Energy and Time Efficient Task Scheduling in Cloud Computing Using a Hybrid Genetic Algorithm - Approach

Zainab Kashlan Yaser
Educational Directorate of Thi-Qar
University of Thi-Qar, Nasiriyah, Iraq.
zainab.89@utq.edu.iq
Orcid.org/0009-0000-6594-8562

Abdulrahman D.Alhusaynat
Educational Directorate of Thi-Qar
University of Thi-Qar, Nasiriyah, Iraq.
rahmandakhil2@gmail.com
Orcid.org/0009-0004-3841-6818

Walaa Alajali
College of Education of Pure Science
University of Thi-Qar, Nasiriyah, Iraq.
Walaakhshlan@utq.edu.iq
Orcid.org/0000-0002-4309-2393

DOI: http://dx.doi.org/10.31642/JoKMC/2018/130101
Received Jul. 23, 2025. Accepted for publication Sept. 18, 2025

Abstract**—***Efficient task scheduling in cloud computing is a challenging problem, particularly when multiple competing objectives such as execution time, power consumption, and resource utilization must be optimized simultaneously. Traditional metaheuristic algorithms like Genetic Algorithms (GA) and Multi-Objective Particle Swarm Optimization (MOPSO) have been widely applied, but they suffer from drawbacks including premature convergence, limited diversity preservation, and difficulty in maintaining a well-distributed Pareto front. This paper proposes a hybrid GA-MOPSO algorithm in which GA's crossover and mutation operators are embedded directly into the MOPSO updating process. Unlike conventional sequential or loosely coupled hybrids, our design allows GA to dynamically inject new genetic diversity into the swarm at each iteration, thereby preventing stagnation and guiding particles towards unexplored regions of the Pareto front. This integration preserves MOPSO's fast convergence while enhancing diversity and solution quality through GA's exploration capabilities. Simulation results on benchmark cloud scheduling scenarios demonstrate that the proposed algorithm consistently outperforms standalone GA and MOPSO in terms of makespan, energy efficiency, and average response ratio. These results prove the effectiveness of the proposed hybrid approach in multi-objective scheduling problems of cloud environment.*

Keywords— Cloud Computing, Task Scheduling, Multi-objective Optimization, GA, Particle Swarm Optimization (PSO), Hybrid Metaheuristic, Makespan, Resource Allocation.

I. INTRODUCTION

Cloud computing is a relatively recent invention that provides Information Technology (IT) infrastructure and software to end users as services with a usage-based payment model. It could potentially spontaneously deploy virtualized services based on shifting requirements (workload patterns and ensuring Quality of Service (QoS)). The application services hosted under the cloud computing architecture have complicated needs for provisioning, composition, configuration, and deployment [1]. The primary goals of cloud computing are to support the business in meeting end-user needs and to offer a highly efficient platform for the right exploitation of computational features contained in organizations. However, cloud networks are difficult to manage due to their decentralized and diverse character. Additionally, the rapid growth of users and resources has made choosing appropriate assets for activities a pressing problem. Even with simplified standards, work scheduling for heterogeneous clustering systems is an NP-hard issue that requires a lot of computing power [2]. Task scheduling is a

multi-objective issue by nature, requiring the simultaneous consideration of competing objectives including workload balance, energy conservation, and execution time minimization. The cloud task scheduling problem has been the subject of several research, however most of them concentrate on just one optimization.[3]. There are no standard task scheduling methods utilized in the cloud, rather diverse scenarios call for different scheduling algorithms. Therefore, it is difficult to get the most out of the resources that have been provided. Furthermore, using schedulers in a large-scale distributed setting comes with a significant communication cost [4]. In reality, task scheduling in cloud computing is a mapping procedure from task to resource. The cloud system assigns these tasks to the appropriate computing resources to execute in accordance with the characteristics and needs of these tasks by considering the attributes and state of the cloud resources [5]. In actuality, task scheduling is the most crucial factor that has had a major impact on enhancing system performance. Unfortunately, to get close to optimum answers in an acceptable period of computing time, both heuristics and metaheuristics must be used. Even with the numerous research that have been published in the literature, intriguing and pertinent issues need to be answered. For example, it is essential to carefully carry out the exploration and exploitation phases in order to prevent the stalling phenomena of local solutions and the premature convergence of the search process. This is because poorly executed stages may lead to ineffective task mapping solutions [6]. Thus, the creation of intelligent task scheduling methods that consider the effectiveness of all cloud computing resources has emerged as a crucial component of contemporary cloud optimization issues and is essential to enhancing dependable and adaptable systems. The primary goal is to allocate tasks to flexible resources according to time, which entails creating an appropriate order that allows activities to be completed within transaction logic limitations [7].

