Energy Efficient Workflow Allocation in Cloud Computing Using Improved Grey Wolf Optimization

Md. Mazhar Nezami, Anoop Kumar
Department of Computer Science-College of Computing and Mathematics, Banasthali Vidyapith, Rajasthan, India

***Abstract*—Cloud computing has emerged as a dominant platform for hosting complex applications, offering scalable and flexible resources on demand. However, the dynamic and heterogeneous nature of cloud environments poses significant challenges for efficient workflow scheduling, particularly when aiming to minimize total execution time, energy consumption, and operational cost. In this research, we propose a novel hybrid approach that integrates the Heterogeneous Earliest Finish Time (HEFT) algorithm with an Improved Grey Wolf Optimizer (IGWO) enhanced by differential evolution strategies and survival-of-the-fittest mechanisms. These enhancements strengthen exploration and exploitation by adaptively mutating and refining task allocations while eliminating weaker solutions. The use of HEFT-based initialization provides a strong starting population, and the DE-driven IGWO refinement accelerates convergence and avoids premature stagnation. Together, these two-level optimization strategy ensures faster convergence and higher energy-efficient workflow scheduling compared to earlier HEFT metaheuristic approaches. To evaluate the effectiveness of the proposed hybrid method, extensive experiments were conducted on randomly generated workflows with varying task and dependency complexities. The performance analysis demonstrates that the hybrid HEFT-IGWO approach consistently outperforms standard HEFT, traditional GWO, and standalone metaheuristic techniques in terms of minimizing makespan, reducing energy consumption, and lowering cloud infrastructure costs. This study highlights the potential of combining heuristic initialization with evolutionary optimization to achieve energy-efficient, cost-effective workflow scheduling in cloud computing environments.**

***Keywords*—Cloud computing; energy efficient; workflow; Heterogeneous Earliest Finish Time (HEFT); Grey Wolf Optimization (GWO); makespan; cost**

I. Introduction

Cloud computing has revolutionized the way computational resources are provisioned and consumed, offering scalable, on-demand access to a vast pool of virtualized hardware and software services. It enables organizations to deploy complex applications and workflows without investing heavily in physical infrastructure [1], [2].

A workflow in cloud computing represents a set of interdependent tasks organized to achieve a specific computational objective, such as scientific workflows, big data processing workflows. The workflow scheduling refers to organizing workflow tasks on VMs within the cloud platform, addressing the challenge of balancing energy efficiency with QoS optimization [3].

As cloud environments grow larger and more heterogeneous, efficient workflow allocation, assigning tasks to appropriate virtual machines (VMs), becomes critical. Poor task allocation results in high direct and indirect energy consumption, leading to high costs, reduced reliability, and environmental issues like CO2 emissions [4][5]. If a data center hosts 10,000 servers and each consumes an average of 300 watts, the total energy consumption can reach 3 megawatts per hour. At an average electricity cost of $0.12 per kilowatt-hour, operating such a centre would cost approximately $360 per hour, not including cooling and infrastructure overheads.

Despite significant progress in workflow scheduling, existing algorithms still struggle to maintain a balance between performance efficiency and energy optimization. Traditional heuristics, such as the HEFT algorithm, offer fast scheduling decisions but are limited by their static and local search behaviour. Conversely, metaheuristic algorithms like Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Grey Wolf Optimizer (GWO) improve global search ability but may converge prematurely or require many iterations.

Therefore, there is a critical need for a hybrid scheduling framework that can combine heuristic efficiency with adaptive global optimization to effectively minimize makespan, energy consumption, and operational cost in dynamic, heterogeneous cloud environments [6].

This study seeks to determine how heuristic and metaheuristic methods can be effectively integrated to enhance workflow scheduling efficiency in cloud computing. It also incorporates differential evolution and survival-of-the-fittest mechanisms within an Improved Grey Wolf Optimizer (IGWO) can strengthen convergence stability and solution diversity. Furthermore, the research investigates the extent to which the proposed hybrid HEFT–IGWO model can reduce makespan, energy consumption, and operational cost compared to traditional scheduling techniques. This research contributes to the advancement of green and sustainable cloud computing by proposing an intelligent hybrid model that improves both computational performance and energy efficiency. The proposed method offers a practical solution for multi-objective workflow allocation, reducing makespan, energy consumption, and cost simultaneously.

The study is designed in the rest of the sections as follows: Section II presents a literature survey. Section III presents the problem definition, problem formulation, and suggested HEFT-IGWO. Section IV describes the detailed algorithmic design, including initialization, mutation, crossover, and survival mechanisms of the HEFT–IGWO approach. Section V provides

the experimental setup, simulation results, and performance analysis, comparing the proposed method with existing approaches in terms of makespan, energy consumption, and cost. Finally, Section VI concludes the study, summarizing key findings and outlining future research directions.

II. Literature Survey

Many studies have examined methodologies for time and cost-efficient workflow allocation in a cloud computing environment, with limited focus on energy. The hybrid strategies that integrate heuristics like HEFT with metaheuristics have shown promising results in minimizing makespan, energy consumption, and cost simultaneously, forming a strong motivation for the hybrid HEFT-IGWO approach proposed in this study.

