Adaptive Honey Badger Optimization with Bernoulli Chaotic Mapping and Decreasing Neighborhood for Efficient Cloud Task Scheduling

Yanfang XING
School of Media Technology, Communication University of China Nanjing,
Nanjing City, Jiangsu Province, 211172, China

***Abstract*—One of the most important challenges in optimizing task scheduling in cloud computing is the dynamic nature of resources, the heterogeneity of tasks, and conflicting optimization criteria, such as minimizing task completion period, optimizing resource utilization, and shortening task migration time. Meta-heuristics such as the Honey Badger Algorithm (MBA) often converge on suboptimal solutions and may not strike the perfect balance between exploitation and exploration in task scheduling optimization. To address the problems associated with traditional HBA and similar algorithms, this research introduces a novel optimization technique called the Multi-strategy Honey Badger Algorithm (MHBA). The proposed MHBA integrates three optimization strategies: horizontal crossing coupled with adaptation, an optimum decreasing neighborhood, and a Bernoulli shift scheme. The MHBA is simulated using CloudSim and compared with other advanced techniques. The experimental findings confirm MHBA's efficacy in reducing makespan by up to 25.6%, increasing resource utilization by up to 16.7%, and decreasing migration time by up to 27.5% when applied to the HPC2N dataset. The same was evident in the NASA dataset, where MHBA achieved reductions in makespan of up to 25.9%, improvements in resource utilization of up to 21.6%, and reductions in migration time of up to 24.1%.**

Keywords*—Cloud computing; multi-objective task scheduling; honey badger; Bernoulli chaotic mapping; exploration–exploitation balance

I. INTRODUCTION

Cloud computing is a revolutionary technological shift towards a new paradigm for scalable and on-demand provision of various services through virtualization [1]. Today, organizations increasingly use cloud computing to process large volumes of data, run distributed applications, and offer elastic computing services to users worldwide [2]. In this context, task scheduling is a fundamental issue for mapping various computational tasks to a set of Virtual Machines (VMs) [3]. This factor is crucial to determining a system's performance. In a heterogeneous environment with changing scenarios, scheduling mechanisms must effectively handle multiple tasks while ensuring optimal resource allocation [4]. Therefore, the development of intelligent scheduling algorithms is a prominent issue for cloud computing systems, as it affects the efficiency and cost-effectiveness of a cloud infrastructure [5].

However, scheduling in cloud computing remains a complex optimization problem, as many tasks must be allocated to heterogeneous VMs while optimizing multiple conflicting objectives [6]. Some common scheduling objectives in cloud computing include minimizing makespan, maximizing resource utilization, and diminishing task migration overhead. These objectives are conflicting, and it is really hard to achieve optimal performance across all of them. In addition, as the number of tasks and resources increases, the scheduling process becomes increasingly complex. In cloud computing, task scheduling is known to be an NP-complete problem. In addition, the dynamic nature of the cloud computing environment, particularly in resource provisioning, makes it a complex task that requires adaptive optimization techniques to achieve near-optimal performance.

Several strategies have been proposed to overcome these issues, including heuristics and more sophisticated techniques known as metaheuristics, inspired by natural processes. In fact, algorithms such as Differential Evolution (DE) [7], Genetic Algorithm (GA) [8], Particle Swarm Optimization (PSO) [9], and Ant Colony Optimization (ACO) [10] have proven effective for optimization problems. Nevertheless, many of them possess severe limitations in solving large-scale scheduling problems in cloud computing environments. For instance, traditional metaheuristics may suffer from premature convergence, in which a local optimum is found before a global one. Furthermore, many existing approaches may struggle to establish an appropriate equilibrium between exploration and exploitation throughout the search process.

These limitations emphasize the importance of developing more powerful and flexible optimization strategies to effectively explore the search domain and possess strong local search capabilities. Recently, various advances in swarm-based optimization methodologies have been proposed to improve the efficiency of optimization algorithms by combining multiple improvement strategies. Among such strategies, the Honey Badger Algorithm (HBA) has been found to possess strong global search capabilities and simplicity [11]. However, like other metaheuristic optimization techniques, the traditional HBA may face limitations, such as insufficient population diversity and low adaptability in the later stages of optimization. Hence, motivated by such limitations, this research aims to introduce a novel enhanced version of the Multi-strategy Honey Badger Algorithm (MHBA) to improve diversity, maintain adaptive exploration–exploitation strategies, and efficiently solve multi-target task scheduling problems in a cloud computing setting. The main contribution of this research can be summarized as follows:

- Proposing MHBA to address the multi-target task scheduling issue in cloud computing.
- Introducing the Bernoulli chaotic mapping to optimize the population diversity in the initialization phase.
- Suggesting the Optimum Decreasing Neighborhood (ODN) approach to optimize search space exploration.
- Incorporating the Horizontal Crossing coupled with Adaptation (HCA) mechanism to maintain a balanced exploration–exploitation process.

The remaining part of this study is presented as follows. Section II reviews recent research on cloud task scheduling and optimization algorithms. Section III describes the system model and mathematical description of the multi-target task scheduling problem. Section IV introduces the proposed algorithm and its application to a cloud task scheduling system. Section V presents the experimental evidence and compares the proposed method with other algorithms. Finally, in Section VI, the study is summarized, and future research directions for intelligent cloud task scheduling are proposed.

II. Related Work

Task scheduling is a major concern in cloud computing due to its effects on system performance, resource allocation, and quality of service. Several heuristics and metaheuristic methods have been developed to address the task scheduling problem due to its inherent NP-hard nature. Some of the works include the following. In their study, Pirozmand, et al. [12] present the Improved Particle Swarm Optimization (IPSO) as a scheduling algorithm in cloud computing. The proposed IPSO algorithm uses a multi-adaptive learning approach, with two particle types: ordinary particles and locally best particles, to ensure efficient convergence.

In another work, Gong, et al. [13] proposed the Enhanced Marine Predator Algorithm (EMPA) as a cloud computing task scheduler. The EMPA algorithm develops a scheduling model that optimizes for both minimum makespan and maximum resource utilization. The algorithm employs Whale Optimization operators, a nonlinear inertia-weight coefficient, and a golden sine strategy to enhance search capabilities and convergence performance. In a different study, Pabitha, et al. [14] developed Chameleon and the Remora Search Optimization Algorithm (CRSOA) as a multistage cloud computing task scheduler.