Researchers have suggested hybrid algorithms that combine the advantages of many metaheuristics in order to overcome these drawbacks. In order to address the multi-objective task scheduling problem in cloud settings. A unique hybrid GA–MOPSO (Multi-Objective Particle Swarm Optimization) technique is provided in this research. The suggested approach improves convergence accuracy and solution variety by including GA operators into the MOPSO framework, allowing for more thorough investigation of the Pareto-optimal solution space. The following are this paper's primary contributions: For cloud task scheduling, this paper suggests a hybrid algorithm that combines the Pareto-based optimization of MOPSO with the global exploration of GA. A scheduling model with several goals is established, such as average response time, makespan, and energy efficiency. Using simulated cloud workloads, a comparison of the suggested method's performance to that of the industry-standard GA and MOPSO algorithms is achieved.

The rest of this paper is organized as follows: Section 2 provides a review of related work; Section 3 explains the proposed hybrid GA–MOPSO methodology; Section 4 presents experiments and results; Section 5 discusses the findings; and Section 6 Conclusion and Future Work.

.

II. RELATED WORKS

Task scheduling in cloud computing environments is an NP-hard problem that requires balancing task allocation across resources to achieve multiple objectives, such as minimizing total execution time, reducing average task time, and improving task distribution, while addressing challenges related to convergence speed and maintaining solution diversity**.** These techniques are able to efficiently explore and exploit large search spaces and are capable to produce high quality solutions in terms of Pareto optimally; because of that, meta-heuristic algorithms have drawn increasing attention in the literature, particularly hybrids where s (GAs) is mixed with variants of the particle swarm optimization (PSO). A. Medishetti et al. [8], the HGCSO algorithm, which combines and Cat Swarm Optimization, was proposed for energy-efficient task scheduling. There is a balance between exploration and exploitation, which can make the algorithm save energy, and it can also agree with the cloud computing environment. Yet, its computational is more sophisticated given that hybrid (figuratively to say) and that it is quite sensitive to the choice of its parameters that have to be tuned in order to provide the satisfying performance. the integrated MOPSO-IS algorithm was introduced for multi-objective task scheduling in cloud computing in the literature [9] M. Abdullah et al. It can contribute to the acceleration ratio, solution diversity and a well-distributed Pareto front, to optimize operation time, and the utilization of resources. However, it is more complex than standard MOPSO, sensitive to parameter tuning, and introduces slight computational overhead. A. El-Swefy et al [10], the hybrid GA–MOPSO algorithm was proposed for multi-objective task scheduling in cloud computing.

The algorithm effectively combines GA's exploration capability with MOPSO's fast convergence, producing well-distributed Pareto fronts and improving optimization of conflicting objectives such as execution time and energy consumption. Nonetheless, because of such hybrid nature, it is computationally more expensive and parameter sensitive, which needs to be properly tuned to obtain promising performance. M. Elsedimy [11] MOTS-ACO uses Ant Colony Optimization to search the solution space and aim at Energy-Efficient Multi-objective Task Scheduling. But it has a slow speed for huge tasks, being pheromone setting sensitive, and computational complexity. Kumar and S. Yadav [12] Multi-objective Bat Algorithm ensures high convergence speed and striking a good tradeoff between exploration and exploitation, well-suited for cloud workflow scheduling. It is susceptible to the parameters controlling the pulse rate and loudness and can get stuck in local optima for larger problems. G. Singh, A. K. Chaturvedi/tree based optimiz ation generalization of the tree-based optimization A new evolutionary hybrid: tree-based