Li H et al. [7] proposed the SEPSO (Swarm Entropy-based Particle Swarm Optimization) framework. It is designed to optimize cloud provider profits by minimizing the monetary cost of energy and leased public cloud resources. SEPSO monitors swarm diversity during iterations to avoid premature convergence and enhance exploration by dynamically adjusting search parameters to balance exploration and exploitation throughout the optimization process. Although it improves task sequencing and allocation across public and private clouds, SEPSO mainly targets cost reduction from the provider's view and lacks adaptability to dynamic or large-scale workflows.

Li H et al. [8] proposed the Chaotic-nondominated-sorting Owl Search Algorithm (COSA) to schedule resource-constrained multiple workflows. It combines an Owl Search Algorithm (OSA) with an NSGA-II to optimize makespan, cost, and energy under the specified deadline and budget constraints in hybrid clouds. For better balancing of exploration and exploitation, multiple workflows are scheduled in DVFS-enabled cloud using a hierarchical evolving mechanism that employs COSA for the Worse Half of Population (WHP) and NSGA-II for updating the Better Half of Population (BHP). However, its reliance on fixed workflow and resource settings limits adaptability in dynamic cloud environments, and chaotic control increases computational overhead.

Thekkepuryil et al. [9] proposed a hybrid algorithm combining ALO and PSO to enhance workflow scheduling to optimize key metrics like execution time, cost, and load balancing in cloud environments. The Data Encryption Standard (DES) ensures data security during scheduling, addressing both performance and confidentiality concerns. The hybrid algorithm combines the global search ability of ALO with the local search refinement of PSO. However, its reliance on static workflows and fixed resources limits adaptability in real-world dynamic cloud environments, and the inclusion of DES adds computational overhead.

Saeedi et al. [10] optimizes task-resource mapping in workflows through the following steps. Initialize particles representing task-resource mappings, along with their positions and velocities, evaluate each particle, store non-dominated solutions generated during the search process in external archive, iteratively updates the archive, ideal point, and hyperplane. The crowding distance determines which solutions remain in the archive when it is full. The archive of optimized solutions is returned. However, I-MaOPSO reliance on roulette wheel leader selection can lead to inaccurate results when individuals have similar fitness values, reducing diversity and convergence stability.

Alaei et al. [11] proposed an adaptive fault detection method in cloud computing based on the Improved Differential Evolution (IDE) algorithm combined with an ANFIS prediction model. This hybrid model enhances reliability by predicting faults before they occur, helping reduce makespan, cost, and energy consumption while improving fault tolerance. However, the model primarily addresses VM faults and overlooks other significant reliability issues, such as network, storage, and I/O failures.

Mohammadzadeh et al. [12] proposed the HGALO-SCA algorithm, which addresses the limitations of ALO and SCA algorithms by leveraging their strengths. HGALO-SCA enhances ALO searchability by incorporating SCA oscillatory behaviour, allowing global exploration. The integration of the ALO elite strategy accelerates the convergence while maintaining global search accuracy. The comparison with SPEA2 shows better performance in balancing trade-offs. However, random chaos parameters may also increase computational complexity and reduce consistency in results.

Yao et al. [13] proposed Endocrine-based Coevolutionary Multi-Swarm for Multi-Objective Optimization (ECMSMOO) addresses the workflow scheduling in cloud computing, which is an NP-complete problem by optimizing multiple conflicting objectives, including execution time, cost, and energy consumption. The manager server collects available cloud resources to reduce the impact of elastic resource fluctuations during scheduling. The endocrine-based evolutionary strategy mimics hormone-regulated particle behaviour to enhance search efficiency and avoid local optima. The swarms collaborate and compete to ensure a strong search process across the multi-objective optimization space. Furthermore, the hormonal control mechanism adds algorithmic complexity and overhead, which can hinder real-time scheduling.

Mohammadzadeh et al. [14] introduced an improved version of the GWO algorithm, known as HCGWO. This improvement achieves better optimization outcomes by combining chaos theory and the hill-climbing method. The study extends the proposed IGWO algorithm to a binary version specifically designed for the workflow allocation problem. This involves utilizing various S and V functions to deal with the workflow allocation problem with the goal of reducing execution costs, makespan, and power consumption. chaotic map generation introduces additional computational overhead.

Hassan et al. [15] proposed Smart Energy and Reliability Aware Scheduling (SERAS) algorithm addresses the dual challenge of energy efficiency and system reliability in cloud computing environments. By integrating the DVFS technique, SERAS dynamically adjusts the processor frequencies of VMs while ensuring tasks meet their deadlines. The SERAS divides the deadline across tasks, enabling more precise scheduling and resource allocation. However, algorithm complexity ($O(n^2)$) limits scalability for large workflows.

Belgacem A, Beghdad-Bey K [16] suggested HEFT-ACO, which effectively reduces costs and makespan by combining the ACO algorithm with the HEFT heuristic. The simulation results were conducted on the Amazon EC2 cloud platform using three real-world scientific workflows, viz. Montage, CyberShake, and Ligo show better trade-offs between makespan and cost by leveraging the strengths of both techniques. The workflow characteristics, viz., balance or asymmetry, are not considered.

Taghinezhad A et al. [17] introduce BDCE and BDD, two energy-efficient heuristic algorithms for workflow scheduling in cloud environments, aiming to minimize energy consumption while meeting budget and deadline constraints, target resources with DVFS capabilities, while BDCE works with non-DVFS-enabled resources to optimize cost, scheduling length, and energy savings while ensuring QoS compliance. The result metrics are compared against established methods like BDSD, DBCS, BDHEFT, ERES, and the Safari algorithm using scientific workflow applications, showing improvements in energy savings, cost efficiency, and scheduling performance. However, both BDCE and BDD focus primarily on medium-scale workflows.