Selselejoo and Ahmadifar [15] introduce the DT-GWO, a hybrid scheduling technique that integrates the Decision Tree (DT) classifier with the Grey Wolf Optimization (GWO). Emara, et al. [16] suggest a novel adaptive task scheduling approach that uses the enhanced Harris Hawks Optimization (HHO) method. The method incorporates several enhancements, including selecting the best solution rather than a random one in the exploration phase and introducing mutation operations in the exploitation phase to prevent convergence to local optima.

Du, et al. [17] enhance the Reptile Search Algorithm (RSA) by proposing the ACRL-RSA, where adaptive chaotic reverse learning strategies are utilized. Population diversity is improved by adopting chaotic mapping and enhanced exploration capabilities in the optimization process. Experimentation conducted using the CloudSim toolkit revealed that ACRL-RSA performs better in minimizing makespan, computation cost, and energy expenditure. Kathole, et al. [18] introduce the HL-GOA as an optimization technique for task scheduling and VM placement in cloud computing. The HL-GOA model considers several factors, including cost, time, throughput, resource utilization, and makespan.

Awad, et al. [19] suggest a combined PSO-GWO-based scheduling approach for workflow management in fog and cloud computing systems. Tawfeek, et al. [20] introduce a novel artificial bee colony-genetic algorithm-based framework for multi-objective cloud workflow scheduling. The push-pull algorithm for task scheduling in cloud computing is introduced by Rajeshwari and Namratha [21]. This scheduling approach assigns arriving tasks to VMs using the push-pull model.

TABLE I. An Overview of Existing Approaches

Algorithm	Key technique	Objectives	Advantages	Limitations
IPSO [12]	Multi-adaptive PSO learning strategy	Makespan and load balancing	Faster convergence than PSO	Risk of local optimum
EMPA [13]	Improved marine predator algorithm with WOA operators	Makespan and resource utilization	Improved exploration capability	Parameter sensitivity
CRSOA [14]	Chameleon and remora hybrid optimization	Makespan, cost, and load balancing	Handles uncertainty factors	Increased computational complexity
DT-GWO [15]	Decision tree and grey wolf optimization	Makespan, cost, and resource utilization	Improved task classification	Limited scalability
Enhanced HHO [16]	Adaptive Harris hawks optimization	Throughput and makespan	Better exploration and exploitation	Mutation tuning required
RSA-ACRL [17]	Chaotic reverse learning in RSA	Makespan, energy, and cost	Increased population diversity	Higher computational overhead
HL-GOA [18]	Lemur-gannet hybrid optimization	Cost, throughput, and makespan	Multi-constraint optimization	Algorithm complexity
IPSO-GWO [19]	Hybrid PSO-GWO	Cost, energy, and makespan	Balanced search strategy	Limited adaptability
ABC-GA [20]	Artificial bee colony and genetic algorithm	Time, cost, and energy	Strong hybrid search capability	Computational cost
LPP [21]	Linked list push-pull scheduling	SLA and response time	Effective load balancing	Limited global optimization ability

Despite the development of several metaheuristic algorithms for effective task scheduling in cloud computing environments, several limitations remain to be addressed. For example, as shown in Table I, most algorithms are designed to optimize specific aspects of the problem, while the multi-objective nature is not adequately addressed. In addition, some algorithms suffer from early convergence and insufficient population diversity, making it difficult to explore the solution space. It is because of such challenges that hybrid algorithms were developed to address weaknesses in other algorithms.

Another major limitation evident in most algorithms is the lack of an effective trade-off between exploration and exploitation in the optimization process. Most optimization algorithms have strong exploratory capabilities but fail to adequately explore the search space because of poor local search refinement strategies. However, optimization algorithms with effective local search strategies tend to converge to poor-quality solutions. Moreover, most optimization algorithms used in multi-objective optimization problems fail to incorporate adaptive search strategies that can change their behavior at different stages of the optimization process. To address this issue in multi-objective cloud task scheduling problems, this study proposes MHBA, which improves population diversity and adapts the crossover strategy during search.

III. PROBLEM FORMULATION

A. Cloud Computing System Architecture

Cloud computing is a computing model that allows users to access shared computing resources over the internet without owning their own infrastructure. Service providers offer users computing resources, such as storage capacity, software applications, databases, and data analysis capabilities, through a network-based computing model known as the cloud. Benefits of cloud computing include elasticity, availability, scalability, and reliability, among others. Cloud systems can take various forms based on operational requirements and security considerations. These include public, private, and hybrid clouds. Furthermore, cloud computing services are typically provided through a layered architecture comprising infrastructure-as-a-service, platform-as-a-service, and software-as-a-service. As shown in Fig. 1, the layers include infrastructure as a service, which provides the bottom layer of basic computing infrastructure; platform as a service; and software as a service.

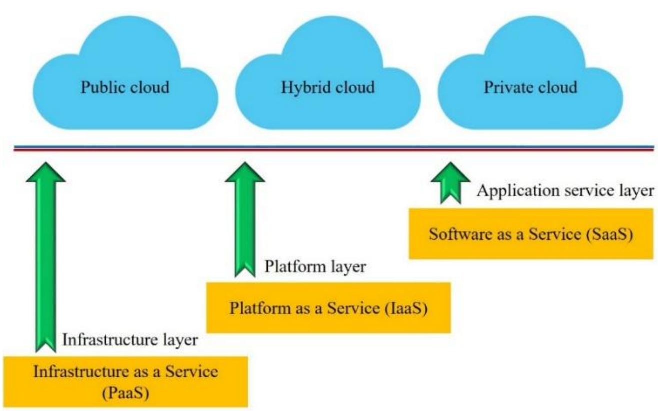

Fig. 1. Layered service architecture for cloud environments.

In practical cloud systems, task scheduling involves migrating computational workloads from user devices to large-scale cloud data centers. The objective is to ensure that tasks are processed efficiently while meeting user requirements for performance and service quality. Due to continuously changing workloads and resource conditions, scheduling mechanisms must be highly adaptable and capable of dynamically adjusting allocation strategies. Consequently, an effective scheduling system must consider factors such as system performance, resource constraints, task priorities, and user requirements to achieve efficient resource utilization. The architecture for service delivery in a cloud computing model is shown in Fig. 2 below. The architecture has several layers that process user requests and manage computational resources. At the user end, several clients simultaneously submit their work requests to the cloud computing system. These requests are handled by the task management layer.

