Optimizing Cloud Task Scheduling with Animal-Inspired Metaheuristic Algorithms

Komal Shah, Yiheng Niu, Xuehan Yi, Fangxiao Guo, Michal Aibin
Khoury College of Computer Sciences, *Northeastern University*, Vancouver, Canada
vancouver@northeastern.edu

***Abstract*—Cloud computing addresses the growing demand for computational power by providing on-demand resources. Efficient task scheduling is essential to ensure optimal resource utilization. In this paper, we explore metaheuristic algorithms inspired by animal behavior, including Cheetah Optimization (CO), Horse Herd Optimization (HOA), Grey Wolf Optimizer (GWO), and hybrid algorithms such as mGWO-WOA and mGWO-EBWOA. We compare their performance in cloud task scheduling based on makespan and execution costs. While the hybrid algorithms do not outperform the GWO baseline and CO offers a modest improvement, the HOA demonstrates significant gains, improving scheduling efficiency by up to 13.9% in high-task scenarios, making it the most effective approach in complex environments.**

***Index Terms*—cloud computing, task scheduling, heuristics**

I. Introduction

Infrastructure as a Service (IaaS) gives access to scalable and elastic computing devices for spreading wide-scale applications in the cloud computing domain. The IaaS model provides virtualized computing devices called virtual machines (VMs) with preconfigured CPU processors, storage, area, memory, and bandwidth to the end users for their use cases. The first benefit is that resources are used based on user needs, allowing them to pay per usage of the infrastructure. The second benefit is that IaaS cloud computing allows straightforward resource provisioning, enhancing user application production. The third benefit is that users can access rented resources anywhere and anytime based on the coveted level of service [1], [2]. However, it is still challenging to preconfigure adequate resources to perform large-scale tasks on IaaS. This problem is known as task scheduling optimization, which is the focus of this paper.

Task scheduling in cloud computing can be modeled as a bin-packing problem and is a non-deterministic polynomial-time hard (NP-hard) problem [3], [4]. Being an NP-Hard Problem, no known algorithms can optimally schedule tasks in a cloud environment for any given set of tasks to be scheduled on any given set of machines in polynomial time. Due to the recent increase in functional applications deployed on the cloud and difficulties associated with performing massive-scale requests on the fly, task scheduling of applications on the massive scale has fallen under the emerging investigation in the cloud computing environment [5]–[7]. The typical phenomenon of randomness that accompanies task execution time, the dynamic computing resource, and uncertain feedback regarding the action of an object in the scheduler causes task scheduling to be challenging.

Metaheuristic algorithms have been well-developed and adopted in the industry. Researchers have tried various algorithms, from early ant colony optimization and particle swarm optimization to the nearest cheetah optimizer [8] and grasshopper optimization [9], [10]. Research for cloud task scheduling problems has been done using metaheuristic algorithms such as Grey Wolf Optimization (GWO) [11] and Whale Optimization Algorithm (WOA) [12]. Based on the existing literature, approaches usually consider single-objective conditions, which concentrate primarily on makespan or cost. In real task scheduling problems, multiple parameters must be considered to achieve optimized performance.

In this paper, we plan to use makespan and job execution cost as our optimization objective. We compare the efficiency of algorithms such as Cheetah Optimization (CO) and Horse Herd Optimization (HOA) and create modified hybrid algorithms, such as 1) WOA and modified GWO (mGWO); 2) modified WOA (mWOA) and mGWO for this task. As a baseline, we will compare the results of our proposed algorithms against GWO, as plenty of research has been done on using GWO for task scheduling. Also, the GWO algorithm is closely related to two of our hybrid algorithms.

The rest of the paper is organized as follows. Section II introduces a survey of various task scheduling approaches. Section III presents the proposed algorithms in detail, followed by sections related to simulation setup, results and discussion.