Khaleel M [18] proposed the ELSCiW (Energy-Latency-aware Scientific Workflow scheduling) framework focuses on optimizing cloud workflow scheduling by balancing energy consumption and latency while maintaining high QoS. The approach involves a two-step optimization process: first, node efficiency evaluation, which compares the number of handled transactions to power consumption, optimizing processor utilization, and then task mapping with mean GWO, which applies to map tasks to cloud resources efficiently. However, it primarily focuses on the energy–latency trade-off.

Zeedan et al.[19] proposed Enhanced Binary Artificial Bee Colony-based Pareto Front (EBABC-PF) to optimize makespan, processing cost, and resource utilization without violating SLA. The algorithm uses HEFT to prioritize the work, Greedy Randomized Adaptive Search Procedure (GRASP) to develop an initial solution, and Enhanced Binary Artificial Bee Colony (BABC) to schedule tasks onto VMs. Moreover, it emphasizes performance and cost but gives limited attention to energy efficiency.

III. Problem Definition

When a user submits a workflow application as a Directed Acyclic Graph (DAG), as shown in Fig. 1, the workflow interface converts the DAG into an admission queue that maintains execution order based on dependencies. The workflow scheduler allocates tasks to VMs using a scheduling algorithm, optimization criteria for energy efficiency, and a resource estimator to match tasks with suitable resources.

A. Resource Model

In our cloud computing model, the infrastructure comprises a collection of cloud servers (CS), each hosting multiple computing nodes on which VMs are deployed. A centralized global cloud manager oversees the operation, consisting of a resource manager and a scheduler. The resource manager maintains real-time status and key attributes of all available VMs, such as processing speed, energy consumption, and reliability. This information is utilized by the scheduler to efficiently assign tasks when a workflow application is submitted, based on task dependencies and VM availability.

All VMs are assumed to support Dynamic Voltage Scaling (DVS), allowing them to adjust their voltage and frequency levels dynamically. This enables energy savings by switching to the lowest voltage levels during idle periods. The energy consumption model is based on the CMOS power dissipation model, where energy usage is influenced by both the characteristics of the device and the voltage supply associated with each task.

The energy consumption of tasks on VMs during active (busy) execution time and when the VMs are idle can be calculated using:

$$E_{busy} = \sum_{j=1}^{M} \alpha * V_j^2 * ET(T_i, VM_j)$$

$$E_{idle} = \sum_{j=1}^{M} \alpha * V_{j,low}^2 * \Delta idle_j$$

where, n is the total number of tasks, VM_j is the virtual machine on which task $T_i$is executed, V_j is the supply voltage of VM_j, and ET (T_i, VM_j) is the execution time of task T_i on VM_j.$VM_{j,low}$is the minimum voltage level on VM_j, and $\Delta idle_j$ is the amount of idling time for VM_j. The total energy consumption can be calculated as:

$$\text{TEC}=E_{busy}+ E_{idle}$$

B. Application Model

A DAG represents a workflow application represented by Wf = (T, E), where $T = \{T_i, 1 \le i \le \text{N}\}$, is a set of tasks of the workflow, and E is the set of edge characterizes precedence constraints between tasks. The edge $T_i \rightarrow T_j$,shows the precedence relation between T_i and T_j in the DAG as shown in Fig. 2, which will be processed on virtual machines $V = \{VM_j \mid 1 \le j \le \text{M}\}$ with various computing speed and cost. The execution time of task T_i on VM_j is denoted by ET (T_i, VM_j and CT (T_i, T_k) denotes communication. According to the dependencies among the tasks, task T_i (child task) will execute after $Pred(T_i)$, (parent tasks of T_i). The start time ST (T_i, VM_j), finish time $FT(T_i, VM_j)$ of a task T_i on VM_j, and the makespan(MS) are calculated as follows:

