**Enhanced Golden Jackal Optimization algorithm-based task scheduling in cloud for big data application**

**Abstract**

One of the most challenging aspects of cloud computing is effectively scheduling tasks across virtual machines to optimize resource utilization and minimize wait times. This challenge is compounded by the limitations of traditional metaheuristic algorithms, which often suffer from early convergence, resulting in a suboptimal balance between local and global search strategies. Additionally, many recent approaches fail to address essential cloud computing characteristics, such as heterogeneity, adaptability, dynamism, and elasticity, which are crucial for meeting user requirements and ensuring system reliability. To address these issues, this research proposes a multi-objective task scheduling strategy specifically designed for big data applications in cloud computing. Central to this strategy is the Enhanced Golden Jackal Optimization (EGJO) method, which integrates Opposition-Based Learning (OBL) to maintain population diversity and improve convergence toward optimal solutions. The multi-objective function of the proposed strategy considers critical factors such as cost, makespan, energy consumption, and resource utilization. The approach includes efficient solution representation, fitness function derivation, and the effective application of EGJO and OBL operators. The strategy's effectiveness is rigorously evaluated using various performance metrics and benchmarked against existing methods, demonstrating significant improvements in task scheduling outcomes.

**Keywords: -**Enhanced Golden Jackal Optimization, , task scheduling, big data and multi-objective.

**1. Introduction**

Cloud computing has swiftly become a foundational technology, revolutionizing data and resource management across various networks, including personal devices, data centers, and cloud-based services [1]. It represents a sophisticated form of parallel and distributed computing, where virtual machines (VMs) deliver computing power under service-level agreements (SLAs) that define the contract between providers and users. This flexibility, combined with the capacity to offer customizable software services, positions cloud computing as a pivotal infrastructure in the information technology landscape [2].

Cloud computing's capability to handle large data volumes has rendered it essential for various applications, including data analysis, storage, and IoT. Its inherent advantages—time sharing, virtualization, and adaptability—make cloud computing an ideal solution for organizations striving to keep pace with big data evolution trends [3,4]. However, the exponential growth in data generated by an increasing number of cloud users and IoT devices poses significant challenges. Managing this data effectively within cloud systems is becoming increasingly complex, as transferring all data to cloud data centers can lead to excessive bandwidth usage, higher costs, and latency issues.

Despite the widespread adoption and benefits of cloud computing, it is not without its challenges [5,6]. Among the most critical are task scheduling and resource allocation. Task scheduling, which involves assigning jobs to central data sources, remains a complex problem, particularly in dynamic and open cloud environments [7]. Rule-based scheduling algorithms, while popular due to their simplicity, often fall short in addressing the complexities of modern cloud systems [8-10]. These algorithms primarily focus on minimizing makespan by reducing task wait times and optimizing task-to-VM mapping to shorten transfer times. However, this approach does not necessarily guarantee enhanced performance across the board.

To tackle these challenges, this paper introduces an Enhanced Golden Jackal Optimization (EGJO) algorithm aimed at improving task scheduling in cloud environments. The algorithm focuses on optimizing multiple objectives, such as makespan, cost, energy consumption, and resource utilization, providing a more efficient and resilient solution to task scheduling in cloud computing.

**2. Literature survey**

Gaith Rjoub et al. [11] proposed four task scheduling methods for cloud computing, leveraging deep and reinforcement learning to reduce resource usage and task waiting times. They applied three techniques: Reinforcement Learning (RL), Deep Q Networks, Recurrent Neural Network Long Short-Term Memory (RNN-LSTM), and a combined Deep Reinforcement Learning with LSTM (DRL-LSTM) approach. Real-world data from Google Cloud Platform indicated that DRL-LSTM surpassed the others, cutting CPU usage costs by up to 67% compared to Shortest Job First (SJF) and by up to 35% relative to Round Robin (RR) and enhanced Particle Swarm Optimization (PSO) methods.

Mohammed Zaki Hasan and Hussain Al-Rizzo[12] developed a task scheduling algorithm using Canonical Particle Swarm Optimization (CPSO) aimed at enhancing resource allocation in both homogeneous and heterogeneous IoT cloud environments. Their approach models tasks with stochastic processing times as particle swarms, optimizing task allocation and execution order to Virtual Machines (VMs) based on a defined objective function sensitive to random variables. The CPSO method demonstrated superior throughput and VM utilization compared to Longest Processing Time (LPT) and Shortest Processing Time (SPT) algorithms.

Dibyendu Mukherjee et al. [13] developed an Adaptive Scheduling Algorithm for Task Loading (ASA-TL) to manage access requests effectively and prevent overloading in cloud data centres. This algorithm dynamically adjusts task scheduling based on demand, improving resource allocation and system efficiency. This system effectively balances load using virtual servers, considering the status and importance of digital devices for fair task assignment. The algorithm ensures that task inputs are consistently distributed among multiple processors. Experimental results indicate that ASA-TL outperformed in response time, data centre processing time, and overall costs.

Hicham Ben Alla et al. [14] proposed a task scheduling optimization approach for cloud computing, utilizing dynamic dispatch queues and hybrid metaheuristic algorithms to minimize waiting times, allocate tasks to appropriate resources, and enhance cloud performance. They employed two hybrid algorithms: Fuzzy Logic with Particle Swarm Optimization (FLPSO) and Simulated Annealing with Particle Swarm Optimization (SAPSO) for Task Scheduling with Dynamic Dispatch Queues (TSDQ). Their experiments demonstrated the effectiveness of the technique for both synthetic and real-world tasks.