II. Problem Statement

In any cloud computing domain, the efficiency of the task scheduling algorithm plays a significant role in determining the resource utilization of the cloud system [13]. Hence, devising efficient algorithms for this problem is of utmost importance. Task Scheduling Problem can be defined as what task execution policy can be used for scheduling jobs in a cloud environment to generate optimal utilization of resources. The optimal utilization of resources can be measured using various criteria such as reducing overall makespan or costs.

A logical view of any typical task scheduling process is shown in Figure 1. It comprises clients who will submit tasks or jobs for execution, a task scheduler and a series of heterogeneous VMs. Once tasks are submitted for execution, the task scheduler is responsible for submitting them on VMs. Task scheduler generates a policy for executing these jobs

979-8-3315-2096-0/25/$31.00 ©2025 IEEE

based on heuristic information such as resources available on various and meta-data of the submitted tasks. Since we want to compare the performance of various algorithms for the task scheduling problem, to simplify the problem, we assume that the tasks are independent of each other.

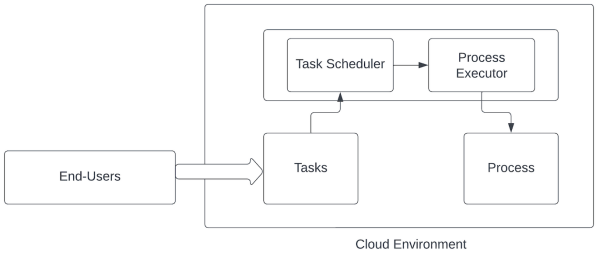

Fig. 1: Logical Overview of Task Scheduling

A. Problem Description

Task scheduling is an essential process for matching customer requests with available resources in a data center. The Task Manager is critical in executing specific algorithms to optimize this process. Imagine a set of tasks is defined as $T = \{T_1, T_2, ..., T_i, ..., T_N\}$, where $i \in [1, N]$ and N is the total number of tasks to be scheduled. Every task T_i can be described as $T_i(sz_i, ddl_i, st_i)$, where sz_i, ddl_i, and st_i denote the size, the deadline and the start time of the task. The set of tasks is presented to the Task Manager at different times.

Each processing element, such as a virtual machine (VM), has a unique processing rate and varying memory. A set of VMs is defined as $VM = \{vm_1, vm_2, ..., vm_j, ..., vm_M\}$, where $j \in [1, M]$ and M is the total number of virtual machines. This means performing tasks on different VMs can result in different execution times and costs. Every VM can be described as $vm_i(cv_j)$, where cv_j denotes the processing capacity of the VM.

Based on the detailed information of the input tasks and the available computing resources, tasks and resources will be mapped by a certain strategy. The scheduler assigns tasks $\{T_1, T_2, ..., T_i, ..., T_N\}$ to available VMs $\{vm_1, vm_2, ..., vm_j, ..., vm_M\}$ to minimize the overall makespan and cost. The final scheduling result can be represented by a matrix A as follows:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & ... & a_{1M} \\ a_{21} & a_{22} & a_{23} & ... & a_{2M} \\ ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... \\ a_{N1} & a_{N2} & ... & ... & a_{NM} \end{bmatrix} \quad (1)$$

where a_{ij} is a decision variable, in which $a_{ij} = 1$ means that the ith task is performed on the jth VM, otherwise $a_{ij} = 0$, and $\sum_{j=1}^{M} a_{ij} = 1$ for each $i \in [1, N]$. The notations used are listed in Table I.

To achieve multi-objective optimization and illustrate the overall processing capacity and resource utilization of a cloud computing system in a task scheduling scenario, we use two attributes to represent each resource node. These two attributes are processing capability and load capability, represented by the CPU computing power of a node and the number of data centers occupied by the cloud service, respectively. Each computing task can be characterized using these attributes: quantifying computing capability with the time required for the tasks to be completed on the VMs and quantifying cost with the occupation of data center numbers. Based on this foundation, we can model the underlying computing system as two vectors, namely the completion time vector CT and the total cost TC. In the following, we adopt the notations provided in Table I for clarity and consistency.