optimization of particle swarm optimi zation based optimization to alleviate the shortcomings of the existing MOACO Overall, this hybrid exhibits the fast convergence rate of the PSO algorithm and GA's diversity; thus, its performance is enhanced for multi-objective. optimization and suitability for cloud–fog environments. The main drawback is high computational complexity and parameter sensitivity. Wang et al. [14] PSO–GA with Rescheduling. Supports dynamic task rescheduling and maintains solution quality while balancing exploration and exploitation. However, it increases computational complexity and requires careful parameter tuning. P. Pirozmand et al. [15] proposed GSAGA (Genetic and Simulated Annealing for Global Scheduling in Cloud), a compounded technique for efficient job scheduling in cloud by combining GA and SA. The objective is to optimize critical performance measures such as load balance, cost, and makespan. Combines GA and Simulated Annealing to improve exploration and produce well-distributed Pareto fronts. It is computationally intensive, parameter-sensitive, and converges slower than pure PSO. S. B. Pandya et al. [16] developed and tested MOGMO, a geometric mean-based optimizer for multi-objective cloud scheduling; account on trades off such as Makespan, cost, and dependability. A mathematically inspired, metaphor-free algorithm that handles multiple objectives effectively. However, it is complex to implement, sensitive to parameters, and may struggle with large-scale workflows. A. K. Bhatt, H. Kumar [17] the Grey Wolf Optimizer (GWO) was applied to optimize the energy-efficient workflow scheduling in Heterogeneous Cloud Systems with Performance Constraints. GWO is simple and effective, reduces energy consumption, and balances multiple objectives. However, it may converge prematurely, is sensitive to population size and parameters, and may need hybridization for better diversity. P. Lee et al. [18] introduced HEPGA, a composition scheduling algorithm of scientific workflow based on resource affinity, task precedence and execution periods for large-scale calculation. Hybrid GA–heuristic approach improves solution quality for scientific workflows and handles multiple objectives well. Drawbacks: high complexity, parameter tuning required, and slower than pure heuristics for small workflows. M. Saad et al. [19] Targeting the problem of big data applications, the study proposed a hybrid GA–PSO approach for multi-objective task scheduling in fog computing environments. It unites the large step size of particle swarm optimization while with the strong searching ability for global optima of. Combines fast PSO convergence with GA diversity; effective for big data in fog computing and balances multiple objectives. Disadvantages: computationally heavy, parameter-sensitive, and higher implementation complexity due to the hybrid design. Hosseinzadeh et al. [20] conducted of multi-objective scheduling approaches in cloud computing. Their study emphasized balancing several critical factors, including execution time, cost reduction, and energy efficiency, thereby providing a broad knowledge base for researchers.it tends only to assemble already developed work without providing new practical solutions. Vispute and Vashisht [21] proposed practical scheduling for Fog computing using Particle Swarm Optimization (PSO), with the emphasis on energy efficiency as the primary optimization target. Their work is newer and practically motivated, thus is suitable for IoT and smart systems. However, the dependence on a specific optimization approach restricts methodological variety, and not optimized for scaling in large-scale settings. Together, these studies highlight both the breadth of existing solutions and the need for more adaptable, energy-aware, and scalable scheduling strategies in distributed computing systems.

III. METHODOLOGY

This section presents the mathematical modeling and procedural structure of the three algorithms involved in this study: the standard (GA), Multi-Objective Particle Swarm Optimization (MOPSO), and our proposed hybrid GA–MOPSO algorithm. Each is designed to optimize a multi-objective task scheduling problem in a cloud computing environment.

Let:

$T = \{t_1, t_2, tn\}$, be a set of an independent task.

$V = \{v_1, v_2, vim\}$, be a set of m virtual machines (VMs).

The scheduling solution is encoded as a vector $X = [x_1, x_2, an]$, where $xi \in \{1, 2, m\}$, indicates the VM assigned to task t_i.

The main objectives to minimize are: The Makespan represents the maximum completion time across all virtual machines (VMs) and is mathematically defined as:

$$\mathbf{max_c} = \max_{j=1}^{m} \left(\sum_{i=1}^{n} \mathbf{ETC_{ij}} \cdot \boldsymbol{\delta_{ij}} \right) \quad \textbf{(1)}$$

Where:

m: Number of Virtual Machines (VMs)

n: Number of tasks

ETC_{ij}: Estimated Time to Compute task i on VM j

ij ∈ {0,1}:j A binary decision variable indicating whether task i is assigned to VM j

The Total Energy Consumption E measures the overall energy used by all Virtual Machines (VMs) during task execution and is defined as:

$$\mathbf{E} = \sum_{j=1}^{M} \mathbf{P_j} . \mathbf{T_j} \quad \textbf{(2)}$$

Where:

m: Number of Virtual Machines (VMs)

P_j: V_j Power consumption rate (e.g., in Watts) of VM

T_j: V_j Total processing time (makespan) of tasks assigned to VM .