Xingwang Huang et al. [15] developed a PSO-based scheduler aimed at optimizing task scheduling in cloud environments. The scheduler employs five distinct PSO variants—linear, sigmoid decreasing, chaotic, simulated annealing, and logarithm decreasing—that adjust the inertia weight dynamically. By incorporating these variants, the system optimizes task allocation to Virtual Machines (VMs) to reduce makespan. Among these, the logarithm decreasing inertia weight strategy is particularly effective in minimizing completion time. The scheduler also combines this approach with three established heuristic algorithms to address scheduling challenges more robustly.

Prasanta Kumar Bal et al. [16] developed a hybrid machine learning technique (RATS-HM) aimed at optimizing resource allocation and task scheduling in cloud computing environments. To tackle scheduling challenges, they applied an advanced Cat Optimization Algorithm, which effectively schedules user tasks while managing resource constraints. They further introduced ICS-TS to optimize passive resources by dividing the cloud environment into workspace and state space. Additionally, the GO-DNN method was integrated to minimize resource usage, supporting efficient management of numerous user requests across multiple servers in large-scale cloud settings.

Souvik Pal et al. [17] developed a Deep Learning Algorithm for Big Data Task Scheduling System (DLA-BDTSS) aimed at IoT and cloud computing. This multi-objective approach minimizes makespan and maximizes resource utilization. The authors introduced a regional exploration search technique to improve the optimization algorithm's data exploitation and local optimum avoidance. Comparisons using CloudSim tools showed that DLA-BDTSS outperformed established task allocation methods.

Ding Ding et al. [18] proposed a Q-learning-based task scheduling framework for energy-efficient cloud computing (QEEC), which operates in two phases. In the first phase, a centralized dispatcher directs incoming user tasks to the appropriate server request queues. In the second phase, each server's Q-learning scheduler determines task processing timing and employs an updating policy to allocate tasks to Virtual Machines (VMs), using incentives to reduce response time and enhance CPU utilization.

**3. Problem definition with contribution**

Task scheduling in cloud computing presents significant challenges, particularly due to the need to efficiently allocate multiple processes across virtual machines (VMs). This NP-hard problem is especially pronounced in big data applications, where inadequate scheduling can lead to reduced processing performance, longer wait times, extended makespan, and increased costs. Finding an optimal method for distributing numerous jobs among available VMs is complex. While various metaheuristic algorithms have been employed to tackle this issue, they often face premature convergence, leading to an imbalance between local and global search strategies. Additionally, many contemporary approaches neglect critical cloud computing characteristics such as heterogeneity, flexibility, dynamism, and elasticity, which are vital for meeting user requirements and ensuring system reliability.

Therefore, there is a pressing need for an optimization method that effectively addresses big data scheduling challenges in cloud environments. This paper proposes a multi-objective task scheduling approach for big data applications in cloud computing, utilizing an Enhanced Golden Jackal Optimization (EGJO) algorithm. The main contributions of this approach are as follows:

* The Enhanced Golden Jackal Optimization (EGJO) algorithm is utilized for multi-objective task scheduling in cloud computing for big data applications.
* An Opposition-Based Learning (OBL) strategy is integrated with EGJO to maintain population diversity and improve convergence to optimal solutions.
* The multi-objective function focuses on makespan, cost, energy consumption, and resource utilization.
* The effectiveness of the approach is evaluated using metrics such as makespan, resource utilization, and cost..

**5. Proposed task scheduling on cloud environment**

The primary objective of scheduling algorithms is to enhance resource utilization while minimizing makespan and cost without compromising cloud service parameters. To achieve this, the Enhanced Golden Jackal Optimization (EGJO) algorithm is proposed. The scheduling approach, illustrated in Figure 1, involves three key processes: **Information Providing Process:** The task scheduler gathers data on tasks and resources from the task and resource managers. **Selection Process:** The scheduler selects the target resource based on parameters such as reliability, task size, task priority, dynamic slotted length, and activity-based cost. The allocation plan is then sent to the resource manager. **Task Distribution Process:** The task manager assigns each task to the appropriate resources.