$$ST(T_i, VM_j) = max\ \{Avl(VM_j),\ max\{FT[Pred(T_i)] + CT(Pred(T_i), T_k)\}$$

$$FT(T_i, VM_j) = ST(T_i, VM_j) + ET(Pred(T_i), VM_j)$$

$$Makespan\ (MS) := max \sum_{i=1}^{N} FT(T_i, VM_j)$$

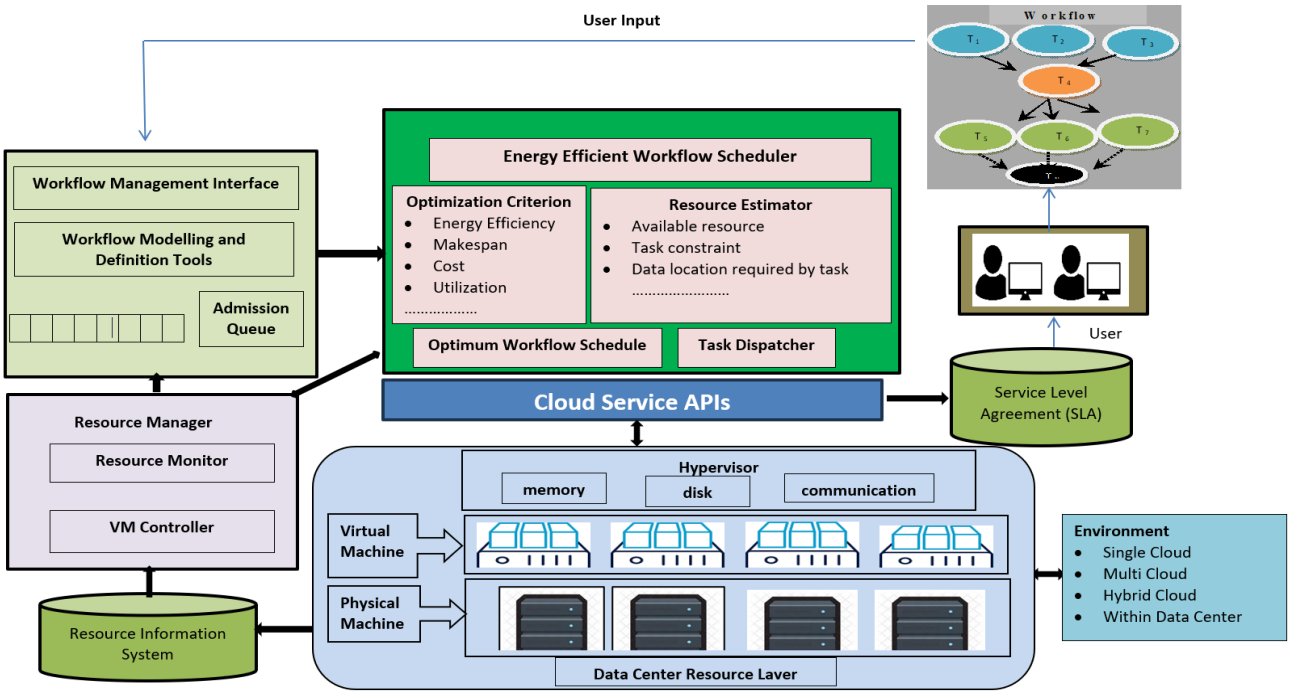

Fig. 1. Workflow allocation framework.

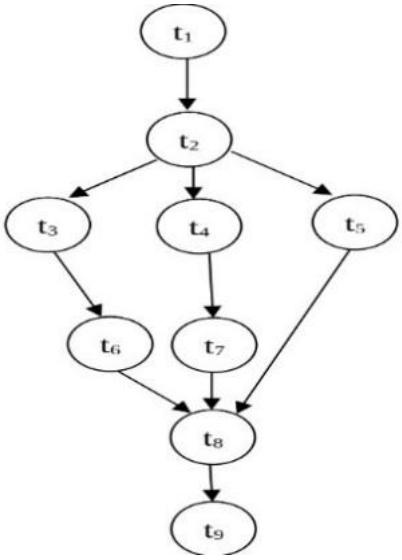

Fig. 2. Sample workflow.

C. Cost Model

The pricing in cloud computing is determined by the billing interval (BI) for resource leasing. As a result, even if only a portion of the time is used, the client must pay for the full interval.

$$Cost(T_i, VM_j) = \sigma * \left\lceil \frac{ET\ (T_i, VM_j)}{\tau} \right\rceil * P_0 * e^{\frac{cpu\ cycles\ of\ VM_j}{slowest\ cpu\ cycle}}$$

where, σ is a random variable used to generate different combinations of VM pricing and capacity, τ unit chargeable time, P_0 is the base price of VM. The total execution cost can be defined as:

Let Bi, j be a Boolean variable, such that

$$Bi,j = \begin{cases} 1, & if\ task\ T_i\ is\ assign\ to\ V_j \\ 0, & otherwise \end{cases}$$

$$TCost = \sum_{i=1}^{N} \sum_{j=1}^{M} Bi,j * Cost(T_i, VM_j)$$

D. Problem Definition

A schedule (Sc) is presented as Sc = (*SVM, Allocation, MS*, $TCost$, TEC), where *SVM* corresponds to virtual machines, *Allocation* represents the task to VM mappings, $TCost$: Total execution cost, *MS*: Makespan, and TEC: Energy consumption of the workflow. The problem can be defined as:

Minimize:

$$F = \gamma \text{ x } MS + \vartheta \text{ x } TCost + (1 - \gamma - \vartheta) \text{ x } TEC$$

Subject to

$$i)\ \sum_{j=1}^{M} Bi,j = 1,\ i = 1,2,3.....n$$

$$ii)\ 0 < \gamma, \vartheta < 1$$

Constraints: i) indicate that any task can be assigned to only one VM and the constraint, and ii) limit the range of γ, ϑ that balances optimization functions.

IV. Algorithm

The Grey Wolf Optimization algorithm is a meta-heuristic optimization technique that was inspired by the social hierarchy and hunting habits of grey wolves. It was first presented by Mirjalili et al. in 2014 [20]. We propose an energy-efficient workflow allocation algorithm based on improving the Grey Wolf Optimizer is a version of GWO that has been created to improve its search performance. This algorithm has gained significant attention due to its simplicity, flexibility, and

effectiveness in solving various complex optimization problems. GWO mimics the leadership structure and collaborative hunting strategy of grey wolves, which enhances its ability to explore and exploit the search space efficiently. In a grey wolf pack, there are four types of wolves, as shown in Fig. 3, based on their social hierarchy:

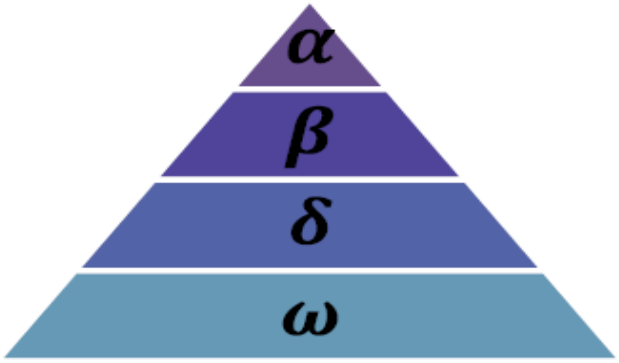

Fig. 3. Hierarchy flow up for the GWO algorithm.