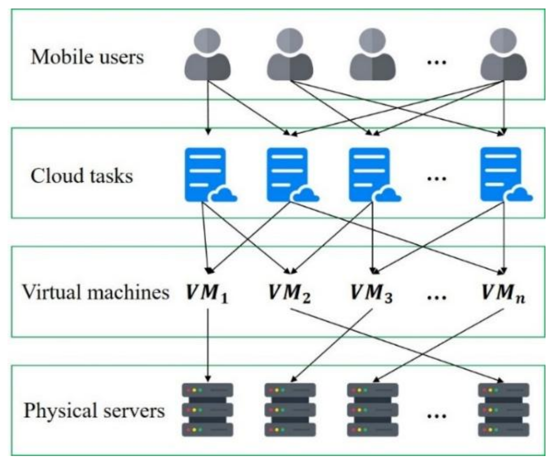

Fig. 2. Layered service delivery architecture in cloud computing.

To achieve this objective, the scheduling algorithm must consider these characteristics when allocating resources. The cloud computing architecture typically consists of multiple VMs running on physical computers. Computing resources can be allocated according to the system's needs. The resource manager keeps track of the current state of each computing resource, whether it is busy, idle, or faulty. Given the above facts and the chosen scheduling algorithm, the jobs are allocated to VMs. The performance analysis of the job, like its execution time, will be delivered to the user. Task scheduling is an important process in a cloud computing system. As shown in Fig. 3, the scheduling process typically comprises two main layers: the scheduling control layer and the task execution layer. First, users' submitted jobs are divided into individual tasks. Then, the scheduler evaluates all tasks to determine their relationships with computing resources. Resource allocation is then performed based on the task's needs.

Once the scheduling process is complete, the execution process is sent to the task-processing layer, where computing nodes perform the operations. While executing, the scheduler continuously monitors task status and adjusts allocations as needed. At the end of the execution process, the computation results are then returned to the user. Effective scheduling processes enable the cloud computing environment to work effectively.

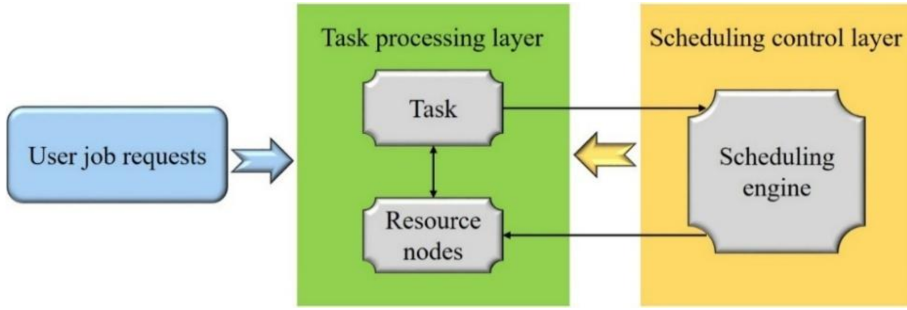

Fig. 3. Conceptual workflow of the cloud task scheduling mechanism.

B. Task and Resource Modeling

In cloud computing task scheduling optimization, the primary objective is to determine an optimal mapping between computational tasks and available computing resources. To formally describe the scheduling process, a mathematical model is constructed that captures the relationship between tasks and computing nodes. Let the cloud environment consist of a set of computing nodes $R = \{R_1, R_2, \dots, R_m\}$ and a set of tasks $T = \{T_1, T_2, \dots, T_n\}$, where m denotes the number of VMs, and n represents the number of submitted tasks, typically satisfying $n > m$. The scheduling decision can be represented using an assignment matrix as follows:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} \tag{1}$$

where, a_{ij} is a binary decision variable indicating whether the task T_i is assigned to the resource R_j. If $a_{ij} = 1$, task T_i is executed on the j^{th} VM; otherwise, the value is zero. To accurately describe scheduling performance, three fundamental metrics are considered: makespan, resource utilization, and migration time. Makespan represents the completion time of the final task in the scheduling sequence, calculated using Eq. (2).

$$M = \max_{i \in \{1,\dots,m\}} C_i \tag{2}$$

where, C_i signifies the finishing time of the task T_i. Resource utilization quantifies the proportion of computational resources effectively used by scheduled tasks, defined as follows:

$$U = \frac{\sum_{j=1}^{n} \sum_{i=1}^{m} R_{ij} \cdot y_{ij}}{\sum_{j=1}^{n} R_j} \tag{3}$$

where, R_{ij} denotes the resource requirement of the task T_i on VM j, y_{ij} is a binary decision variable indicating whether the task T_i is assigned to VM j, and R_j represents the total capacity of VM j. Migration time captures the overhead incurred when tasks are transferred between VMs, expressed by Eq. (4).

$$T_{mig} = \sum_{i=1}^{m} \sum_{j=1}^{n} y_{ij} \cdot o_{ij} \tag{4}$$

where, o_{ij} denotes the migration duration required for the task T_i to move to VM j. To evaluate scheduling quality, a composite fitness function is formulated by combining the three metrics. The overall optimization objective is defined by Eq. (5).

$$F = \omega_1 \left(\frac{1}{M}\right) + \omega_2 U - \omega_3 T_{mig} \tag{5}$$

where, ω_1, ω_2, and ω_3 represent the weighting coefficients that determine the importance of each objective.

IV. Proposed Algorithm

HBA operates on a population and mimics honey badger behavior when hunting in the wild. This algorithm consists of two main predation phases: the digging and honey phases. The latter allows the search technique to perform both exploration and exploitation to solve problems effectively. In nature, honey badgers collaborate with honey-guiding birds, as the latter are responsible for detecting bees' nests. Honey guides find the nest; however, honey badgers break open the hive with their claws. This cooperation forms the foundation for the algorithm. During the search process, potential solutions mimic honey badgers' prey-hunting behavior by scent-tracking and following clues.

