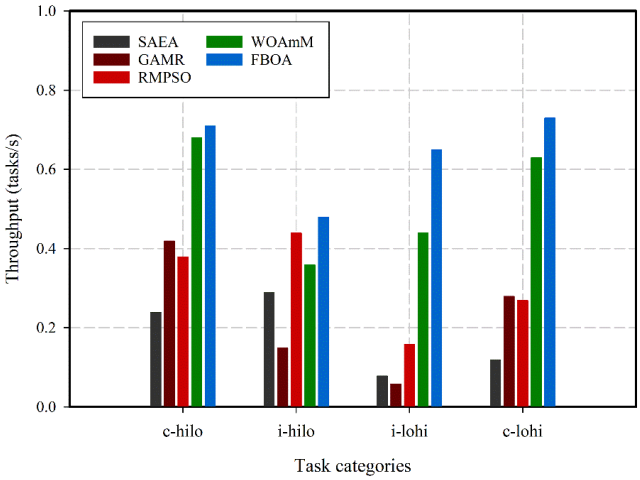


Fig. 9. Throughput utilization for HCSP dataset.

## VI. Discussion

Compared to existing alternatives, the suggested FBOA significantly improves task scheduling problems. This success in the FBOA highlights the novelties achieved in coupling enhancements, a fuzzy decision strategy, and the idea of a virtual butterfly when overcoming deficiencies in BOA.

The fuzzy decision-making mechanism dynamically balances the phases of exploration and exploitation. Such adaptability allows the algorithm to cope with complex search spaces and prevent it from converging too early, guaranteeing robust optimization even for more complicated tasks. A proper example is that FBOA improves the makespan results on both GoCJ and HCSP datasets, proving that it can efficiently map tasks into VMs with good scalability.

The introduction of the concept of the virtual butterfly enables the central agent to gather and disseminate information in the swarm regarding the ongoing search process, orienting the same toward the most promising regions. This feature enhances

the solution quality and convergence rates, which is evident in the superior performance of FBOA in terms of the minimization of resource wastage and higher throughput value. FBOA maximizes the efficiency of a cloud environment since it ensures equal workload distribution and optimal utilization of resources.

These results hold important implications for cloud computing. With reduced makespan, improved resource utilization, and higher throughput, FBOA can contribute to creating efficient and scalable task scheduling solutions. For instance, such enhancements in makespan, resource utilization, and throughput shall enable service providers to reduce costs and improve user reliability, rendering cloud computing infrastructures more viable and competitive.

Furthermore, the adaptiveness of FBOA towards dynamic workloads and heterogeneous resources makes it likely to enable dynamic and real-time applications such as IoT ecosystems and high-performance computing. Therefore, contributions in this paper open new paths for further research in advanced cloud management intelligence and thus set a roadmap for optimizations in this rapidly changing field.

These benefits of FBOA are partially outweighed by several shortcomings. First, the virtual butterfly mechanism may lead to a significant increase in computation overhead in scenarios with extremely high heterogeneity among tasks or VMs. Further, this algorithm has only been experimented with in simulations with certain datasets. The real implementation could bring in unforeseen challenges, as scalability and adaptability may arise in infrastructures that are more dynamic or involve multi-clouds.

## VII. Conclusion

In this study, we proposed FBOA to cope with the complexity of task allocation in cloud computing setups. By integrating fuzzy logic into the standard BOA, FBOA dynamically balanced exploration and exploitation, adapting to varying workload demands and resource availability. This adaptability ensured tasks were scheduled efficiently across VMs, minimizing makespan, reducing resource wastage, and improving overall system performance. The effectiveness of the proposed FBOA was validated through comprehensive simulations using the CloudSim toolkit, employing diverse datasets. The results demonstrated that FBOA consistently outperformed other metaheuristic algorithms such as WOA, RMPSO, MRMPSO, SAEA, and GAMR across critical metrics, including execution time, response time, throughput, resource utilization, and makespan. Notably, FBOA achieved lower increases in makespan, better resource allocation, and higher throughput, making it a robust and scalable solution for cloud environments. The superior performance of FBOA results from its ability to adjust the search behavior using a fuzzy decision-making mechanism and the introduction of the virtual butterfly concept, which helps avoid premature convergence and improves solution diversity. These features allow FBOA to effectively respond to the varying demands of cloud computing workloads, ensuring efficient resource use while meeting SLAs. Future research could explore further enhancements to FBOA, such as hybridizing it with other optimization techniques or applying it to emerging cloud paradigms like edge and fog computing to extend its applicability and effectiveness further.

## References

[1] B. Pourghebleh and N. J. Navimipour, "Data aggregation mechanisms in the Internet of things: A systematic review of the literature and recommendations for future research," Journal of Network and Computer Applications, vol. 97, pp. 23-34, 2017.