Alpha (α): The leader of the pack, responsible for making decisions and guiding the hunting process.

Beta (β): The second-in-command, assisting the alpha in decision-making and maintaining discipline in the pack.

Delta (δ): The third level in the hierarchy, responsible for reporting to the alpha and beta and assisting them in their tasks.

Omega (ω): The lowest-ranking wolves, following the orders of the higher-ranking wolves and playing a crucial role in the pack's social structure.

The following mathematical models of GWO are useful in different scenarios of any real-world application.

1) Encircling prey: Grey wolves encircle their prey during the hunt. This encircling behaviour is mathematically modelled as follows:

$$\vec{D} = |\vec{C} * \overrightarrow{X_p}(t) - \vec{X}(t1)|$$

$$\vec{X}(t+1) = \overrightarrow{X_p}(t) - \vec{A} * \vec{D}$$

Here, t represents the current iteration, $\overrightarrow{X_p}$ is the position vector of the prey, $\vec{X}$ is the position vector of a grey wolf, and $\vec{A}$ and $\vec{C}$ are coefficient vectors calculated as:

$$\vec{A} = 2\,\vec{a} * \overrightarrow{r_1} - \vec{a}$$

$$\vec{C} = 2\,\overrightarrow{r_2}$$

$$\mathrm{a} = 2\left(1 - \frac{t}{T}\right)$$

In these equations, $\vec{a}$ linearly decreases from 2 to 0 over the course of iterations, while $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ are random vectors within the range [0, 1].

2) Hunting: The hunting phase involves following the best individuals in the pack, specifically the α, β, and δ wolves. The positions are updated according to these leaders as:

$$\overrightarrow{D_\alpha} = |\overrightarrow{C_1} * \overrightarrow{X_\alpha} - \vec{X}|$$

$$\overrightarrow{D_\beta} = |\overrightarrow{C_2} * \overrightarrow{X_\beta} - \vec{X}|$$

$$\overrightarrow{D_\delta} = |\overrightarrow{C_3} * \overrightarrow{X_\delta} - \vec{X}|$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1} * \overrightarrow{D_\alpha}$$

$$\overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2} * \overrightarrow{D_\beta}$$

$$\overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3} * \overrightarrow{D_\delta}$$

The position of a grey wolf is then updated as follows:

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3}$$

3) Attacking prey: The attacking phase is indicated by reducing the value of $\vec{a}$ as iterations progress. When $\vec{A}$ falls within the range [-1, 1], wolves are encouraged to attack the prey by shrinking their encircling behaviour.

Search for Prey: When $|\vec{A}|>1|$, wolves diverge to explore new areas.

The improved GWO using differential evolution for stronger exploration and diversification through mutation, crossover, and survival of the fittest for maintaining the best-performing solutions and eliminating poor-quality candidates. Additionally, we introduce a hybrid initialization technique by incorporating a solution generated using the HEFT algorithm, ensuring that the search starts from at least one good-quality position. The population of N wolves (solutions) is initialized randomly. Each wolf represents a mapping of tasks to VMs. To enhance quality and convergence, one individual is replaced by a HEFT-based solution, leveraging domain knowledge to start the search from a near-optimal region. Each solution is evaluated using a fitness function based on makespan, cost, and energy. Then sort the wolves based on their fitness values and choose the best $\overrightarrow{X_\alpha}$, second best $\overrightarrow{X_\beta}$, and third best $\overrightarrow{X_\delta}$. During each iteration, the positions of the wolves are updated using the DE strategy. This involves mutating the positions to create new candidates, crossing over elements between pairs of candidates to combine their features, and selecting the best candidates to form the next generation.

4) Mutation operation: The mutation operation of differential evolution is its most significant feature. To generate variance in this process, two weighted difference vectors are appended to the selected individual. The difference vector from the parents, which consists of two different individuals $(X_{r_1}^t, X_{r_2}^t)$ from the parent generation (the t-th generation), is the basic component of the DE variation mechanism. The definition of the difference vector is as follows:

$$Dd_{r_{12}} = X_{r_1}^t - X_{r_2}^t$$

where, r_1, r_2 are index numbers of distinct population members, and as a result, the mutation operation can be written as follows:

$$V_i^{t+1} = X_{r_3}^t + \mathrm{F} * (X_{r_1}^t - X_{r_2}^t)$$

where, F is the scaling factor, and r_1, r_2 and r_3 are different integers in the scope $(1, 2, \ldots n)$ from the current target vector index(i).