Notation	Definition
i	Index for tasks
N	Total number of tasks
j	Index for VMs
M	Total number of VMs
sz_i	Size of T_i
ddl_i	Deadline of T_i
st_i	Start time of T_i
CT_{ij}	Completion time of vm_j and T_i
ET_{ij}	Execution time of vm_j and T_i
WT_{ij}	Wait time of vm_j and T_i
cv_j	Processing capacity of vm_j
pc_j	Number of processors in vm_j
$MIPS_j$	Million Instructions per second of one processor in vm_j
CT	Total completion time
TC	Total cost
CPS_i	Cost per second of T_i

TABLE I. Notations

B. Models and Metrics

1) Makespan: CT_{ij} represents the completion time of executing task T_i on vm_j, for task T_i, assigned to virtual machine vm_j. If the completion time CT_{ij} is within the deadline of T_i, which is ddl_i, then this task can be assigned to the VM. The completion time CT_{ij} consists of waiting and execution times. Waiting time (WT_{ij}) is the difference between the start time (st_i) and the execution time (ET_{ij}). The completion time can be calculated using Eq.2:

$$CT_{ij} = WT_{ij} + ET_{ij} \quad (2)$$

The execution time is estimated using the task size and the processing capacity of VMs. It can be calculated using Eq. 3:

$$ET_{ij} = \frac{sz_i}{cv_j} \quad (3)$$

where sz_i is the number of instructions that task T_i needs to be executed on vm_j, and cv_j is the processing capacity which can be calculated using Eq. 4:

$$cv_j = pc_j * MIPS_j \quad (4)$$

where pc_j is the number of processors in vm_j, and $MIPS_j$ is the Million Instructions per Second of one processor in vm_j. Thus, makespan can be expressed using Eq. 5:

$$Makespan = max\{CT_{ij}, \forall tasks\ t_i\ assigned\ to\ vm_j\} \quad (5)$$

The objective function is represented by Eq. 6:

$$F_M^{min} = max\{CT_{ij}\} \tag{6}$$

2) Cost: Another objective of our proposed algorithm is to reduce the execution cost. It is the cost of each task on a specific VM. To calculate the cost for each task, we need to multiply the cost per second and the actual execution time, as Eq. 7:

$$Cost_i = CPS_i * ET_{ij} \tag{7}$$

In the equation above, $Cost_i$ denotes the execution cost of T_i, CPS_i denotes the cost per second of T_i, ET_{ij} denotes the execution time of T_i on vm_j.

The objective is to reduce the total cost of all tasks so that the objective function can be represented in Eq. 8:

$$F_C^{min} = min\left\{\sum_{i=1}^{N} Cost_i\right\} \tag{8}$$

C. Fitness Function

The primary metrics of our approach are the makespan and cost of task scheduling in cloud computing. Our goal is to minimize these two objectives simultaneously. It is accomplished by formulating a fitness function, which enables us to evaluate the quality of the obtained solutions. However, it should also be noted that, like many optimization problems, the fitness function we have developed can only provide us with an approximate optimal solution. The fitness function we use for evaluating the time and cost parameters is defined in Eq.9

$$F_{opt} = w_1 F_M^{min} + w_2 F_C^{min} \tag{9}$$

where F_{opt} denotes the fitness function, F_M^{min} is the objective to minimize makespan, and F_C^{min} is the objective to minimize the cost. w_1 and w_2 are the weight of two objectives, whose value is between 0 and 1 and add up to 1. With a larger weight, there is a higher priority to minimize the objective.

III. Algorithms

Let us now discuss how nature-inspired algorithms (GWO, WOA, CO and HOA) are adapted for the task scheduling problem, where tasks need to be efficiently allocated to virtual machines (VMs) in a cloud environment.