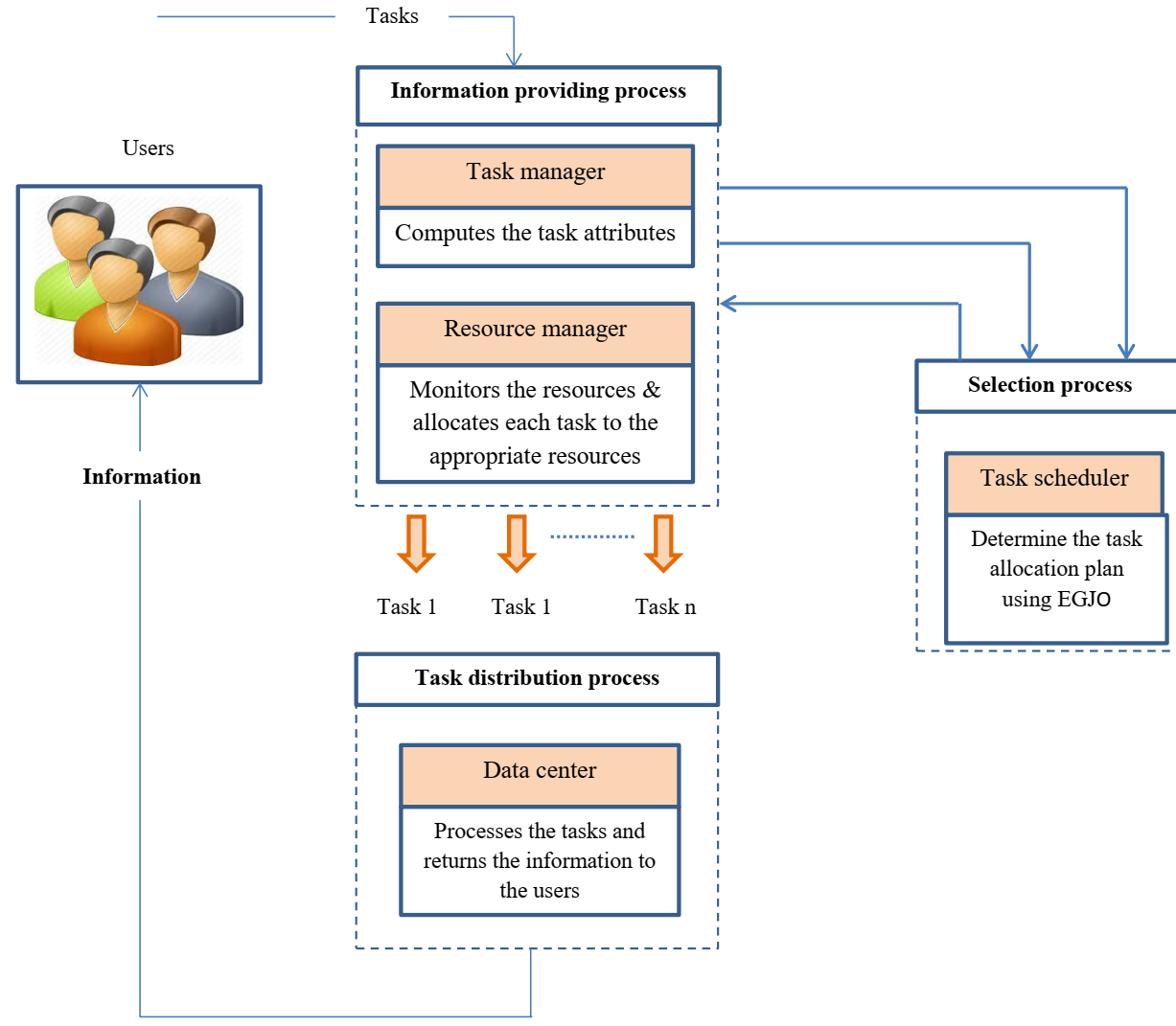


Figure 1: Structure of proposed scheduling methodology

**Phase 1: Information providing process**

Tasks are queued and grouped into batches due to their large volume. The proposed algorithm divides tasks into blocks to optimize resource utilization and improve load balancing. An adaptive splitting method is employed to group tasks into batches, with the batch length determined by equation (1).

 (1)

To address the requirement that the batch length must always be less than or equal to the average available resources, you can formulate this constraint mathematically. Here’s a refined formulation that incorporates the concepts you've mentioned:

 (2)

 (3)

To achieve optimal task mapping, our focus is on minimizing completion times while maximizing resource utilization. In the next phase, we will develop an allocation plan based on the current state of resource utilization.

**Phase 2: Selection process**

In the scheduling phase, we allocate task batches to Virtual Machines (VMs) using the EGJO algorithm, inspired by the hunting behaviours of golden jackals (Canis aureus), which exhibit both solo and cooperative strategies. These strategies are abstracted to explore and exploit search spaces in optimization problems. Reduced population variety might cause the original GJO algorithm to become stuck in local optima, especially in complex optimization circumstances. To address this, we integrate the Opposition-based Learning (OBL) strategy with GJO, which helps maintain diversity and improve convergence toward optimal solutions. We take into account four important factors while scheduling: make span, cost, energy use, and resource use. Below is an explanation of the scheduling procedure in detail.



Figure 2: (A) Golden jackal pair, (B) The golden jackal is in search of prey (C) pursuit and capture of prey and (D,E)Attacking prey

**Step 1: Solution encoding**

Solution encoding plays a vital role in achieving effective task scheduling within cloud environments. This phase begins with the initialization of various parameters, which include the total number of tasks, the count of hosts, the number of virtual machines (VMs), CPU capacity, population size, maximum iterations, and specific parameters pertinent to the Enhanced Golden Jackal Optimization (EGJO) algorithm. The initialization process for the GJO algorithm involves creating a population of random solutions, known as Jackals, which are iteratively refined to identify the optimal solution. During the solution encoding phase, tasks are allocated to the available virtual machines in a manner that minimizes makespan, cost, and energy consumption, while simultaneously maximizing resource utilization. The structure of the solution is summarized in Table 1.

 (4)