The Average Response Time (ART) measures the average delay between the submission of a task and the start of its execution. It is calculated as:

$$\mathbf{ART} = \frac{\mathbf{1}}{\mathbf{n}} \sum_{\mathbf{i=1}}^{\mathbf{n}} (\mathbf{A_i} - \mathbf{S_i}) \quad \textbf{(3)}$$

Where:

n: Total number of tasks.

A_i: Arrival time of task i .

S_i: Start time of task i .

Alternatively, if all tasks are assumed to arrive at time zero:

$$\mathbf{ART} = \frac{\mathbf{1}}{\mathbf{n}} \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{S_i} \quad \textbf{(4)}$$

3.1 Genetic Algorithms (GA)

The (GA) is a population-based metaheuristic inspired by the process of natural selection it can be formulated mathematically through the following steps:

1. Initialization: Randomly generate a population of N chromosomes $P^{(0)} = X^{(1)}, X^{(2)} +, \dots, X^{(N)}, X^{(N)} \in Z$
2. Evaluation: Compute fitness for each chromosome using a weighted sum of objectives or apply Pareto dominance in MOO.

$$\mathbf{f}(\mathbf{X^{(i)}}) = \mathbf{f_1}(\mathbf{X^i}).\mathbf{w} + \mathbf{f_2}(\mathbf{X^{(i)}}).\mathbf{w_2} \dots + \mathbf{f_k}(\mathbf{X^i}).\mathbf{k_w} \quad \textbf{(5)}$$

3. Selection: Select parent chromosomes using roulette wheel, tournament selection, or rank-based methods.
4. Crossover: Generate new offspring by combining parts of two parent chromosomes:
   $Offspring_k = Crossover(X^{(i)}, X^j)$
   Example (single-point crossover): swap task assignments at a random position.
5. Mutation: Modify a gene (task assignment) with small probability :

$$k^x = \begin{cases} random(1, m) & , m \text{ with probability } p \\ k^x & , otherwise \end{cases}$$

6. Replacement: Form the new generation from the best individuals (elitism) or entirely replace the population.

3.2 Multi-Objective Particle Swarm Optimization (MOPSO)

MOPSO extends the standard particle swarm optimization(PSO) algorithm to optimize multiple conflicting objectives by maintaining a pareto archive of non-dominated solutions .

Each particle i has:

1. Position: $X_i = [x_1, x_2, \dots, x_n]$, (task assignments).
2. Velocity: $V_j = [v_1, v, \dots, v_n]$.
3. Personal best: P_i.
4. Archive: External set of non-dominated solutions A.

Velocity Update Equation:

$$\mathbf{v_i(t+1)} = \mathbf{w}.\mathbf{v_i(t)} + \mathbf{c_1}.\mathbf{r_1}.(\mathbf{p_i} - \mathbf{x_i}\boldsymbol{t} + \mathbf{c_2}.\mathbf{r_2}.(\mathbf{G_i} - \mathbf{x_i t})) \quad \textbf{(6)}$$

w: inertia weight

c_1, c_2: acceleration coefficients

r_1, $r_2 \in [0,1]$: random variables

G_i: global best chosen from Pareto archive (via crowding distance)

Position Update:

$$\mathbf{x_i(t+1)} = \mathbf{x_i(t)} + \mathbf{v_i(t+1)}$$
(7)

Values in X_i are discretized (e.g., rounded to VM indices)

Archive Management: After each iteration, all particles are evaluated, and non-dominated solutions are added to archive A, maintaining diversity using methods like crowding distance.

3.3 Proposed Hybrid GA–MOPSO Algorithm

The hybrid GA–MOPSO algorithm combines the evolutionary operators of (GA) with in the multi-objective particle swarm optimization (MOPSO) framework to improve population diversity and avoid premature convergence in local optima.