5) Crossover operation: A crossover operation is carried out using the mutation vector X_i^{t+1} for the individual X_i^t wolves, producing a trial individual U_i^{t+1}. A random selection technique is used to make sure that at least one bit of U_i^{t+1} is derived from V_i^{t+1} to ensure that X_i^t evolves. The crossover probability factor (CR) for the remaining bits of U_i^{t+1} determines whether each bit originates from X_i^t or V_i^{t+1}. The expression for the crossover operation as follows:

$$U_{ij}^{t+1} = \begin{cases} V_{ij}^{t+1}, rand(\mathrm{j}) \leq CR \ OR \ J = randn(i) \\ X_{ij} \quad , rand(\mathrm{j}) > CR \ AND \ J \neq randn(i) \end{cases}$$

$$J = 1, 2, .., \mathrm{D}$$

where, rand(j) ∈ [0, 1] obeys the random-uniform distribution, j is the j-th variable (gene), CR is the crossover probability, and rand(i) ∈ [1, 2, … D].

6) Selection operation: The greedy choice used in the selection process. Following the mutation and crossover operations, the trial individual U_i^{t+1} is created and then compared with the target individual X_i^t. The comparison can be expressed numerically as:

$$X_i^{t+1} = \begin{cases} U_i^{t+1}, & f(U_i^{t+1}) < f(X_i^t) \\ X_i^t & f(U_i^{t+1}) \geq f(X_i^t) \end{cases} \quad i = 1,2, \ldots n$$

In summary, the proposed enhances the original GWO by incorporating evolutionary principles and the DE strategy, significantly improving its search performance. This improved algorithm effectively balances exploration and exploitation, making it a powerful tool for solving complex optimization problems.

An illustration to explain HEFT-IGWO, traditional GWO, and HEFT has been presented in this section with a random workflow consisting of nine tasks with precedence among themselves, as shown in Fig. 2. We have taken only three VMs, each with different computing capacities, energy consumption, and cost, but there may be more in real scenarios. Each wolf in the GWO represents a possible allocation of tasks to VMs.

V. Results and Discussion

We use simulation for the performance study of proposed algorithms with the traditional GWO algorithm and HEFT algorithms. The simulation was performed using Python on a system with an i5 processor having a frequency of 2.1 GHz and 16 GB RAM running Windows 11. A random workflow application used to assess the HEFT-IGWO algorithm. Table I shows the simulation parameters and their corresponding values.

TABLE I Simulation Parameters

Parameter	Values
Tasks (N) number	5-100
VMs (M) number	3
Processing Speed	1000-5000 MIPS
VM distances	1-100
Computing Capacity of VM	30-1000
Inter task communication	1-1000

This section provides an overview of the performance findings in terms of makespan, energy, and cost, corresponding to three virtual machines. The performance of the three algorithms was evaluated in terms of makespan for different numbers of tasks ranging from 5 to 100 is shown in Fig. 4. The HEFT algorithm serves as the baseline for comparison.

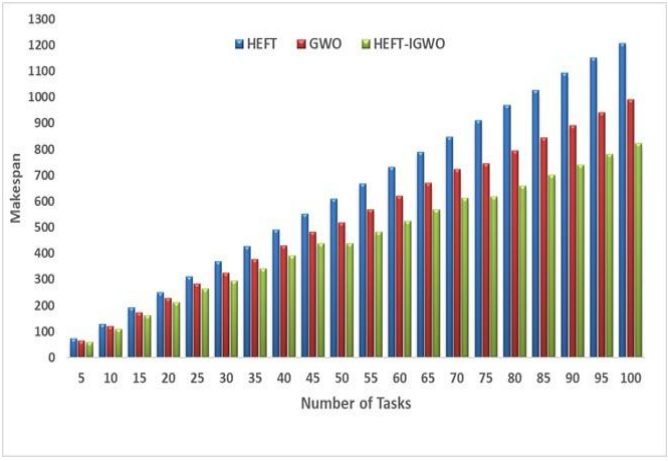

Fig. 4. Makespan vs. Number of workflow tasks.

Its makespan values increase almost linearly with the number of tasks. For small workflows (5 to 25 tasks), HEFT performs reasonably well because the search space is limited, and its ranking and earliest finish time heuristic is efficient. However, as the number of workflow tasks increases (50 to 100), HEFT performance degrades significantly, producing the highest makespan values among the three methods. For small to medium workflows (5 to 45 tasks), GWO provides an average improvement of around 8–12% in makespan. As the number of workflow tasks increases, GWO advantage becomes more significant due to its iterative exploration and leader based hunting mechanism, which helps find more balanced task–VM mappings. For larger workloads (50 to 100 tasks), the improvement rises to 15–18%. The Hybrid HEFT–IGWO approach achieves the best performance across all workflows, demonstrating the benefit of combining HEFT based initialization with improved GWO exploration through differential evolution operators and survival of the fittest. For small and medium workflows (5 to 45 tasks), HEFT–IGWO improves makespan by 15–20% compared to HEFT. For large workflows (50 to 100 tasks), the improvements become more substantial, ranging from 28–32%. These significant gains are due to the ability to start with a near optimal HEFT solution and then further refine allocations using DE mutation, crossover, and iterative leader updates.

ALGORITHM 1: HEFT-IGWO

Input: Workflow W (T, E), VM set V, population size N, generations G
Output: Best schedule with minimized makespan, cost, and energy
Phase 1: Initialization
Compute upward ranks of tasks using HEFT ordering.
Seed HEFT-based elite solution as one candidate wolf.
Generate N-1 random task-to-VM mappings as additional wolves.
Evaluate fitness of all wolves (makespan, energy, cost).
Identify alpha, beta, delta leaders (best three wolves).