[2] M. Shoeibi, A. E. Oskouei, and M. Kaveh, "A Novel Six-Dimensional Chimp Optimization Algorithm—Deep Reinforcement Learning-Based Optimization Scheme for Reconfigurable Intelligent Surface-Assisted Energy Harvesting in Batteryless IoT Networks," Future Internet, vol. 16, no. 12, p. 460, 2024, doi: https://doi.org/10.3390/fi16120460.

[3] B. Pourghebleh, N. Hekmati, Z. Davoudnia, and M. Sadeghi, "A roadmap towards energy - efficient data fusion methods in the Internet of Things," Concurrency and Computation: Practice and Experience, vol. 34, no. 15, p. e6959, 2022.

[4] F. Kamalov, B. Pourghebleh, M. Gheisari, Y. Liu, and S. Moussa, "Internet of medical things privacy and security: Challenges, solutions, and future trends from a new perspective," Sustainability, vol. 15, no. 4, p. 3317, 2023.

[5] A. Rahman et al., "Impacts of blockchain in software - defined Internet of Things ecosystem with Network Function Virtualization for smart applications: Present perspectives and future directions," International Journal of Communication Systems, p. e5429, 2023.

[6] 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.

[7] 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.

[8] B. Godavarthi, N. Narisetty, K. Gudikandhula, R. Muthukumaran, D. Kapila, and J. Ramesh, "Cloud computing enabled business model innovation," The Journal of High Technology Management Research, vol. 34, no. 2, p. 100469, 2023.

[9] 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.

[10] J. Zhou et al., "Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing," Journal of cloud computing, vol. 12, no. 1, p. 85, 2023.

[11] 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.

[12] 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.

[13] M. Yadav and A. Mishra, "An enhanced ordinal optimization with lower scheduling overhead based novel approach for task scheduling in cloud computing environment," Journal of Cloud Computing, vol. 12, no. 1, p. 8, 2023.

[14] 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.

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

[16] E. Bozorgi, S. Soleimani, S. K. Alqaiidi, H. R. Arabnia, and K. Kochut, "Subgraph2vec: A random walk-based algorithm for embedding knowledge graphs," arXiv preprint arXiv:2405.02240, 2024, doi: https://doi.org/10.48550/arXiv.2405.02240.

[17] K. Dubey and S. C. Sharma, "A novel multi-objective CR-PSO task scheduling algorithm with deadline constraint in cloud computing," Sustainable Computing: Informatics and Systems, vol. 32, p. 100605, 2021.

[18] S. Mangalampalli, G. R. Karri, and A. A. Elngar, "An efficient trust-aware task scheduling algorithm in cloud computing using firefly optimization," Sensors, vol. 23, no. 3, p. 1384, 2023.

[19] T. Bezdan, M. Zivkovic, N. Bacanin, I. Strumberger, E. Tuba, and M. Tuba, "Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm," Journal of Intelligent & Fuzzy Systems, vol. 42, no. 1, pp. 411-423, 2022.

[20] Z. Wu and J. Xiong, "A novel task-scheduling algorithm of cloud computing based on particle swarm optimization," International Journal of Gaming and Computer-Mediated Simulations (IJGCMS), vol. 13, no. 2, pp. 1-15, 2021.

[21] S. Mangalampalli, G. R. Karri, and M. Kumar, "Multi objective task scheduling algorithm in cloud computing using grey wolf optimization," Cluster Computing, vol. 26, no. 6, pp. 3803-3822, 2023.

[22] F. A. Saif, R. Latip, Z. M. Hanapi, and K. Shafinah, "Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing," IEEE Access, vol. 11, pp. 20635-20646, 2023.

[23] G. Narendrababu Reddy and S. P. Kumar, "Multi objective task scheduling algorithm for cloud computing using whale optimization technique," in Smart and Innovative Trends in Next Generation Computing Technologies: Third International Conference, NGCT 2017, Dehradun, India, October 30-31, 2017, Revised Selected Papers, Part I 3, 2018: Springer, pp. 286-297.

[24] X. Tang, C. Shi, T. Deng, Z. Wu, and L. Yang, "Parallel random matrix particle swarm optimization scheduling algorithms with budget constraints on cloud computing systems," Applied Soft Computing, vol. 113, p. 107914, 2021.

[25] B. M. H. Zade, N. Mansouri, and M. M. Javidi, "SAEA: A security-aware and energy-aware task scheduling strategy by Parallel Squirrel Search Algorithm in cloud environment," Expert Systems with Applications, vol. 176, p. 114915, 2021.

[26] Z. Peng, P. Pirozmand, M. Motevalli, and A. Esmaeili, "Genetic Algorithm - Based Task Scheduling in Cloud Computing Using MapReduce Framework," Mathematical Problems in Engineering, vol. 2022, no. 1, p. 4290382, 2022.