A. Population Initialization

As with any other population-based algorithm, the search procedure begins with an initialization stage that generates candidate solutions. Let us assume that there are N_p honey badgers in the population. The location of the k^{th} badger is randomly generated from the search space using Eq. (6).

$$Y_k = L_b + (U_b - L_b) \cdot \rho, \quad k = 1,2,\dots,N_p \tag{6}$$

where, L_b and U_b represent the lower and upper bounds of the search space, respectively, while ρ is a uniformly distributed random number in the range [0,1]. The initialization process ensures that candidate solutions are uniformly distributed across the search space.

B. Digging Phase (Exploration)

During the digging phase, honey badgers detect prey using their powerful olfactory capabilities. This phase promotes global search, enabling the algorithm to find prey across extensive areas. The position updating equation used in this phase is given by Eq. (7).

$$y_i = y^* + G \cdot \kappa \cdot \Psi \cdot y^* + G \cdot r_2 \cdot \delta \cdot \Delta_i |\cos(2\pi r_3)[1 - \cos(2\pi r_4)]| \tag{7}$$

where, y^* represents the current optimal solution in the population, κ is a constant with a minimum value of 1, $r_2, r_3,$

and r_4 represent randomly generated numbers within [0,1], Δ_i denotes the distance between the current solution and the current optimal solution, and G serves as the directional parameter, regulating the search orientation. The digging phase includes an intensity operator that calculates the honey badger's attraction to the prey. This intensity is determined using Eq. (8).

$$\Psi = \rho_1 \cdot \frac{Q}{4\pi\Delta_i^2} \tag{8}$$

where, ρ_1 is a random number within [0,1] and Q represents the source strength. The source strength and distance term are computed as follows:

$$\mathrm{Q} = (y_i - y_{i+1})^2 \tag{9}$$

$$\Delta_i = y^* - y_i \tag{10}$$

To control the search direction, a directional control parameter is employed during the digging phase and is defined by Eq. (11).

$$G = \begin{cases} 1, & r_5 \leq 0.5 \\ -1, & \text{otherwise} \end{cases} \tag{11}$$

where, r_5 represents a randomly generated number within the interval of [0,1]. If G is positive, the direction of movement will be towards the best solution; otherwise, it will be away from it. Moreover, a density parameter is used to control the gradual shift of the search mode from exploration to exploitation, decreasing nonlinearly with time and given by Eq. (12).

$$\delta = \theta \cdot \exp\left(-\frac{t}{T_{max}}\right) \tag{12}$$

where, θ is a constant usually set to 2, t is the current iteration number, and T_{max} denotes the maximum number of iterations. The adaptive parameter enables the algorithm to smoothly transition between global search and local refinement.

C. Honey Phase (Exploitation)

The next step of the algorithm models the interaction between the two species as cooperation. At this stage, the honey guides help locate the hive, and then the honey badger goes to the site to exploit the resources. The updating process at the honey stage can be mathematically defined by Eq. (13). It shows that the candidates approach the best-known location in a controlled random manner.

$$y_i = y^* + G \cdot r_6 \cdot \delta \cdot \Delta_i \tag{13}$$

D. Multi-Strategy Honey Badger Algorithm

While HBA shows excellent exploratory ability, it may be prone to an imbalance between exploration and exploitation. To address this drawback of the initial HBA, an improved approach, MHBA, will be introduced in this research. As the name suggests, this algorithm uses three improvement strategies: chaotic Bernoulli initialization, an optimal piecewise decrease strategy, and horizontal crossover. Chaotic maps produce deterministic sequences with high randomness, thereby enhancing population diversity during initialization. In the MHBA algorithm, instead of random initialization, the chaotic Bernoulli map is used. Eq. (14) defines the Bernoulli map.

$$z_{t+1} = \begin{cases} \dfrac{z_t}{1-\varepsilon}, & 0 \leq z_t < 1-\varepsilon \\ \dfrac{z_t-(1-\varepsilon)}{\varepsilon}, & 1-\varepsilon \leq z_t \leq 1 \end{cases} \tag{14}$$

where, ε is a control parameter within the range [0,1]. A commonly used simplified form of the Bernoulli map is $z_{t+1} = 2z_t \bmod 1$ that generates a chaotic sequence used for initializing candidate solutions. To improve the adaptability of search behavior, MHBA employs the ODN technique. It controls the neighborhood search radius throughout the optimization process. The ODN parameter is expressed as:

$$ODN = \begin{cases} 1 + \tau_1 \cdot \eta_1 \cdot U(0,1), & 0 \leq t \leq \dfrac{T_{max}}{3} \\ 1 + \tau_2 \cdot \eta_2 \cdot U(0,1), & \dfrac{T_{max}}{3} < t \leq \dfrac{2T_{max}}{3} \\ 1 + \tau_3 \cdot \eta_3 \cdot U(0,1), & \dfrac{2T_{max}}{3} < t \leq T_{max} \end{cases} \tag{15}$$

Three nonlinear decline functions regulate the neighborhood size as follows:

$$w_1 = w_{max} - (w_{max} - w_{min})\left(\frac{t}{T_{max}}\right)^2 \tag{16}$$

$$w_2 = w_{max} - (w_{max} - w_{min})\frac{t}{T_{max}} \tag{17}$$

$$w_3 = w_{max} - (w_{max} - w_{min})\left[\frac{2t}{T_{max}} - \left(\frac{t}{T_{max}}\right)^2\right] \tag{18}$$

Early iterations emphasize exploration while later iterations focus on exploitation. By integrating ODN, the updated position in the digging phase becomes:

$$y_i = y^* \cdot ODN + G \cdot \kappa \cdot \Psi \cdot y^* + G \cdot r_2 \cdot \delta \cdot \Delta_i |\cos(2\pi r_3)(1 - \cos(2\pi r_4))| \tag{19}$$

Similarly, the honey stage is modified as follows:

$$y_i = y^* + ODN + G \cdot r_6 \cdot \delta \cdot \Delta_i \tag{20}$$

To enhance the exploration process, MHBA uses a crossover technique based on a swarm intelligence algorithm. This crossover creates new candidates by merging the two parents' information to create children's solutions using Eq.