A. Baseline Comparison Algorithm - Grey Wolf Optimizer (GWO)

As our baseline solution, we chose the Grey Wolf Optimizer (GWO). In task scheduling, grey wolves represent different VM configurations, and the prey represents the optimal allocation of tasks to VMs. The alpha (α), beta (β), and delta (δ) wolves represent the top candidate solutions for task scheduling, with alpha being the most optimal scheduling solution at any given iteration. We execute the following processes for the GWO algorithm:

- **Encircling Prey**: The VMs encircle the optimal task assignment by gradually adjusting their allocation strategies. VMs with higher computational power or better resource configurations are moved closer to tasks with higher demands, ensuring that the most critical tasks are efficiently handled.
- **Hunting**: The top wolves (VMs) are assigned the most demanding tasks based on their configurations. This step helps improve the makespan and resource utilization. Beta and delta wolves assist in refining the allocations by adjusting VM-task assignments based on current performance.
- **Attacking Prey**: As the wolves (VMs) get closer to the optimal allocation, the algorithm reduces the search space, focusing on refining the task scheduling solution until the best allocation (prey) is captured.

B. Hybrid Approach - Modified Grey Wolf and Whale Optimizer Algorithm (mGWO-WOA)

In the mGWO-WOA hybrid, the Grey Wolf Optimizer maps tasks to VMs, performing the initial exploration of task scheduling solutions. The Whale Optimization Algorithm refines these solutions, ensuring task allocation is resource-efficient and cost-effective.

- **GWO Phase**: During the initial GWO phase, VMs are represented by the wolves, and tasks are mapped to them based on the alpha (α), beta (β), and delta (δ) wolves (i.e., the best-performing VM-task configurations). This phase explores various VM configurations to handle different task requirements as in standard GWO.
- **WOA Phase**: After GWO's initial exploration, the Whale Optimization Algorithm refines the solution by performing deeper searches around the best configurations found. Here, VMs make smaller adjustments to task allocation, ensuring that tasks are scheduled optimally in terms of time and resource usage.

Since the GWO phase is similar to the one described above, let us provide more details for the WOA Phase:

- **Encircling Prey**: VMs try to "encircle" the optimal task assignment by adjusting their configurations to meet the computational demands of the tasks. The leader VM (the whale closest to the optimal solution) directs the others to improve the scheduling based on current results.
- **Bubble-Net Attack**: This corresponds to adjusting the task allocation across VMs, either by shrinking the search space to refine the allocation or by using spiral movements (exploring new configurations) to balance the overall load across VMs. The VMs are optimized based on how well they handle assigned tasks, in terms of both makespan and cost.
- **Search for Prey**: When a VM configuration is far from optimal, the algorithm performs a global search, exploring alternative VM configurations to improve task allocation efficiency. This phase ensures that no better VM configuration is missed during the scheduling.

This combined approach allows for wide exploration (to find robust initial VM-task mappings) and focused refinement (to improve the efficiency of the best mappings).

C. Hybrid Approach - Grey Wolf and Elite-Based Whale Optimizer Algorithm (mGWO-EBWOA)

In this approach, mGWO generates an initial solution, and EBWOA further refines it by using elite-based selection strategies to improve the quality of task scheduling.

- **GWO Phase**: Similar to the previous hybrid, GWO maps tasks to VMs. The wolves' roles help explore various VM configurations, identifying initial matches between tasks and VMs.
- **Elite-Based Selection in EBWOA**: The top-performing VMs (elite whales) are selected to handle the most critical tasks. Elite-based selection ensures that the best configurations from the GWO phase are retained and improved, avoiding poor configurations while refining the overall task allocation. This step guarantees that tasks are optimally distributed to VMs, leveraging the best-performing VM setups found during the GWO phase. The elite-based whale strategy ensures that the top VM task allocations continue to improve, avoiding local optima and enhancing the final scheduling result.