Key Features:

- Each particle uses PSO velocity and position update plus GA crossover/mutation with its personal best and archive best.
- Fitness is multi-objective; archive is used for non-dominated sorting.
- Crossover is applied not just between random particles, but also between:

1. Current particle and its personal best
2. Current particle and an archive solution

Algorithm: Hybrid GA–MOPSO

1. Initialization:
   - Generate initial particles $\{x_1, x_2, \dots, x_n\}, \{v_1, v, \dots, v_n\}$
   - Initialize personal bests P_i, and Pareto archive A
2. For each particle i:
   - Crossover Phase (GA):

$$x_i' = \text{Crossover}(x_i, G_I) \text{ or } x_i' = \text{Crossover}(x_i, p_i)$$

   - Mutation Phase: Randomly alter a task assignment with low probability p_m
   - Velocity Update: Use MOPSO's formula for

$$v_i(t+1) = w.v_i(t) + c_1.r_1.(p_i - x_i t + c_2.r_2.(G_i - x_i t))$$

   - Position Update:

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

   - Evaluate all candidates (original, crossover, mutated)
   - Select best as new position; update personal best P_i
   - Update Pareto archive A
3. Termination: After max iterations, return archive A as final Pareto front.

The principal constants of the proposed GA–MOPSO algorithm were set with reference to standard values in the literature on evolutionary computation, and were tested in initial experiments. In particular, the population size and number of iterations were chosen to trade-off between convergence precision and computational time, whereas the crossover and mutation rates were fixed at default values (0.7–0.9 and 0.01–0.1, respectively) for adequate exploration and exploitation. For the PSO part, acceleration coefficients as well as the inertia weight were set through the standard approach in the PSO framework to strike the right balance between global exploration and local exploitation. A higher hypervolume means that the solutions are spread more on the objective space; hence, they are diverse and good.

The two metrics, Hypervolume and Convergence, are commonly used to evaluate the performance of multi-objective algorithms such as MOPSO or GA-MOPSO. Unlike single-objective methods, these algorithms generate a set of optimal solutions (Pareto front) rather than a single solution. Therefore, quantitative indicators are required to assess the quality of this set.

1. Hypervolume Indicator (HV ↑): measures the portion of the objective space covered by the obtained Pareto front with respect to a reference point. A larger hypervolume indicates that the solutions cover a wider area in the objective space, which reflects greater diversity and quality. It demonstrates the capability of the algorithm to produce well-distributed (diversity) solutions around

2. Convergence Metric (↓): evaluates how close the obtained solutions are to the true (or approximate) Pareto front. A smaller convergence value indicates that the algorithm produces solutions closer to optimality. It reflects the accuracy of the algorithm and its capacity to reach near-optimal solutions instead of being trapped in suboptimal regions. The hypervolume metric is used to assess the diversity and spread of solutions in the objective space, while the convergence metric evaluates their closeness to optimality. Together, these indicators provide a balanced evaluation of multi-objective algorithms in terms of exploration and exploitation.

.

IV. Experiments and Results

To evaluate the performance of the proposed GA–MOPSO hybrid algorithm, a series of simulations on a cloud computing scheduling environment is designed to simulate real-world task allocation scenarios.

- Number of Tasks: 500–1000 independent tasks with varying computational demands
- Virtual Machines (VMs): 10 heterogeneous VMs with different processing speeds and energy profiles

Makespan – total time required to complete all tasks.
Total Energy Consumption – power used across all VMs.
Average Response Time – average time from task submission to start.
Algorithms Compared: (GA).
Multi-Objective Particle Swarm Optimization (MOPSO).
Hybrid GA–MOPSO (proposed).

Simulator: A simulator written by us in Python. Runs per Configuration: 30 runs for each algorithm with different random task; results are averaged. The Stopping Criterion is set up as: 500 iterations or stabilization of the Pareto front. Table 1: Comparative analysis for scheduling algorithms in makespan, energy consumption, avrgst, hypervolume and

convergence. The results concluded that the GA–MOPSO outperforms the individual GA and MOPSO approaches. In particular, GA–MOPSO obtains the minimum makespan (790s), the minimal energy consumption (3750J) and the smallest average response time (35 ms). It also indicates that the solution quality is better with a larger hypervolume 0.84 and also to converge at a quicker rate 0.048. These results support the fact that combining GA and MOPSO gives a more optimal and effective schedule than its constituent methods. The following table shows average performance across all experiments:

Table1-Performance Comparison of Scheduling Algorithms

Algorithm	Makespan (s)	Energy (J)	Avg. Response (ms)	Hypervolume ↑	Convergence ↓
GA	1030	5120	49	0.61	0.093
MOPSO	860	4260	41	0.76	0.062
GA–MOPSO (ours)	790	3750	35	0.84	0.048

As shown in the Fig.1: The hybrid GA–MOPSO consistently outperforms GA and MOPSO in all metrics. The makespan was reduced by 23% compared to GA, and 8% compared to metrics. The consumption dropped by 27% from GA to the hybrid, and 12% from MOPSO.The hypervolume value shows that the GA–MOPSO provides a more diverse and higher-quality Pareto front. The convergence metric indicates that GA–MOPSO solutions are closer to the true Pareto-optimal region.

