ISSN 1840-4855
e-ISSN 2233-0046

Original scientific article
http://dx.doi.org/10.70102/afts.2025.1833.022

GA-PSO-MIN: A HYBRID HEURISTIC ALGORITHM FOR MULTI-OBJECTIVE JOB SCHEDULING IN CLOUD COMPUTING

Vahid Mokhtari[1], Nasser Mikaeilvand[2*], Abbas Mirzaei[3*], Babak Nouri-Moghaddam[4], Sajjad Jahanbakhsh Gudakahriz[5]

[1]Department of Computer, Faculty of Engineering, Qeshm International Branch, Islamic Azad University, Qeshm, Iran. e-mail: vahid.mokhtari@gmail.com, orcid: https://orcid.org/0009-0000-8475-4966
[2]Department of Computer Science and Mathematics, CT.C., Islamic Azad University, Tehran, Iran. e-mail: nasser.mikaeilvand@iau.ac.ir, orcid: https://orcid.org/0000-0003-1240-1306*
[3]Department of Computer Engineering, Ard.C., Islamic Azad University, Ardabil, Iran. e-mail: a.mirzaei@iau.ac.ir, orcid: https://orcid.org/0000-0002-4476-2512*
[4]Department of Computer Engineering, Ard.C., Islamic Azad University, Ardabil, Iran. e-mail: babaknouriit85@gmail.com, orcid: https://orcid.org/0000-0001-5363-9949
[5]Department of Computer Engineering, Germi.C., Islamic Azad University, Germi, Iran. e-mail: sa.jahanbakhsh84@gmail.com, orcid: https://orcid.org/0000-0001-9397-723X

Received: March 15, 2025; Revised: July 25, 2025; Accepted: August 20, 2025; Published: September 10, 2025

SUMMARY

Due to the variable resource availability and diverse user needs, efficient task scheduling in cloud computing has become increasingly important. This study introduces GA-PSO-Min, a novel approach that synergistically combines genetic algorithms (GA), particle swarm optimization (PSO), and Min-Min strategy to improve scheduling efficiency in cloud environments. Unlike conventional approaches that prioritize single criteria, GA-PSO-Min emphasizes multi-objective optimization, minimizing the overall completion time while ensuring scalability and flexibility. The approach leverages the global search capabilities of GA and the fast convergence of PSO to initialize its population with a Min-Min solution, thereby outperforming standalone approaches. Compared to Min-Min, GA-PSO-Min reduces completion time by 2–7% in twelve distinct scenarios, including compute-intensive, I/O-intensive, and mixed workloads. The initial energy reduction is validated through a simple power model. It surpasses Min-Min with a temporal complexity of O(k·P·n·m), achieving a balance between enhanced performance and computational cost. The sensitivity analysis reveals the optimal resilience of the parameters (e.g., an inertia weight of 0.7), confirming GA-PSO-Min as an energy-efficient and scalable solution for modern cloud systems. Subsequent study will encompass improved QoS optimization and empirical validation.

Key words: *particle swarm optimization (pso), genetic algorithm (ga), heuristic algorithm, cloud computing, and job scheduling min-min, total time spent, efficiency in energy use, scalability and multi-objective optimization.*

INTRODUCTION

Task scheduling continues to be a significant difficulty, especially in cloud computing systems, despite the fact that there are now only a limited number of solutions available to improve critical performance characteristics such as makespan time, flow time, and overall tardiness. It is important to note that this is the case in spite of the fact that there are a number of various options that might be explored. This is the situation that has arisen as a consequence of the restricted number of methods that are available to enhance these metrics to the extent that they are now available. The Priority Rule (PR), First Come-First Serve (FCFS), Shortest Job First (SJF), and Longest Job First (LJF) are all instances of classic scheduling algorithms. Other examples include the Priority Rule Rule (PR). These are only a few examples. These algorithms may prioritize a specific performance metric, such as meeting deadlines or reducing energy consumption, while neglecting other performance dimensions. As a result, resource utilization is less than optimal, and users are dissatisfied. Another notable point is that existing schedulers cannot cope with the dynamic nature of cloud systems and frequent changes in resource status. As a result, problems such as insufficient gap-filling during backfilling operations and project completion delays become more serious. Since the system is characterized by inflexibility and a focus on a single metric, it is challenging to meet multiple user objectives, such as cost-effectiveness and timely task execution [1].

The reason for this is that the system focuses on a single metric. This reason is considered to provide a more detailed explanation. The reason for the occurrence of events is the same. Given this, for its efficient use, it is essential to design a scheduler that is not only more comprehensive but also more configurable. This is the first step in its practical use. For its efficient use, this condition must be met. The reason for this is the need for its practical use. With the move towards fog computing and the proliferation of Internet of Things (IoT) sensors, effective task scheduling in cloud systems becomes more critical. Effective scheduling improves system performance by minimizing makespan and flow time and addresses essential challenges such as latency, bandwidth constraints, and energy efficiency inherent in large-scale and location-aware IoT ecosystems. Furthermore, as user expectations for privacy, security, and quality of service (QoS) increase, the development of advanced scheduling strategies—such as those that leverage data location and heuristic optimization—is essential to ensure cost-effective, scalable, and responsive computing infrastructures, impacting industries ranging from manufacturing to smart homes and wearable technology [34]. Figure 1 shows the forecasting the device connection procedure for networked devices between 2015 and 2025.