Phase 2: Iterative Hybrid Evolution
For gen = 1 To G do
Update wolves positions using GWO encircling and hunting rules.
Apply higher probability mutation (random task VM flips).
Perform DE style discrete crossover between wolves and mutants.
Evaluate fitness of offspring (makespan, cost, energy).
Apply survival-of-the-fittest: keep best between parent and offspring.
Update leaders alpha, beta, delta based on new population.
Log convergence metrics (fitness, makespan, energy, cost).

Phase 3: Termination
Return alpha wolf (best schedule) with final metrics.
end

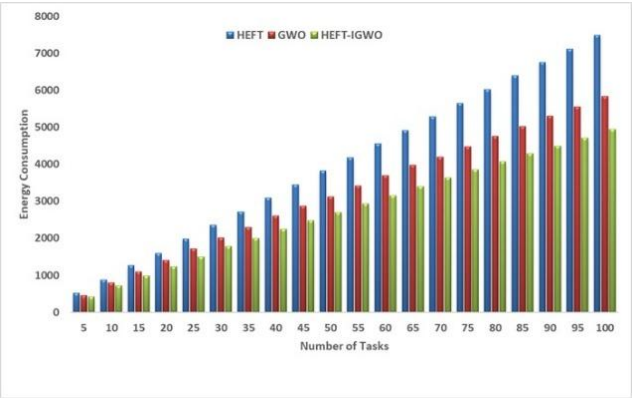

Fig. 5. Energy consumption vs. Number of workflow tasks.

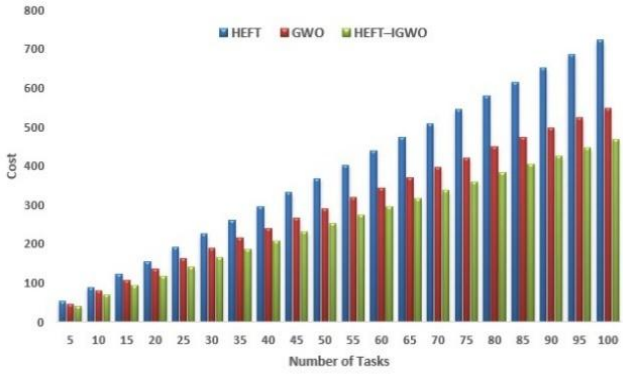

Fig. 6. Cost vs. Number of workflow tasks.

All algorithms show increasing energy consumption as tasks increase, but the growth rate is significantly slower for GWO and HEFT–IGWO than HEFT as shown in Fig. 5. The HEFT algorithm consistently consumes the highest amount of energy. The GWO achieves clear energy reductions across all task sizes, averaging around 10–15% for smaller tasks (5–30) and 18–22% for larger workloads (50–100). The hybrid HEFT–IGWO algorithm (see Algorithm 1) consistently outperforms both HEFT and GWO, demonstrating the largest energy savings across all workflow sizes. For small and medium workflow (5–40 tasks), energy savings are around 20–27%, while for large workloads (50–100 tasks), savings increase to 30–34%. HEFT–IGWO achieves the largest and most consistent energy reductions, particularly for large scale workflows, making it ideal for energy-aware workflow allocation in cloud environments.

Since HEFT focuses on earliest finish time without explicitly optimizing cost, it often results in inefficient VM usage leading to higher costs as workflows scale as shown in Fig. 6. The GWO consistently reduces operating cost compared to HEFT. For the small workflows (5–25 tasks), the cost improvements are in the range of 10–14%. For medium workloads (30–60 tasks), cost improvements increase to around 17–22%, and for larger workloads (65–100 tasks), savings stabilize between 22–24%. The hybrid HEFT–IGWO algorithm provides the largest cost reductions across all task sizes. For small tasks, improvements are already noticeable at ~20–22%, increasing to 26–30% for medium workflows, and reaching ~35% for larger workloads.

VI. Conclusion

This study presented a comparative evaluation of three workflow scheduling techniques HEFT, GWO, and the proposed Hybrid HEFT–IGWO to enhance performance and energy efficiency in cloud computing environments. The proposed HEFT–IGWO hybrid delivers the most notable results. By initializing the search with a HEFT based schedule and refining it through differential evolution and survival-of-the-fittest operations, it effectively balances exploration and exploitation. The hybrid model achieves up to 32% reduction in makespan, 34% lower energy consumption, and 35% cost savings, outperforming both baseline methods across all task scales. Overall, combining heuristic and evolutionary intelligence significantly enhances workflow allocation efficiency. HEFT–IGWO inherits the computational speed of HEFT while leveraging IGWO adaptive global search to achieve optimized, energy aware scheduling. Future work will focus on extending this approach to dynamic, heterogeneous, and QoS driven cloud environments for real time workflow.

References

[1] B. Furht and A. Escalante, Eds., Handbook of Cloud Computing. Boston, MA: Springer US, 2010. doi: 10.1007/978-1-4419-6524-0.

[2] M. Alam, S. Mustajab, M. Shahid, and F. Ahmad, "Cloud Computing: Architecture, Vision, Challenges, Opportunities, and Emerging Trends," in 2023 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), IEEE, Nov. 2023, pp. 829–834. doi: 10.1109/ICCCIS60361.2023.10425507.

[3] M. M. Nezami, A. Kumar, M. Shahid, and M. M. Nezami, "Analysis of Energy Efficient Workflow Allocation in Cloud Computing," in 2023 IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT), IEEE, Apr. 2023, pp. 803–809. doi: 10.1109/CSNT57126.2023.10134644.