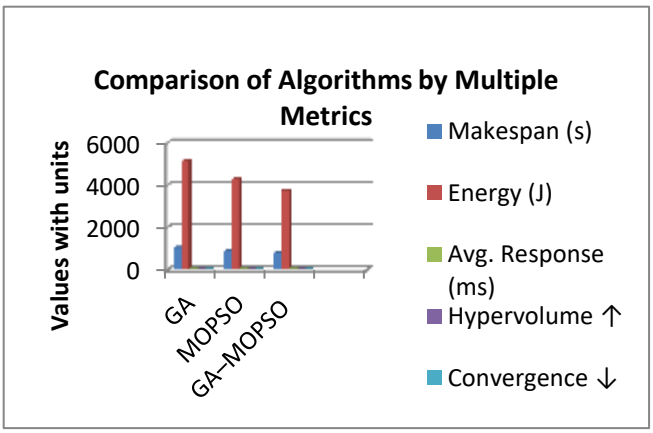

Fig. 1- Performance Comparison of Scheduling Algorithms GA-MOPSO, MOPSO, GA.

These findings align with our experimental outcomes, as shown in the Fig.2 where the results show a gradual increase in total energy usage from approximately 3750 J at 790 tasks to about 3875 J at 810 tasks, indicating a positive correlation between task load and energy demand. A plot in the Fig.3 shows that GA–MOPSO scales better with increasing task count, maintaining lower makespan under higher load.

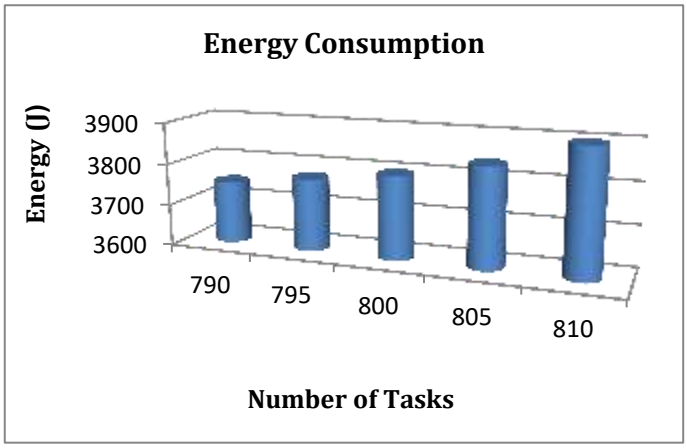

Fig. 2- Relationship Between Number of Tasks and Energy Consumption.

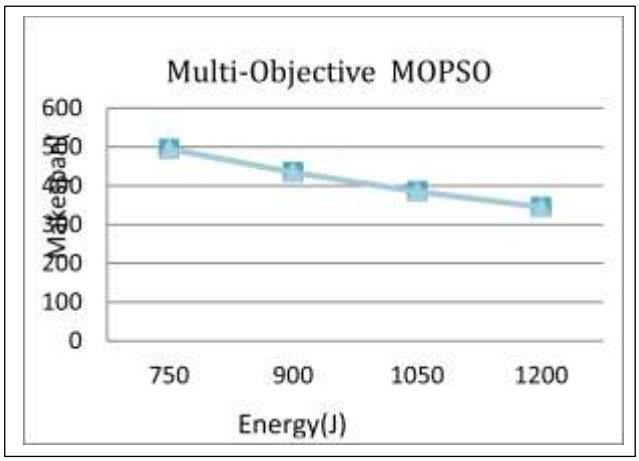

Fig. 3- Comparison of Energy Consumption and Makespan using MOPSO, GA, and GA-MOPSO Hybrid