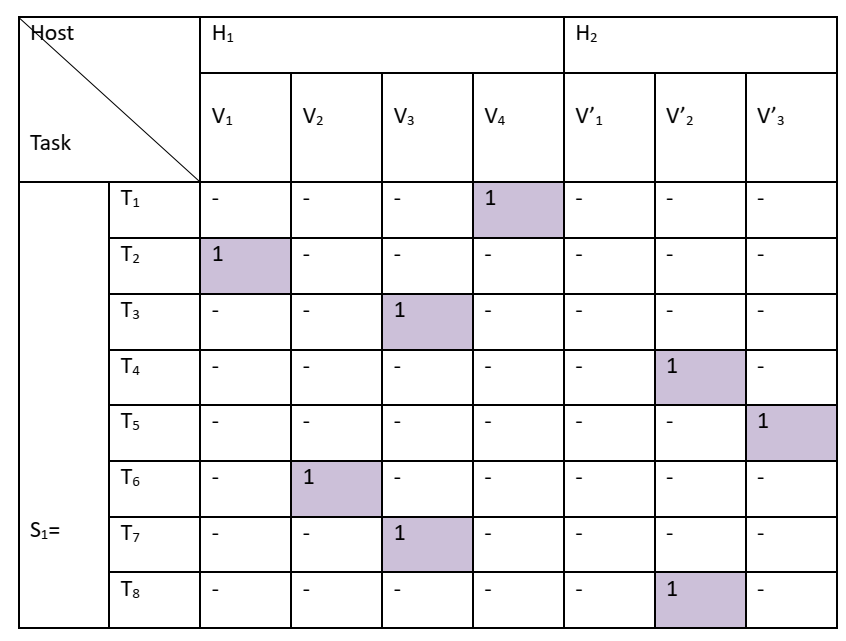


Table 1: Solution encoding format

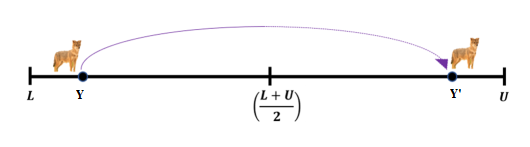
In table 1, we consider the two host (H1,H2), seven virtual machines and eight tasks (T1-T8). The H1 consist of four virtual machines (V1-V4) and H2 consist of three virtual machines (V’1-V’3). In table, “1” represent, the particular task is assigned to the corresponding VM and “-” represent the task is not assigned to this VM.

**Step 2: Opposite solution generation**

After solution initialization process, we generate the opposite solution based on OBL strategy. Opposite solution is used for increases the searching ability of search agent or devils. In OBL, a solution located in the opposite direction,, of a candidate solution,  is calculated to explore more promising regions of the search space. The solution, is calculated as shown in Equation (11)and OBL mechanism is presented in figure 2.

 (5)

Where, LB and UB represent the upper and lower bounds.



**Step 3: Fitness Evaluation:** The parameters considered for the fitness function include makespan, cost, energy consumption, and resource utilization, which are assessed for each proposed solution. This fitness function is crucial for determining the optimal solution. In this research, a minimization approach is utilized for evaluating fitness, and the fitness value is calculated according to equation (12).

 (6)

**Makespan:** The term "makespan" denotes the total duration needed to finish all tasks, representing the time interval from the beginning to the end of a collection of tasks. A smaller makespan value indicates that the scheduler is effectively assigning tasks to virtual machines (VMs), leading to optimal task organization. In contrast, a larger makespan indicates inefficiencies in task allocation. The calculation of makespan can be expressed using equation (7)

 (7)

The completion time of virtual machine is denoted as the running time after executing the last task on  virtual machine. The  is calculated using equation (8).

 (8)

Where; denotes the number of VMs and the  value is within, which denotes the number of tasks/jobs, and the value is within.

**Energy consumption: -** An efficient scheduling system minimizes energy usage while planning tasks using the available resources. The total energy consumption for scheduling is calculated using equation (9).

 (9)

Energy consumption is calculated using Equation (10) when all resources are busy.

 (10)

Where;

🡪 Time to execute the task  using resource rj

🡪 Task scheduled to be  based on voltage 

🡪Frequency of resource having voltage level s

Dynamic power loss  is evaluated using equation (11).

 (11)

Where;

🡪 Constant parameter associated with dynamic power

🡪Supply voltage

🡪 Relevant frequency of 

Energy consumption is calculated using Equation (18) when all resources are idle.

 (12)

Where;

🡪Lowest voltage of resources

🡪Lowest frequency of resources

🡪 Idle period of resource 

**Execution cost:** Execution Cost (EC) represents the cost incurred by cloud computing users when utilizing devices to perform tasks. The primary objective for users is to minimize this cost while ensuring effective resource utilization and the smallest makespan, as determined by Eq. (13).

 (13)

**Resource utilization:** Resource utilization is a performance measure that calculates the utilization of devices and resources. Higher utilization values indicate greater profit for the cloud provider, as defined by Eq. (14).

 (14)

**Step 4: Updation using** GJO**:** The golden jackal has update their solution using two phases namely, the exploration phase or searching prey and Exploitation Phase or Pouncing and Enclosing Prey

**Exploration Phase or Searching Prey**

Jackals can detect and track prey, but when the prey evades capture, they wait to hunt again. Typically, a male jackal leads the hunt while the female follows.

 (15)

 (16)

Here,  as present iteration, as vector position,  and  as current positions of female and male jackals. as refreshed positions of male and  as refreshed positions of female jackals.

The avoiding energy of prey E is computed below:

 (17)

Here,  is denoted as diminishing energy of prey,  defined as original state of energy.

 (18)

Here, specifies arbitrary value in [0, 1].

 (19)

Here, is denoted as constant, is expressed as linearly declines and T is symbolized as the maximum iteration. The is denoted as arbitrary-vector depends on Lévy-distribution and it is expressed below:

 (20)

The *LF is* designed as the fitness function of levy-flight, and it is expressed below:

 (21)

 (22)

Here,  and  as arbitrary values in [0,1],  as constant. Thus, the refreshed position of golden-jackals is expressed as:

 (23)