D. Cheetah Optimization Algorithm (CO)

The Cheetah Optimization Algorithm (CO) uses the metaphor of fast and precise cheetah hunts to represent rapid task scheduling to VMs. The cheetahs (VMs) must make quick decisions about which tasks to allocate, ensuring minimal delay in execution (makespan).

- **Cheetah Speed (Task Execution)**: VMs are represented as cheetahs, with their speed reflecting the quick decision-making process in task allocation. Faster VMs (those with higher processing power) are matched with more demanding tasks, ensuring tasks are completed efficiently and quickly.
- **Prey (Optimal Task Scheduling)**: The optimal solution (prey) represents the best possible mapping of tasks to VMs. The cheetahs' actions focus on narrowing down the most suitable VM configurations to handle the tasks, ensuring that processing times and resource usage are minimized.

CO ensures that task allocation is fast and efficient, making quick adjustments to match each task's computational needs to the best available VMs.

E. Horse Herd Optimization Algorithm (HOA)

The Horse Herd Optimization Algorithm (HOA) simulates the behaviors of a herd of horses with different ages, each group representing different VM configurations for task scheduling. Older horses represent more refined (exploitative) VM configurations, while younger horses explore newer configurations.

- **Herd Behavior (VM Collaboration)**: The herd represents a pool of VMs. Older, more experienced horses (slower VMs) refine the task allocation by improving their configurations, while younger horses (more powerful VM setups) explore different ways of assigning tasks. This collaboration ensures that the algorithm balances exploration and exploitation in the task scheduling.
- **Age Groups (VM Roles)**: Different age groups of horses represent varying VM configurations. Older VMs focus on optimizing specific tasks that are more repetitive and standardized, while younger VMs focus on the ones that require more computational power.
- **Roaming and Grazing (Task Assignment)**: Roaming represents the exploration of different task allocation strategies by younger VMs. Grazing represents the fine-tuning of task assignments by older VMs. This process helps ensure that all tasks are handled efficiently by the most appropriate VMs.

IV. Methodology

Our goal is to compare the fitness of the schedule generated by each of our proposed algorithms against the baseline algorithm. To simulate the task scheduling scenarios, we test scenarios with various numbers of tasks and VMs. The following groups of tests have been considered.

- Varying number of VMs: we keep the number of tasks constant, and we change the number of VMs starting with 10 VMs and going up to 50 VMs.
- Varying number of tasks: we keep the number of VMs constant, and we change the number of tasks starting with 1000, and going up to 10000.

In each one of the testing groups, the targeting parameter changes, whereas other parameters remain constant. The parameter numbers are chosen referring to related task scheduling studies using different meta-heuristic algorithms, along with including the real-world scenarios in consideration. To make the algorithms comparable and control the fuzziness, we use the same random seed with each of our algorithms.

We tuned the two hyper-parameters, iteration and population, for all our meta-heuristic algorithms using a fixed number of 30 VMs and 1000 tasks in the simulation. To find the optimal number of iterations, we set the population size to 30 and ran the algorithm for 15,000 iterations, tracking the fitness value.

To determine the best population size, we fixed the number of iterations to 1000 and varied the population size. Since no asymptotic behavior was observed, we finalized the iteration count at 10,000 and the population size at 24, as they yielded the best fitness scores. These values were used consistently across all algorithms for comparison.

With the hyper-parameters fine-tuned, we start our simulation by testing various numbers of VMs and tasks. Tables II, III and IV shows the parameters used for this work.

TABLE II. VM Configuration

VM Resource	Assigned Value
RAM	512 MB
Storage	10000
Bandwidth	1000
Number of CPU	1
MIPS	500 MIPS

TABLE III. Task Configuration

Task resource	Required value
File storage	300 MB
Output size	300 MB
Job length	100 - 1000 Millions of Instructions (MI)

TABLE IV. Simulation Configuration

Number of VMs	10, 20, 30, 40, 50
Number of tasks	1000, 2000, 3000, ..., 10000
Max rounds of optimization iterations	10000
Population size	24

V. Results

The results presented in Tables VI, VII, VIII and IX show a detailed comparison of the algorithms in different scenarios. The baseline GWO values are listed in Table V for reference.