[4] P. A. Malla, S. Sheikh, M. Shahid, and S. U. Mushtaq, "Energy-efficient sender-initiated threshold-based load balancing e-STLB in cloud computing environment," Concurr Comput, vol. 36, no. 5, Feb. 2024, doi: 10.1002/cpe.7943.

[5] M. Sajid and Z. Raza, "Energy-aware stochastic scheduling model with precedence constraints on DVFS-enabled processors," Turkish Journal of Electrical Engineering and Computer Sciences, vol. 24, no. 5, pp. 4117–4128, 2016, doi: 10.3906/elk-1505-112.

[6] K. K. Chakravarthi, P. Neelakantan, L. Shyamala, and V. Vaidehi, "Reliable budget aware workflow scheduling strategy on multi-cloud environment," Cluster Comput, vol. 25, no. 2, pp. 1189–1205, Apr. 2022, doi: 10.1007/S10586-021-03464-4.

[7] H. Li, X. Li, J. Xu, and L. Chen, "Entropy based swarm intelligent searching for scheduling deadline constrained workflows in hybrid cloud," International Journal of Machine Learning and Cybernetics, vol. 15, no. 4, pp. 1183–1199, Apr. 2024, doi: 10.1007/s13042-023-01962-y.

[8] H. Li, G. Xu, D. Wang, M. Zhou, Y. Yuan, and A. Alabdulwahab, "Chaotic-Nondominated-Sorting Owl Search Algorithm for Energy-Aware Multi-Workflow Scheduling in Hybrid Clouds," IEEE Transactions on Sustainable Computing, vol. 7, no. 3, pp. 595–608, Jul. 2022, doi: 10.1109/TSUSC.2022.3144357.

[9] J. Kakkottakath Valappil Thekkepuryil, D. P. Suseelan, and P. M. Keerikkattil, "An effective meta-heuristic based multi-objective hybrid optimization method for workflow scheduling in cloud computing environment," Cluster Comput, vol. 24, no. 3, pp. 2367–2384, 2021, doi: 10.1007/s10586-021-03269-5.

[10] S. Saeedi, R. Khorsand, S. Ghandi Bidgoli, and M. Ramezanpour, "Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing," Comput Ind Eng, vol. 147, p. 106649, Sep. 2020, doi: 10.1016/j.cie.2020.106649.

[11] M. Alaei, R. Khorsand, and M. Ramezanpour, "An adaptive fault detector strategy for scientific workflow scheduling based on improved differential evolution algorithm in cloud," Appl Soft Comput, vol. 99, p. 106895, Feb. 2021, doi: 10.1016/J.ASOC.2020.106895.

[12] A. Mohammadzadeh, M. Masdari, F. S. Gharehchopogh, and A. Jafarian, "A hybrid multi-objective metaheuristic optimization algorithm for scientific workflow scheduling," Cluster Comput, vol. 24, no. 2, pp. 1479–1503, Jun. 2021, doi: 10.1007/S10586-020-03205-Z.

[13] G. Yao, Y. Ding, Y. Jin, and K. Hao, "Endocrine-based coevolutionary multi-swarm for multi-objective workflow scheduling in a cloud system," Soft Computing - A Fusion of Foundations, Methodologies and Applications, vol. 21, no. 15, pp. 4309–4322, Aug. 2017, doi: 10.1007/S00500-016-2063-8.

[14] A. Mohammadzadeh, M. Masdari, F. S. Gharehchopogh, and A. Jafarian, "Improved chaotic binary grey wolf optimization algorithm for workflow scheduling in green cloud computing," Evol Intell, vol. 14, no. 4, pp. 1997–2025, Dec. 2021, doi: 10.1007/s12065-020-00479-5.

[15] H. A. Hassan, S. A. Salem, and E. M. Saad, "A smart energy and reliability aware scheduling algorithm for workflow execution in DVFS-enabled cloud environment," Future Generation Computer Systems, vol. 112, pp. 431–448, Nov. 2020, doi: 10.1016/j.future.2020.05.040.

[16] A. Belgacem and K. Beghdad-Bey, "Multi-objective workflow scheduling in cloud computing: trade-off between makespan and cost," Cluster Comput, vol. 25, no. 1, pp. 579–595, Feb. 2022, doi: 10.1007/s10586-021-03432-y.

[17] A. Taghinezhad-Niar, S. Pashazadeh, and J. Taheri, "Energy-efficient workflow scheduling with budget-deadline constraints for cloud," Computing, vol. 104, no. 3, pp. 601–625, Mar. 2022, doi: 10.1007/s00607-021-01030-9.

[18] M. I. Khaleel, M. Safran, S. Alfarhood, and M. Zhu, "Energy-latency trade-off analysis for scientific workflow in cloud environments: The role of processor utilization ratio and mean grey wolf optimizer," Engineering Science and Technology, an International Journal, vol. 50, p. 101611, Feb. 2024, doi: 10.1016/j.jestch.2023.101611.

[19] M. Zeedan, G. Attiya, and N. El-Fishawy, "Enhanced hybrid multi-objective workflow scheduling approach based artificial bee colony in cloud computing," Computing, 2022, doi: 10.1007/S00607-022-01116-Y/FULLTEXT.HTML.

[20] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," Advances in Engineering Software, vol. 69, pp. 46–61, 2014, doi: 10.1016/j.advengsoft.2013.12.007.