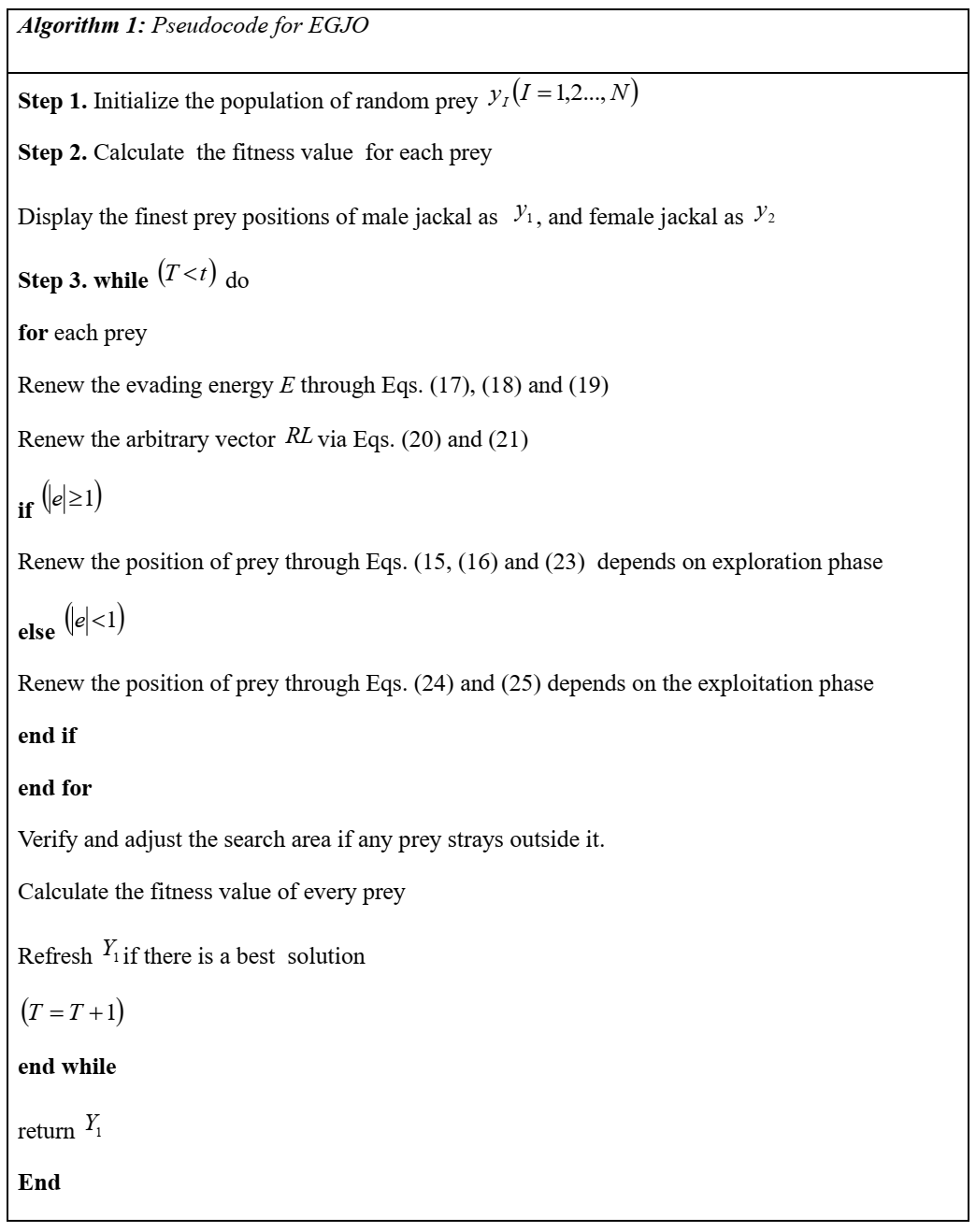
**Exploitation Phase/ Pouncing and Enclosing Prey**

When jackal is attacked by a pair, its ability to avoid prey quickly diminishes, and golden foxes immediately lock onto the prey.

 (24)

 (25)

Here, is symbolized as the position vector, is denoted as the current iteration,  as current positions of male jackals,as current positions of female jackals respectively. The prior section contains some factors including *E* and. Lastly, the position of golden-jackal is updated. Pseudocode for GJO is depicts in algorithm 1.



**Phase 3: Task distribution process**

The task distribution process represents the concluding step in the scheduling workflow. After the scheduling phase, the task manager effectively assigns each task to the most suitable resources. This allocation is guided by the criteria set in earlier stages, ensuring optimal task assignment to enhance performance and resource use. By precisely aligning tasks with appropriate resources, the task distribution process promotes efficient execution and improves overall system performance.

**5. Results and discussion**

This section presents the findings of the proposed task scheduling method. To evaluate its effectiveness, we compared our algorithm against three optimization techniques: the Dragonfly Optimization Algorithm (DA), Grey Wolf Optimization (GWO), and Gravitational Search Algorithm (GSA). The experiments were conducted on a system running Windows 10 with an Intel Core i7 processor and 8GB of RAM, utilizing Python for the implementation. The tests involved varying the number of tasks from 1,000 to 5,000. Performance assessment focused on three primary metrics: makespan, response time, and degree of imbalance (DI).