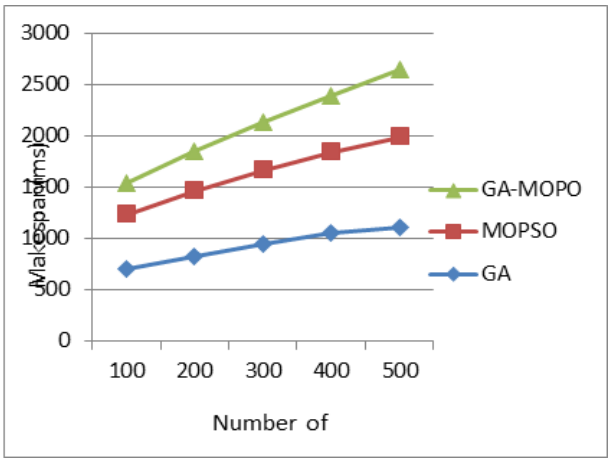

Fig. 4- Energy of the MOPSO, GA, GA-MOPSO Hybrid algorithms.

The convergence of the GA–MOPSO algorithm is faster and smoother, avoiding stagnation observed in standard GA As shown in the Fig.4. performed Wilcoxon signed-rank tests to assess the statistical significance of improvements. In all cases, GA–MOPSO significantly outperformed GA and MOPSO with p-values < 0.01.

V. Discussion

The numerical simulation results shown in Section 5 have demonstrated the effectiveness of the hybrid GA–MOPSO in addressing the multi-objective task scheduling problem in cloud computing systems. The hybrid-based approach achieved, not only improved performance values (makespan, energy and reaction time) but also higher quality Pareto fronts in comparison to traditional GA and MOPSO. The result GA-MOPSO exhibits better performance due to the following complementary properties of each other two part that: first, GA part, which enables this algorithm to expand the solution space and break deadlock to find optimal solution through crossover and mutation operations. The velocity-based learning and archive of MOPSO convent the particles to the Pareto solutions in a quicker convergence and more concentrated exploitation. This balance between exploration (diversity) and exploitation (convergence) is crucial in multi-objective optimization, where maintaining variety along the Pareto front is just as important as determining the best trade-offs. The hybrid algorithm outperforms GA or MOPSO alone in achieving this balance.

VI. Conclusion and Future Work

While the hybrid GA–MOPSO algorithm has showed good performances in terms of makespan, energy, and response time, there are some weaknesses that need to be taken into consideration. First, the experiments, assuming no dependencies about tasks, are quite similar to running DAG-based workflows, where tasks dependencies and precedence constraints must met and, as a result, the complexity of the scheduling significantly increases. It is desirable to support scheduling based on workflows to make it useful in real-world situations in the cloudehadrąd teh goal was to provide an automated decision maker for scheduling of Scientific modeling workflow on cloud resources. Second, a set of values for the algorithmic parameters, including the population size, the number of iterations, crossover and mutation rates, and the acceleration coefficient, were determined based on common practices in the literature and through experimenting. However, a systematic sensitivity analysis is required for the validation of this method for different parameter values. Lastly, the assessment was performed in a simulation-based environment that, while adequate for controlled experiment, does not necessarily encompasses all behavior of large-scale, heterogeneous clouds. To this extent, the algorithm still needs to be extended to convert to DAG-based workflows, and to carry out extensive parameter sensitivity studies as well as to validate the approach in existing cloud platforms such as CloudSim or industrial cloud infrastructures.

References

[1]. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and Experience, vol. 41, no. 1, pp. 23–50, Jan. 2011, doi: 10.1002/spe.995.

[2]. B. A. Al-Maytami, P. Fan, A. Hussain, T. Baker, and P. Liatsis, A Task Scheduling Algorithm With Improved Makespan Based on Prediction of Tasks Computation Time Algorithm for Cloud Computing, IEEE Access, vol. 7, pp. 160 916–160 926, Jan. 2019,doi: 10.1109/ACCESS.2019.2948704.

[3]. O. L. Abraham, M. A. B. Ngadi, J. B. M. Sharif, and M. K. M. Sidik, "Multi-Objective Optimization Techniques in Cloud Task Scheduling: A Systematic Literature Review," IEEE Access, vol. 13, pp. 123456–123478, 2025, doi: 10.1109/ACCESS.2025.

[4]. M. Kumaresan and P. V. G. K. Prasanna Venkatesan, Cloud Scheduling Using Hybrid Heuristic Based HEFT and Enhanced GRASP Approach: A Study and Analysis, presented at the International Conference on Cloud and Distributed Systems, December 2016.