(21) and (22).

$$y_{new}(i) = \begin{cases} c_1 y_i + (1-c_1) y_j + c_2(y_i - y_j), & \mathrm{CSO/rand/1} \\ y_i + c_3(y_i - y_j), & \mathrm{DE/rand/1} \\ y^* + c_4(y_i - y_j), & \mathrm{DE/best/1} \\ y_i + c_5(y^* - y_i) + c_6(y_i - y_j), & DE/rand-to-best/1 \end{cases} \tag{21}$$

$$y_{new}(j) = \begin{cases} c_1 y_i + (1 - c_1) y_j + c_2(y_j - y_i), & \text{CSO/rand/1} \\ y_j + c_3(y_j - y_i), & \text{DE/rand/1} \\ y^* + c_4(y_j - y_i), & \text{DE/best/1} \\ y_j + c_5(y^* - y_j) + c_6(y_j - y_i), & DE/rand-to-best/1 \end{cases} \quad (22)$$

The success ratio of each crossover technique is recorded, and the probability of selecting each technique is adaptively adjusted during the search process. The flowchart of the MHBA is depicted in Fig. 4. The MHBA algorithm begins by initializing a population using the Bernoulli chaotic mapping. During the search process, the algorithm alternates between the digging and honey stages using the PDN neighborhood approach. A horizontal crossover with adaptive selection of crossover techniques is employed to improve exploration. The search process stops when the stopping criterion is met; the optimal solution is yielded as the optimal schedule.

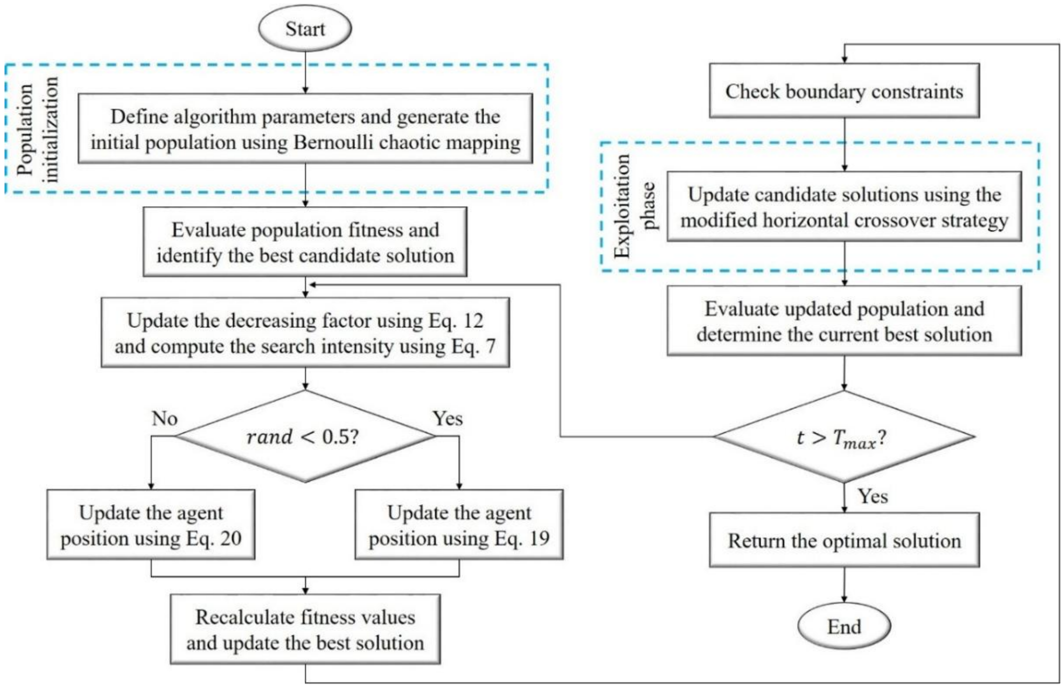

Fig. 4. Flowchart of the proposed algorithm.

V. Results and Discussion

To test the efficacy of the proposed MHBA for task scheduling in cloud computing, three critical parameters are used: makespan, resource utilization, and migration time. These parameters give a complete picture of how effective a task scheduling algorithm is in cloud computing. Makespan is defined as the total time required to execute all tasks scheduled in a system. A low makespan indicates that tasks are executed efficiently and that the task scheduling algorithm is effective. Resource utilization is the ratio of computational resources used to execute tasks to the total computational resources available in a system. If resource utilization is high, it indicates that a task scheduling algorithm is effective and that computational resources are being used efficiently in the system. Migration time is the total time required to migrate tasks from one VM to another within a system that executes tasks using a task scheduling algorithm. A low migration time implies that a task scheduling algorithm is effective.

The performance of the proposed MHBA-based scheduling framework is evaluated through simulation experiments using the CloudSim toolkit, which models and simulates the cloud computing environment. CloudSim provides a simulation framework for emulating complex cloud computing environments, including VMs, hosts, data centers, and task workloads, without requiring actual hardware. The simulation framework provides a wide range of cloud computing components, such as VM allocation, scheduling policies, and resource provisioning mechanisms. This provides a wide range of analysis capabilities for scheduling strategies in various cloud computing environments. To improve simulation reliability and realistic workload modeling, two workload traces, namely HPC2N and NASA, were used in simulation experiments. These workload traces contain detailed workload information regarding the execution behavior of jobs.

The HPC2N dataset is derived from a high-performance computing environment and comprises a heterogeneous workload analogous to large-scale scientific computing activities in a distributed computing environment. NASA is another dataset derived from scientific workflows executed in real computing environments. It comprises details on task execution time, task dependencies, and computational requirements. Using these datasets in the simulation enables the evaluation mechanism to capture realistic scheduling scenarios and workload diversity, which are common in cloud computing environments. By using the HPC2N and NASA datasets in the experiments, the MHBA algorithm is validated for its ability to handle diverse and dynamic task-scheduling scenarios. The configuration parameters utilized in the simulation environment for the cloud computing system are presented in Table II.