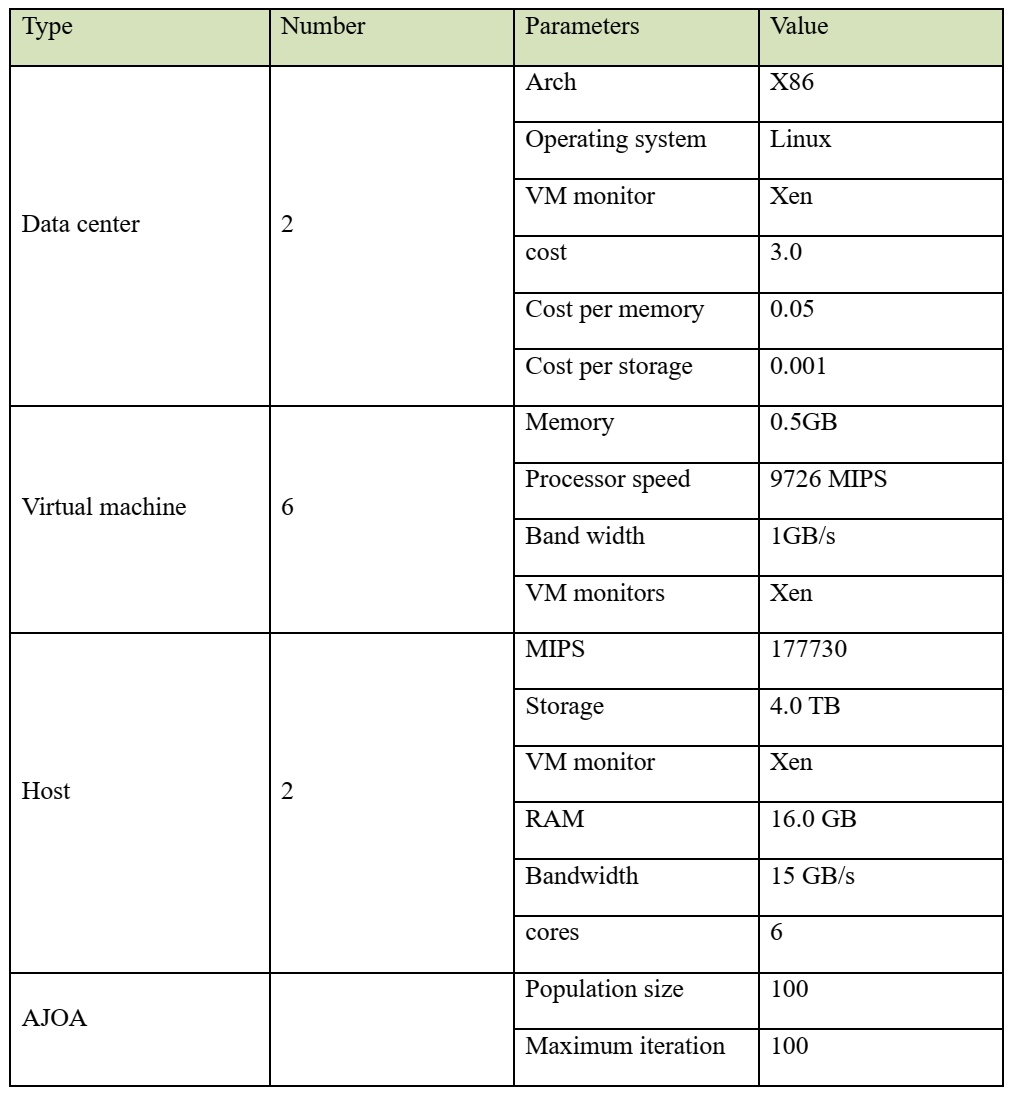


Table 2: The simulator environment

**Experimental results**

In this section, we present the results achieved by our proposed method and illustrate its effectiveness through comparisons with several alternative algorithms. For the performance evaluation, we utilize two key metrics: makespan and degree of imbalance. The calculation of the degree of imbalance follows the formulation provided in equation (30). (30)

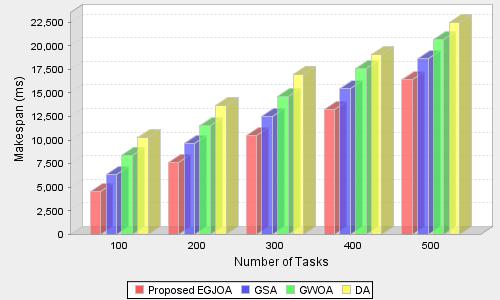


Figure 4: Performance analysis based on makespan

The figure illustrates the makespan (measured in milliseconds) for different numbers of tasks (ranging from 100 to 500) when scheduled using four algorithms: the Proposed EGJOA (in red), GSA (in blue), GWOA (in green), and DA (in yellow). As the number of tasks increases, the makespan rises for all algorithms. The Proposed EGJOA consistently outperforms the other methods, showing the lowest makespan across all task levels. At 100 tasks, the makespan for EGJOA is just under 5,000 ms, while GSA, GWOA, and DA are around 7,000 ms, 8,000 ms, and 9,000 ms, respectively. For 500 tasks, EGJOA maintains a makespan below 17,500 ms, compared to GSA, GWOA, and DA, which exceed 20,000 ms, with DA reaching the highest makespan. This demonstrates the efficiency of EGJOA in minimizing makespan across varying task loads.

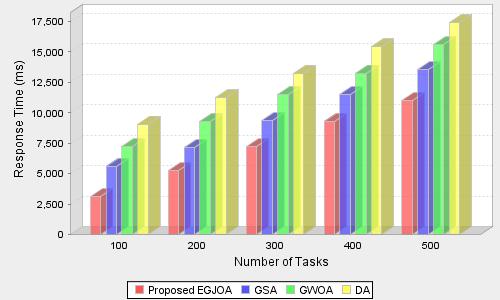


Figure 5: Performance analysis based on response time

The bar chart shows the response times of four task scheduling algorithms (Proposed EGJOA, GSA, GWOA, and DA) as the number of tasks increases from 100 to 500. The Proposed EGJOA consistently performs best, with the lowest response times, starting around 2,500 ms for 100 tasks and reaching about 10,000 ms for 500 tasks. GSA follows with higher response times, ranging from about 3,000 ms to 12,500 ms. GWOA and DA exhibit similar performance, with both increasing to over 15,000 ms for 500 tasks, with GWOA slightly outperforming DA throughout. Overall, EGJOA demonstrates superior efficiency in handling tasks compared to the others.