TABLE V. GWO Baseline Fitness Values

Tasks	10 VMs	20 VMs	30 VMs	40 VMs	50 VMs
1000	240.57	135.65	91.72	71.77	57.51
2000	801.99	426.32	293.44	225.97	183.28
3000	1,672.91	911.15	608.90	467.14	373.74
4000	2,853.31	1,519.43	1,022.34	785.05	630.01
5000	4,375.25	2,344.62	1,582.21	1,198.34	969.14
6000	6,245.45	3,315.88	2,236.20	1,707.55	1,367.25
7000	8,422.21	4,463.15	2,990.00	2,280.55	1,836.05
8000	10,906.65	5,781.79	3,872.02	2,955.85	2,368.68
9000	13,774.90	7,339.16	4,889.58	3,712.58	2,988.36
10000	16,868.99	8,989.25	5,996.30	4,556.98	3,661.66

TABLE VI. mGWO-WOA Fitness Values

Tasks	10 VMs	20 VMs	30 VMs	40 VMs	50 VMs
1000	259.33	145.49	93.93	73.39	58.76
2000	820.42	446.07	306.68	230.62	186.43
3000	1,702.42	920.76	616.44	472.24	376.70
4000	2,904.17	1,558.51	1,034.59	790.72	635.52
5000	4,435.27	2,376.85	1,598.47	1,211.71	974.41
6000	6,313.28	3,364.25	2,251.29	1,703.50	1,373.26
7000	8,442.82	4,505.12	3,011.03	2,294.75	1,833.91
8000	10,984.38	5,847.90	3,904.75	2,963.94	2,383.33
9000	13,860.59	7,380.22	4,916.66	3,739.31	3,004.66
10000	16,989.83	9,035.44	6,047.49	4,605.45	3,668.92

TABLE VII. mGWO-EBWOA Fitness Values

Tasks	10 VMs	20 VMs	30 VMs	40 VMs	50 VMs
1000	242.72	137.60	96.70	72.00	60.54
2000	795.73	432.09	298.08	229.09	188.25
3000	1685.18	917.71	612.43	474.42	385.43
4000	2846.48	1540.41	1031.75	791.61	645.98
5000	4368.02	2382.45	1574.36	1204.61	971.85
6000	6239.74	3356.79	2224.05	1708.32	1391.81
7000	8426.73	4481.71	2994.15	2282.96	1850.77
8000	10867.87	5806.61	3862.43	2951.11	2386.87
9000	13729.71	7373.48	4883.86	3734.71	3011.95
10000	16864.33	8977.00	6018.13	4552.86	3675.95

The hybrid algorithms (mGWO-WOA and mGWO-EBWOA) consistently underperform compared to the baseline GWO. This is likely due to the increased complexity introduced by combining two optimization strategies. Instead of enhancing performance, the hybrids might struggle with balancing exploration and exploitation, leading to inefficient task allocation. The hybrids may also suffer from longer convergence times, where the integration of multiple methods prevents them from efficiently reaching optimal solutions. This combination can cause redundant or conflicting search patterns, diminishing overall performance.