TABLE II. CONFIGURATION PARAMETERS USED IN THE CLOUD SIMULATION ENVIRONMENT

Simulation parameter	Assigned value
Total virtual machine instances	50
Processing capability of each VM	100 – 1000
Memory allocated per VM	1024 MB
Virtual machine memory capacity	9192 MB
Number of cloud data centers	2
Total number of submitted tasks	100 – 1000
Task workload size (instruction length)	1500 – 10,000
Virtualization platform	Xen

The makespan performance of MHBA has been tested against two popular swarm-based optimization techniques: PSO and Cat Swarm Optimization (CSO). Fig. 5 presents a visual representation of the results, and Table III summarizes the results for the HPC2N workload set. From 100 to 1000 cloudlets, all makespan values increase, indicating increased computational complexity. MHBA has been found to possess lower makespan values than the other techniques. When scheduling 100 cloudlets, MHBA achieves a makespan of 36.85, which is reduced by 25.5%, 34.57%, and 21.57% compared with PSO, CSO, and HBA, respectively. When the workload is increased to 500 cloudlets, MHBA still achieves the best performance, with a makespan of 135.18, which is reduced by 12.15%, 20.66%, and 8.9% compared with PSO, CSO, and HBA, respectively. When the workload is increased to 1000 cloudlets, MHBA still outperforms the other algorithms, achieving a makespan time of 247.25, which is reduced by 9.75%, 17.54%, and 7.34. Various parts of MHBA contribute to this improvement. For instance, the random initialization process enhances population diversity, ensuring the algorithm can explore all feasible scheduling options from the start.

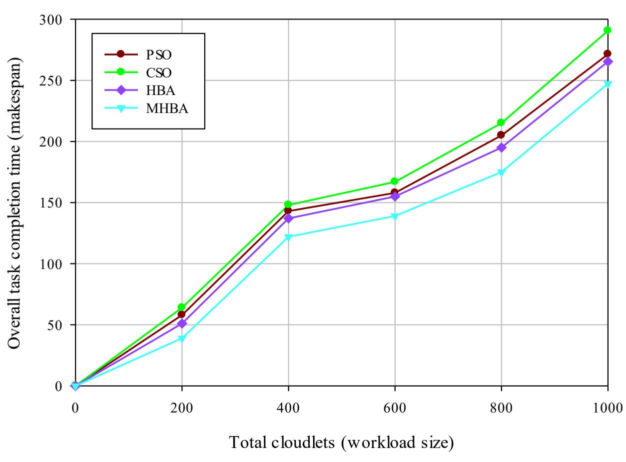

Fig. 5. Comparison of makespan for different scheduling algorithms under the HPC2N workload with varying numbers of cloud tasks.

The resource utilization comparison is shown in Fig. 6 and summarized in Table IV. As the number of tasks increases, the utilization rates of all algorithms decrease slightly. This is because the number of tasks is distributed among the available resources. Though MHBA has lower utilization for 100 cloudlets, the performance is still acceptable. This is because MHBA has strong exploration capabilities during the initial search phase. For smaller-scale tasks, the search space is relatively simple, and other algorithms, such as PSO and CSO, can achieve convergence.

TABLE III. COMPARISON OF MAKESPAN ACROSS SCHEDULING ALGORITHMS USING THE HPC2N DATASET

Workload size (Cloudlets)	PSO	CSO	HBA	MHBA
100	46.25	49.59	41.80	36.85
500	151.61	163.11	147.22	135.18
1000	271.38	290.63	265.42	247.25

MHBA has to spend some extra time exploring the possible scheduling configurations. MHBA outperforms other algorithms on complex tasks. For example, when the number of cloudlets is 500, MHBA has a better value, i.e., 2.42, for the resource utilization. This is because MHBA outperforms PSO, CSO, and HBA by 16.52%, 7.02%, and 8.67%, respectively. For 1000 cloudlets, MHBA outperforms other algorithms, achieving a resource utilization of 2.15. This is because MHBA outperforms PSO, CSO, and HBA by 16.74%, 13.95%, and 12.09%, respectively.

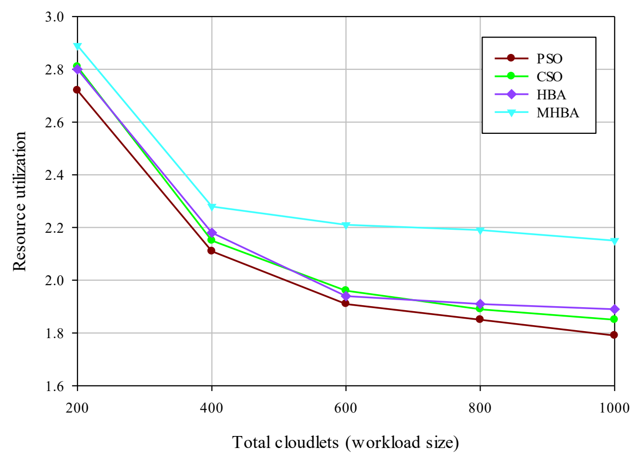

Fig. 6. Comparison of resource utilization for different scheduling algorithms under the HPC2N workload with varying numbers of cloud tasks.

TABLE IV. COMPARISON OF RESOURCE UTILIZATION ACROSS SCHEDULING ALGORITHMS USING THE HPC2N DATASET

Workload size (Cloudlets)	PSO	CSO	HBA	MHBA
100	2.94	3.03	2.97	3.15
500	2.02	2.25	2.21	2.42
1000	1.79	1.85	1.89	2.15

Fig. 7 and Table V show the migration time results. As the number of cloudlets increases (i.e., as system load increases), migration time increases for all algorithms. However, MHBA outperforms PSO, CSO, and HBA in terms of migration time. At 100 cloudlets, MHBA attains a migration time of 29.83, which is 28.76% lower than PSO, 38.08% lower than CSO, and 17.49% lower than HBA. At 500 cloudlets, MHBA attains 112.65, outperforming PSO and CSO by 11.2% and 20.6%, respectively. At 1000 cloudlets, MHBA again outperforms PSO, CSO, and HBA in terms of migration time, achieving 235.87, which is 15.16% lower than PSO, 27.89% lower than CSO, and 11.21% lower than HBA. This is because MHBA can identify effective scheduling configurations early in the optimization process. Chaotic population initialization, dynamic search strategies, and strategy-based crossover in MHBA enable it to converge to effective task assignments early in the optimization process.

In addition to the HPC2N dataset, the proposed MHBA algorithm was further evaluated using the NASA workload traces. The performance results obtained using this dataset are shown in Fig. 8-10 and Table VI-VIII. Based on experimental analysis across various datasets, it is verified that the proposed MHBA algorithm achieves better scheduling efficiency across multiple performance metrics. In particular, the algorithm maintains a stable performance with heterogeneous tasks and varying resource requirements.