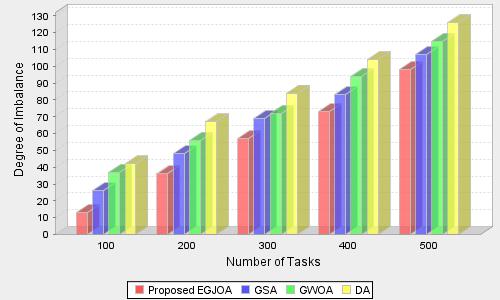


Figure 6: Performance analysis based on Degree of imbalance

The figure 6 compares the degree of imbalance in four task scheduling algorithms (Proposed EGJOA, GSA, GWOA, and DA) as the number of tasks increases from 100 to 500. The y-axis shows the degree of imbalance, ranging from 0 to 130. The Proposed EGJOA consistently achieves the lowest degree of imbalance, starting at around 10 for 100 tasks and rising to about 70 for 500 tasks. GSA shows a higher imbalance, increasing from around 15 for 100 tasks to about 90 for 500 tasks. GWOA and DA exhibit a similar trend, with DA performing the worst. Both GWOA and DA start at around 20 for 100 tasks, with DA reaching a peak imbalance of approximately 120 at 500 tasks, while GWOA is slightly lower at around 100. Overall, the Proposed EGJOA shows the most balanced task scheduling, maintaining a lower degree of imbalance compared to the others as the task count grows.

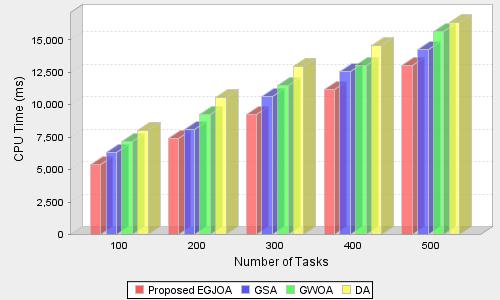


Figure 7: Performance analysis based on CPU time

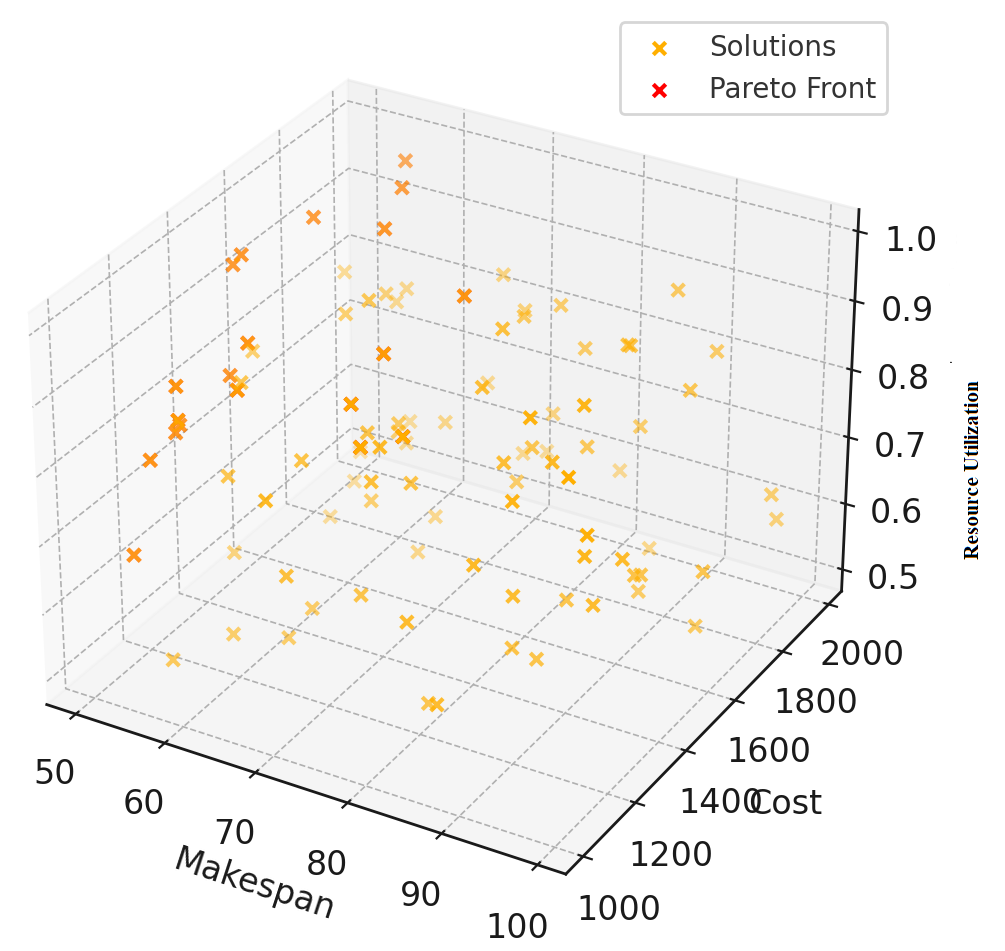


Figure 8: Pareto front representing the optimal trade-offs among makespan, cost, and resource utilization.

The figure illustrates a 3D plot representing the Pareto front for a multi-objective optimization scenario, where three key metrics—makespan, cost, and resource utilization—are analyzed. The X-axis denotes makespan (measured in days or hours), the Y-axis indicates cost (in a specified monetary unit), and the Z-axis reflects resource utilization (expressed as a percentage or ratio). Yellow crosses depict all potential solutions, while red crosses highlight the Pareto-optimal solutions forming the Pareto front. This front consists of solutions for which any improvement in one objective would result in the degradation of at least one other. Solutions positioned closer to the Pareto front signify a better balance among all three objectives, whereas those further away may reflect suboptimal trade-offs, such as decreased resource utilization accompanied by increased costs or make span.