TABLE VIII. CO Fitness Values

Tasks	10 VMs	20 VMs	30 VMs	40 VMs	50 VMs
1000	240.61	132.41	89.48	71.59	57.03
2000	788.42	424.33	288.84	221.03	181.93
3000	1,653.96	891.62	594.46	459.40	370.54
4000	2,836.42	1,526.14	1,008.49	781.03	631.24
5000	4,360.11	2,339.84	1,547.76	1,197.81	948.97
6000	6,205.04	3,276.43	2,208.08	1,682.21	1,355.76
7000	8,356.06	4,448.62	2,950.24	2,267.54	1,805.93
8000	10,811.30	5,742.90	3,852.65	2,928.11	2,368.99
9000	13,729.21	7,284.11	4,857.01	3,680.62	3,005.86
10000	16,825.34	8,893.65	5,909.84	4,564.20	3,658.92

TABLE IX. HOA Fitness Values

Tasks	10 VMs	20 VMs	30 VMs	40 VMs	50 VMs
1000	236.48	133.62	90.53	70.35	56.80
2000	775.31	412.83	289.37	222.68	180.05
3000	1,640.43	891.15	600.81	462.26	369.94
4000	2,654.31	1,508.80	1,015.60	778.32	625.22
5000	4,013.56	2,304.30	1,565.76	1,185.00	960.38
6000	5,625.73	3,288.28	2,207.24	1,687.27	1,364.71
7000	7,543.37	4,388.57	2,951.13	2,252.08	1,824.46
8000	9,640.57	5,669.93	3,854.77	2,918.34	2,362.83
9000	12,048.80	7,225.44	4,872.95	3,675.79	2,980.48
10000	14,524.21	8,836.37	5,918.37	4,483.26	3,632.39

The CO algorithm shows a modest but steady improvement over GWO. On average, CO provides around a 1% improvement in fitness, particularly noticeable as task counts increase. This slight advantage can be attributed to CO's better capability in balancing exploration and exploitation within the search space. CO's underlying mechanisms might enable it to escape local optima more effectively than GWO, especially in higher task scenarios, where more complex scheduling decisions are required. However, its improvements remain modest due to similarities in structure with GWO, which limits the potential for substantial gains.

The most significant performance boost comes from the HOA algorithm, which consistently outperforms GWO. The improvement increases with the number of tasks, starting at 1.7% for 1000 tasks and growing to 13.9% for 10000 tasks in the 10 VMs configuration. This behavior suggests that HOA excels in more complex and constrained environments where larger task loads challenge the system's scheduling efficiency. One potential reason for this could be linked to how the algorithm is modeled after horse herding behavior. In smaller and more manageable environments, horses (or tasks, in this case) can efficiently organize under a few alpha leaders, making quick decisions and optimizing resource usage.

However, as the number of VMs increases, the improvement offered by HOA decreases slightly. For example, the improvement for 1000 tasks drops from 1.7% for 10 VMs to 0.9% for 50 VMs. This could be likened to larger herds where more dominant or younger horses (representing more

VMs) lead to a more complicated decision-making process. When too many "leaders" or resources are available, the efficient coordination that HOA relies on becomes harder to achieve, and the advantages of the algorithm are diminished. In such cases, simpler algorithms like GWO, which don't rely on complex coordination mechanisms, can perform more competitively because the problem is less constrained.

In summary, HOA delivers the most substantial performance gains, particularly in environments with high task counts and fewer VMs, where the scheduling problem is more complex. CO also offers consistent, albeit smaller, improvements, likely due to its enhanced ability to balance exploration and exploitation in task scheduling. The hybrid algorithms, however, fail to surpass the baseline, likely due to the increased complexity and inefficiencies introduced by combining two strategies without effective synergy.

VI. Conclusion

In this study, we explored various meta-heuristic algorithms based on animal behavior to optimize task scheduling in a Cloud Environment. The proposed algorithms, inspired by intelligent swarm behavior, were tested for scheduling jobs across multiple VMs. We compared the performance of our algorithms with the baseline GWO algorithm. While the hybrid algorithms did not surpass GWO, CO showed a modest improvement, and HOA significantly increased efficiency, improving performance by nearly 14% in constrained scenarios.