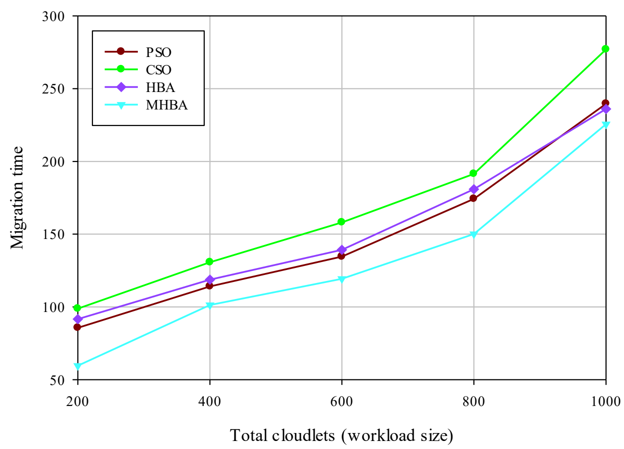

Fig. 7. Comparison of migration time for different scheduling algorithms under the HPC2N workload with varying numbers of cloud tasks.

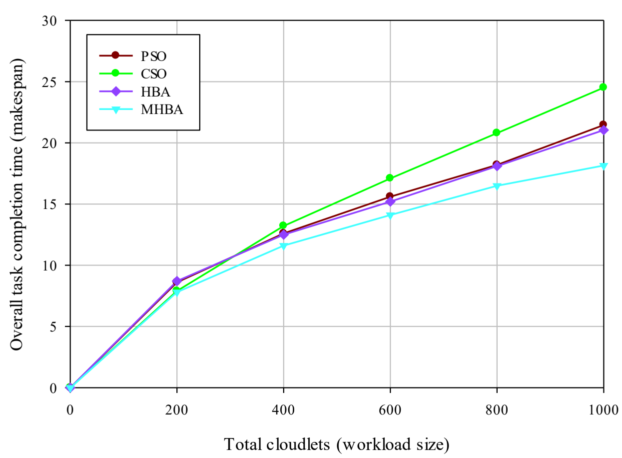

Fig. 8. Comparison of makespan for different scheduling algorithms under the NASA workload with varying numbers of cloud tasks.

TABLE V. COMPARISON OF MIGRATION TIME ACROSS SCHEDULING ALGORITHMS USING THE HPC2N DATASET

Workload size (Cloudlets)	PSO	CSO	HBA	MHBA
100	38.41	41.19	35.05	29.83
500	126.85	140.88	122.50	110.15
1000	239.69	277.01	236.12	225.66

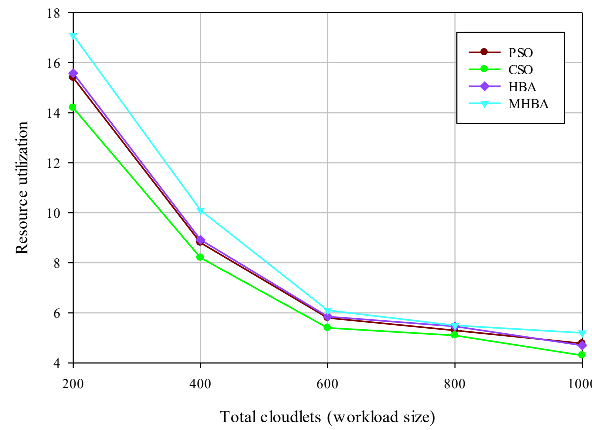

Fig. 9. Comparison of resource utilization for different scheduling algorithms under the NASA workload with varying numbers of cloud tasks.

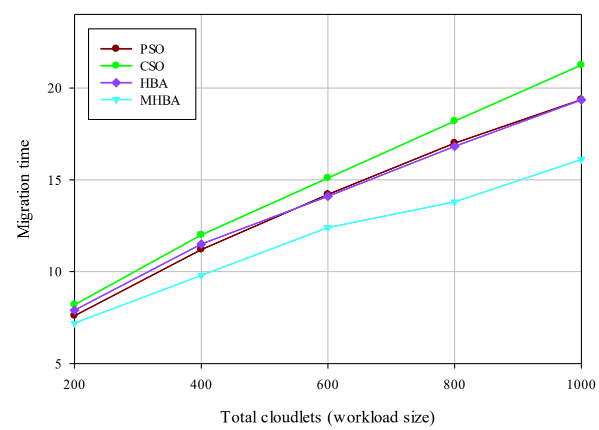

Fig. 10. Comparison of migration time for different scheduling algorithms under the NASA workload with varying numbers of cloud tasks.

TABLE VI. COMPARISON OF TASK MAKESPAN ACROSS SCHEDULING ALGORITHMS USING THE NASA DATASET

Workload size (Cloudlets)	PSO	CSO	HBA	MHBA
100	6.85	7.04	6.91	6.55
500	14.18	15.19	14.22	12.91
1000	21.45	24.51	21.05	18.15

TABLE VII. COMPARISON OF RESOURCE UTILIZATION ACROSS SCHEDULING ALGORITHMS USING THE NASA DATASET

Workload size (Cloudlets)	PSO	CSO	HBA	MHBA
100	18.42	16.80	18.85	21.45
500	6.18	6.25	6.31	6.61
1000	4.78	4.31	4.76	5.28

TABLE VIII. COMPARISON OF MIGRATION TIME ACROSS SCHEDULING ALGORITHMS USING THE NASA DATASET

Workload size (Cloudlets)	PSO	CSO	HBA	MHBA
100	5.92	6.78	5.98	5.71
500	13.06	13.67	13.02	11.20
1000	19.37	21.25	19.35	16.11

VI. Conclusion

In this study, a novel algorithm, MHBA, was proposed for the multi-target task scheduling issue in a cloud computing context. Firstly, a chaotic Bernoulli initialization strategy was adopted to improve the diversity of the initial population and expand the search area. In this way, the algorithm's randomness is enhanced, and its convergence is improved. The second contribution of this study is the development of a piecewise optimum decreasing neighborhood model. In this way, the algorithm's search space is optimized, thereby improving its convergence to the solution. The efficiency of the MHBA algorithm has been validated using experimental results. It can be observed that, compared with the other baseline algorithms, MHBA outperforms them across various metrics at different workloads. The advantages of MHBA become more pronounced as the workload increases. MHBA is better suited to large-scale scheduling problems. In addition, the significance test results indicate that MHBA's advantages are statistically significant.