**5. Conclusion**

This research effectively addresses significant challenges in cloud computing task scheduling, particularly for big data applications, by proposing an Enhanced Golden Jackal Optimization (EGJO) method that incorporates Opposition-Based Learning (OBL). The multi-objective approach takes into account critical parameters such as cost, makespan, energy consumption, and resource utilization, demonstrating marked enhancements in task scheduling efficiency. The integration of OBL contributes to maintaining diversity within the population and improves convergence towards optimal solutions, effectively mitigating the early convergence issues commonly found in traditional metaheuristic algorithms. Through comprehensive evaluation and comparison with existing techniques, the proposed strategy proves to be effective in optimizing task scheduling in cloud environments, fulfilling user requirements, and enhancing overall system reliability.

**Future Directions**

**Dynamic Environments:** Future work could focus on adapting the EGJO approach for highly dynamic cloud environments, where task requirements, resource availability, and user demands fluctuate frequently.

**Scalability:** The methodology can be tested in larger, more complex cloud infrastructures to evaluate its scalability and performance in real-time, large-scale big data scenarios.

**Integration of Security and QoS Metrics:** Incorporating quality of service (QoS) metrics such as security, fault tolerance, and data privacy into the multi-objective function can enhance the robustness of the scheduling approach for sensitive big data applications.

**Energy Efficiency Optimization:** Additional refinements may aim to more aggressively minimize energy consumption, aligning with green cloud computing initiatives to reduce the carbon footprint of data centers.

**REFERENCES**

1. Praveen, S.P., Ghasempoor, H., Shahabi, N. and Izanloo, F., 2023. A Hybrid Gravitational Emulation Local Search-Based Algorithm for Task Scheduling in Cloud Computing. *Mathematical Problems in Engineering*, *2023*.
2. Sun, J., Zhang, Y., Wu, Z., Zhu, Y., Yin, X., Ding, Z., Wei, Z., Plaza, J. and Plaza, A., 2019. An efficient and scalable framework for processing remotely sensed big data in cloud computing environments. *IEEE Transactions on Geoscience and Remote Sensing*, *57*(7), pp.4294-4308.
3. Swarup, S., Shakshuki, E.M. and Yasar, A., 2021. Task scheduling in cloud using deep reinforcement learning. *Procedia Computer Science*, *184*, pp.42-51.
4. Peng, Z., Pirozmand, P., Motevalli, M. and Esmaeili, A., 2022. Genetic Algorithm-Based Task Scheduling in Cloud Computing Using MapReduce Framework. *Mathematical Problems in Engineering*, *2022*.
5. Hasan, M.Z. and Al‐Rizzo, H., 2020. Task scheduling in Internet of Things cloud environment using a robust particle swarm optimization. *Concurrency and Computation: Practice and Experience*, *32*(2), p.e5442.
6. Rjoub, G., Bentahar, J. and Wahab, O.A., 2020. BigTrustScheduling: Trust-aware big data task scheduling approach in cloud computing environments. *Future Generation Computer Systems*, *110*, pp.1079-1097.
7. Abdullahi, M. and Ngadi, M.A., 2016. Symbiotic organism search optimization based task scheduling in cloud computing environment. *Future Generation Computer Systems*, *56*, pp.640-650.
8. Ming, G. and Li, H., 2012. An improved algorithm based on max-min for cloud task scheduling. *Recent Advances in Computer Science and Information Engineering: Volume 2*, pp.217-223.
9. Bhoi, U. and Ramanuj, P.N., 2013. Enhanced max-min task scheduling algorithm in cloud computing. *International Journal of Application or Innovation in Engineering and Management (IJAIEM)*, *2*(4), pp.259-264.
10. Munir, E.U., Li, J. and Shi, S., 2007. QoS sufferage heuristic for independent task scheduling in grid. *Information Technology Journal*, *6*(8), pp.1166-1170.
11. Rjoub, G., Bentahar, J., Abdel Wahab, O. and Saleh Bataineh, A., 2021. Deep and reinforcement learning for automated task scheduling in large‐scale cloud computing systems. *Concurrency and Computation: Practice and Experience*, *33*(23), p.e5919.
12. Hasan, M.Z. and Al‐Rizzo.,2020. Task scheduling in Internet of Things cloud environment using a robust particle swarm optimization. *Concurrency and Computation: Practice and Experience*, *32*(2), p.e5442.
13. Mukherjee, D., Ghosh, S., Pal, S., Aly, A.A. and Le, D.N., 2022. Adaptive Scheduling Algorithm Based Task Loading in Cloud Data Centers. *IEEE Access*, *10*, pp.49412-49421.
14. Ben Alla, H., Ben Alla, S., Touhafi, A. and Ezzati, A., 2018. A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment. *Cluster Computing*, *21*(4), pp.1797-1820.
15. Huang, X., Li, C., Chen, H. and An, D., 2020. Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies. *Cluster Computing*, *23*, pp.1137-1147.
16. Bal, P.K., Mohapatra, S.K., Das, T.K., Srinivasan, K. and Hu, Y.C., 2022. A joint resource allocation, security with efficient task scheduling in cloud computing using hybrid machine learning techniques. *Sensors*, *22*(3), p.1242.
17. Pal, S., Jhanjhi, N.Z., Abdulbaqi, A.S., Akila, D., Alsubaei, F.S. and Almazroi, A.A., 2023. An Intelligent Task Scheduling Model for Hybrid Internet of Things and Cloud Environment for Big Data Applications. *Sustainability*, *15*(6), p.5104.
18. Ding, D., Fan, X., Zhao, Y., Kang, K., Yin, Q. and Zeng, J., 2020. Q-learning based dynamic task scheduling for energy-efficient cloud computing. *Future Generation Computer Systems*, *108*, pp.361-371.
19. Arini, F. Y., Sunat, K., &Soomlek, C. (2022). Golden jackal optimization with joint opposite selection: An enhanced nature-inspired optimization algorithm for solving optimization problems. IEEE Access, 10, 128800-128823.