Despite the promising results, there are limitations to our study. First, we evaluated the algorithms in a static environment, which may not reflect the dynamic nature of real-world cloud systems. Future work should consider testing the adaptability of these algorithms in dynamic environments. Additionally, our comparisons were limited to GWO; comparing our algorithms with other advanced job scheduling techniques would provide a more thorough evaluation. Lastly, while this study focused on task scheduling, future research could extend to other cloud optimization areas, such as resource allocation and load balancing based on short term traffic predictions [14], [15], to further improve system performance.

References

[1] Abdullahi, Mohammed, et al. "An efficient symbiotic organisms search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment." Journal of Network and Computer Applications 133 (2019): 60-74.

[2] Mohammadi, Alireza, and Mohammad Hossein Rezvani. "A novel optimized approach for resource reservation in cloud computing using producer–consumer theory of microeconomics." The Journal of Supercomputing 75.11 (2019): 7391-7425.

[3] Bobroff, Norman, Andrzej Kochut, and Kirk Beaty. "Dynamic placement of virtual machines for managing SLA violations." 2007 10th IFIP/IEEE International Symposium on Integrated Network Management. IEEE, 2007.

[4] Genez, Thiago AL, Luiz F. Bittencourt, and Edmundo RM Madeira. "Workflow scheduling for SaaS/PaaS cloud providers considering two SLA levels." 2012 IEEE Network Operations and Management Symposium. IEEE, 2012.

[5] M. Aibin and K. Walkowiak, "Resource requirements in fixed-grid and flex-grid networks for dynamic provisioning of data center traffic," 2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), Vancouver, BC, Canada, 2016, pp. 1-4, doi: 10.1109/CCECE.2016.7726716.

[6] M. Aibin, "LSTM for Cloud Data Centers Resource Allocation in Software-Defined Optical Networks," 2020 11th IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), New York, NY, USA, 2020, pp. 0162-0167

[7] N. K. K. Kit and M. Aibin, "Study on High Availability and Fault Tolerance," 2023 International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 2023, pp. 77-82.

[8] Mohammad Amin Akbari, Mohsen Zare, Rasoul Azizipanah-abarghooee, Seyedali Mirjalili & Mohamed Deriche, "The cheetah optimizer: a nature-inspired meta-heuristic algorithm for large-scale optimization problems"

[9] J. Vahidi and M. Rahmati, "Optimization of resource allocation in cloud computing by grasshopper optimization algorithm," in Proc. 5th Conf. Knowl. Based Eng. Innov. (KBEI), Feb. 2019, pp. 839–844.

[10] K. Naveen Durai, R. Subha and A. Haldorai, "Hybrid invasive weed improved grasshopper optimization algorithm for cloud load balancing,"Intelligent Automation and Soft Computing, vol. 34, no.1, pp. 467–483, 2022.

[11] G. Natesan, and Chokkalingam, A. (2018). "Task scheduling in heterogeneous cloud environment using mean grey wolf optimization algorithm", ICT Express, 1-5.

[12] S. Mirjalili, A. Lewis, "Advances in Engineering Software", Advances in Engineering Software 95 (2016) 51–67.

[13] Xuan Chen, Long Cheng, Cong Liu, Qingzhi Liu , Jinwei Liu, Ying Mao and John Murphy. "A WOA-Based Optimization Approach for Task Scheduling in Cloud Computing Systems." IEEE SYSTEMS JOURNAL, VOL. 14, NO. 3, September 2020.

[14] M. Aibin, N. Chung, T. Gordon, L. Lyford and C. Vinchoff, "On Short- and Long-Term Traffic Prediction in Optical Networks Using Machine Learning," 2021 International Conference on Optical Network Design and Modeling (ONDM), Gothenburg, Sweden, 2021, pp. 1-6.

[15] M. Aibin, K. Walkowiak, S. Haeri and L. Trajković, "Traffic Prediction for Inter-Data Center Cross-Stratum Optimization Problems," 2018 International Conference on Computing, Networking and Communications (ICNC), Maui, HI, USA, 2018, pp. 393-398.