MHBA's better performance can be attributed to the synergy among its three enhancement strategies. The chaotic initialization mechanism enhances the algorithm's global search, the adaptive neighborhood strategy balances exploration and exploitation, and the horizontal crossover mechanism improves population diversity and information sharing. These mechanisms enable MHBA to have outstanding optimization ability. As for future research, several avenues are outlined. Firstly, it is possible to propose adaptive parameter control mechanisms that can improve the algorithm's robustness in dynamic environments. Secondly, it is possible to incorporate energy/cost constraints into the scheduling process, thereby improving the algorithm's applicability to green computing. Thirdly, it is possible to extend the MHBA framework to collaborative environments and real-time task scheduling, and to improve the algorithm's applicability to emerging/developing distributed computing environments, e.g., the Internet of Things and edge computing.

References

[1] Q. Li and S. Lang, "Dynamic resource mapping in cloud computing workflows using the modified artificial fish swarm algorithm with dynamic vision and step-size adaptation," *Computing,* vol. 108, no. 4, p. 59, 2026.

[2] 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.

[3] T. Hagras and G. A. El-Sayed, "A fault-tolerant and load-balancing scheduler for independent tasks on cloud-based virtual machines," *Cluster Computing,* vol. 29, no. 1, p. 61, 2026.

[4] A. Ayyadi and A. Jahani, "Addressing cost and resource variability for big data task scheduling in heterogeneous cloud environments: A. Ayyadi, A. Jahani," *The Journal of Supercomputing,* vol. 81, no. 6, p. 762, 2025.

[5] Q. Jiang, X. Xin, T. Zhang, and K. Chen, "Energy-Efficient Task Scheduling and Resource Allocation in Edge Heterogeneous Computing Systems Using Multi-Objective Optimization," *IEEE Internet of Things Journal,* 2025.

[6] K. Danach, K. Hamze, H. Harb, and H. Kanj, "Carbon - Aware Scheduling in Cloud Computing Operations: A Multi - Objective Optimisation Approach," *IET Smart Grid,* vol. 9, no. 1, p. e70056, 2026.

[7] F. Safi-Esfahani, H. Larian, S. Saeedi Mobarakeh, and S. Mirjalili, "Dynamic scheduling of independent tasks in cloud computing environment applying improved chicken swarm optimization and differential evolution," *The Journal of Supercomputing,* vol. 81, no. 8, p. 867, 2025.

[8] B. Hariharan, C. Anupama, R. K. Neerukonda, and R. K. Dhanaraj, "Energy - Aware System for Dynamic Workflow Scheduling in Cloud Data Centers: A Genetic Algorithm with DQN Approach," *Networked Sensing Systems,* pp. 121–143, 2025.

[9] B. Mondal, "Optimization techniques for big data: A VM load balancing in cloud computing using particle swarm optimization," in *Mathematical Modeling for Big Data Analytics*: Elsevier, 2026, pp. 143–156.

[10] S. Roy, R. Acharya, C. Chowdhury, S. Chattopadhyay, and P. Bhaumik, "An ant colony optimization based job allocation strategy for IoT-edge-cloud computing environment," *Journal of Ambient Intelligence and Humanized Computing,* vol. 17, no. 1, pp. 125–139, 2026.

[11] F. A. Hashim, E. H. Houssein, K. Hussain, M. S. Mabrouk, and W. Al-Atabany, "Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems," *Mathematics and Computers in Simulation,* vol. 192, pp. 84–110, 2022.

[12] P. Pirozmand, H. Jalalinejad, A. A. R. Hosseinabadi, S. Mirkamali, and Y. Li, "An improved particle swarm optimization algorithm for task scheduling in cloud computing," *Journal of Ambient Intelligence and Humanized Computing,* vol. 14, no. 4, pp. 4313–4327, 2023.

[13] R. Gong, D. Li, L. Hong, and N. Xie, "Task scheduling in cloud computing environment based on enhanced marine predator algorithm," *Cluster Computing,* vol. 27, no. 1, pp. 1109–1123, 2024.

[14] 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.

[15] M. Selselejoo and H. Ahmadifar, "DT-GWO: A Hybrid Decision Tree and GWO-Based Algorithm for Multi-Objective Task Scheduling Optimization in Cloud Computing," *Sustainable Computing: Informatics and Systems,* p. 101138, 2025.

[16] F. A. Emara, A. A. Gad-Elrab, A. Sobhi, A. S. Alsharkawy, M. E. Embabi, and M. Abd El-Baky, "Multi-objective task scheduling algorithm for load balancing in cloud computing based on improved Harris hawks optimization," *The Journal of Supercomputing,* vol. 81, no. 6, pp. 1–38, 2025.

[17] L. Du, T. Xie, and B. Chen, "Adaptive chaotic reverse learning-enhanced reptile search algorithm for efficient task scheduling in cloud computing," *The Journal of Supercomputing,* vol. 82, no. 2, p. 105, 2026.

[18] A. B. Kathole, A. S. Mirge, and S. G. Patil, "Optimizing cloud task scheduling and VM placement with multi-objective constraints using hybrid lemurs and gannet optimization algorithm," *Multimedia Tools and Applications,* vol. 85, no. 2, p. 107, 2026.

[19] S. Awad, M. Gamal, K. A. El Salam, and R. F. Abdel-Kader, "Efficient workflow scheduling in fog-cloud collaboration using a hybrid IPSO-GWO algorithm," *Scientific Reports,* vol. 16, no. 1, p. 4359, 2026.

[20] M. A. Tawfeek, I. Alrashdi, M. Alruwaili, B. Aldughayfiq, and H. Allahem, "An enhanced hybrid artificial bee colony and genetic algorithm for multi-objective workflow scheduling in the cloud," *Computing,* vol. 108, no. 2, p. 28, 2026.

[21] B. Rajeshwari and M. Namratha, "A Novel Push Pull Optimization Algorithm Fostered Deadline Aware Task Scheduling in Virtualized Cloud Computing," *Franklin Open,* p. 100499, 2026.