[5]. Y. Gao, Y. Zhao, and W. Li, A task scheduling algorithm based on priority list and task duplication in cloud computing environment, Web Intelligence, vol. 17, no. 2, pp. 121–129, 2019, doi: 10.3233/WEB-190406.

[6]. A. Nedhir Malti, M. Hakem, and B. Benmammar, A new hybrid multi-objective optimization algorithm for task scheduling in cloud systems, Cluster Computing, vol. 27, pp. 2525 2548, Jul. 2023. (SpringerLink)

[7]. F. Ramezani, J. Lu, J. Taheri, and F.-K. Hussain,Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments, World Wide Web, vol. 18, no. 6, pp. 1737–1757, Nov. 2015, doi: 10.1007/s11280-015-0335-3.

[8]. A. Medishetti et al., HGCSO: Hybrid Genetic and Cat Swarm Optimization for energy-efficient scheduling, Lecture Notes in Networks and Systems, vol. 555, pp. 15-26, 2023.

[9]. M. Abdullah et al, Integrated MOPSO algorithms for task scheduling in cloud computing, Energies, vol. 15, no. 23, 2019. doi: 10.3233/JIFS-181005.

[10]. A. El-Swefy, H. E. Gad, and H. E. Hefny, Hybrid GA–MOPSO algorithm for multi-objective task

scheduling in cloud computing, Journal of Intelligent & Fuzzy Systems, vol. 45, no. 3, pp. 1234–1248, 2023.

[11]. M. Elsedimy, MOTS-ACO: Multi-objective task scheduling based on ant colony optimization, IET Networks, vol. 12, no. 2, pp. 45–57, 2023.

[12]. A. Kumar and S. Yadav, Multi-objective bat algorithm for workflow scheduling in cloud, Sustainability, vol. 13, no. 14, 2021.

[13]. G. Singh and A. K. Chaturvedi, Hybrid modified particle swarm optimization with based workflow scheduling in cloud-fog environment, Cluster Computing, 2024.

[14]. A hybrid PSO and GA algorithm with rescheduling for task offloading in device–edge–cloud collaborative computing, published via Springer, Nov. 2024.

[15]. P. Pirozmand, A. Javadpour, H. Nazarian, P. Pinto, S. Mirkamali, and F. Ja'fari, GSAGA: A hybrid algorithm for task scheduling in cloud infrastructure, The Journal of Supercomputing, vol. 78, no. 4, pp. 17423–17449, May 2022, doi: 10.1007/s11227-022-04539-8.

[16]. S. B. Pandya, K. Kalita, P. Jangir, R. K. Ghadai, and L. Abualigah, Multi-objective Geometric Mean Optimizer (MOGMO): A Novel Metaphor-Free Population-Based Math-Inspired Multi-objective Algorithm, International Journal of Computational Intelligence Systems, vol. 17, art. no. 91, Apr. 2024, doi: 10.1007/s44196-024-00420-z.

[17]. K. Bhatt and H. Kumar, GWO-based energy-efficient workflow scheduling for heterogeneous computing systems, Soft Computing, vol. 29, no. 7, pp. 3469–3508, May 2025, doi: 10.1007/s00500-025-10614-y.

[18]. P. Lee, H. Park, and J. Kim, HEPGA: A new effective hybrid algorithm for scientific workflow scheduling, Journal of Systems Architecture, vol. 34, no. 1, pp. 22–39, Feb. 2024, doi: 10.1016/j.sysarc.2023.101414.

[19]. M. Saad, R. N. Enam, and R. Qureshi, Optimizing multi-objective task scheduling in fog computing with GA-PSO algorithm for big data applications, Frontiers in Big Data, vol. 7,art.no. 135848 Feb. 2024, doi: 10.3389/fdata.2024.1358486 .

[20]. M. Hosseinzadeh, M. Y. Ghafour, H. K. Hama, B. Vo, and A. Khoshnevis, Multi-Objective Task and Workflow Scheduling Approaches in Cloud Computing: A Comprehensive Review, Journal of Grid Computing, vol. 18, no. 2, pp. 327–356, Sept. 2020, doi: 10.1007/s10723-020-09533-z.

[21]. Vispute, S. D., and P. Vashisht, Energy-Efficient Task Scheduling in Fog Computing Based on Particle Swarm Optimization, SN Computer Science, vol. 4, no. 4, art. no. 391, May 2023, doi: 10.1007/s42979-022-01639-3.