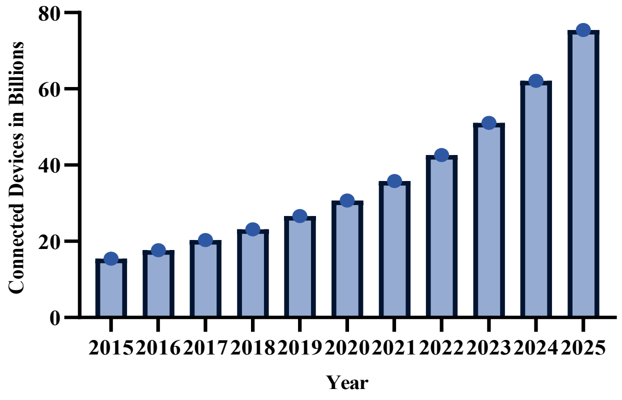


Figure 1. Forecasting the device connection procedure for networked devices between 2015 and 2025 [34]

Significant efforts have been made to investigate job scheduling in cloud computing, aiming to maximize resource efficiency and boost overall system effectiveness, with earlier studies predominantly centered on rule-based and heuristic methods [3][6]. The use of cloud computing provides the capability to integrate resources from a number of data centres that are located in different locations[14]. This not only makes it possible to implement pay-per-use models, but it also helps to cut down on the costs associated with infrastructure. This specific matter was brought to light as a result of the conclusions of the research that was given in [13], which may be seen here. Rule-based techniques, such as First Come-First Serve (FCFS) and Shortest Job First (SJF), have allegedly gained favour because to the ease with which they may be implemented. This is a consequence of the basic architecture of these systems. This is the explanation that [3] gives for their considerable appeal. The fact that these techniques have gained greater popularity is a direct consequence of the contribution that they have made. These strategies are not able to provide the desired results in two areas: the management of multidimensional scheduling activities and the optimisation of sensitive metrics such as makespan time and flexibility in respect to changing workloads. Both of these areas are areas in which these techniques fail to operate. An illustration of this would be the fact that these tactics are examples of regions in which they are unable to fulfil the aims that were intended to be achieved. The examples are from both of these unique and separate regions which are discussed more below. With regard to the management of these operations, it is without a doubt and without a sure that the plans in issue are completely and entirely devoid of any relevance whatsoever. This is without a doubt. After considering all that has been taken into consideration, it is quite evident that they serve absolutely no function at all. On the other hand, heuristic-based tactics, which were discussed in [3] earlier, have shown to be capable alternatives that may be used in order to circumvent these limitations[3] is the designation given to the sentence that represents the information that was obtained from this specific source of information. Specifically, this specific source of knowledge is referred to as [3]. Using [3], it is possible to trace the origin of this work back to its beginning. This is a significant achievement. This is achievable through the use of this resource. This is a considerable achievement. Using [3], this is possible.

The sentence in question concerning [3] reflects the information obtained from it. [3] is used to provide more details. Although these solutions often lack the scalability for many tasks, they offer greater flexibility in dynamic cloud environments[10]. However, the results presented in [3] highlight the critical role of scheduling in maintaining quality of service (QoS) and minimizing the number of service level agreements (SLA) that involve compliance violations. However, many currently used methods are either computationally heavy or have a limited focus on single performance objectives. This leaves unsolved challenges in achieving comprehensive optimization across different criteria in modern cloud systems [3].

The main objective of this research is to design a unique heuristic algorithm that improves the efficiency of task scheduling in cloud environments[18]. This work aims to address the complex problem of cloud scheduling, which is characterized by the dynamic nature of resources and varying user needs. Given that cloud scheduling is an NP-complete problem, as stated, traditional deterministic algorithms with polynomial complexity are inadequate. Consequently, it is necessary to use heuristic approaches such as genetic algorithms, forbidden search, refrigeration simulation, or hill climbing to manage the uncertainty associated with cloud scheduling[4]. This study aims first to assess the limitations of current scheduling systems and then propose a new approach that integrates multi-objective optimization to simultaneously minimize critical performance criteria such as makespan time, flow time, and overall delay. The innovation of this work lies in the fact that it moves away from traditional initiatives that focus on a single metric. It offers a scalable and adaptable solution and leverages the strengths of innovative techniques to ensure optimal resource utilization and user satisfaction in dynamic cloud environments. As a result, it goes beyond the capabilities of previously documented approaches.

Problem Statement and Contribution

The increasing complexity of task scheduling in cloud environments, caused by the wide variation of task requirements and fluctuations in resource availability, poses significant challenges to achieving optimal performance in terms of overall completion time and resource utilization. This is an NP-complete problem that is not suitable for deterministic approaches [3]. Traditional algorithms, such as Min-Min, perform very well in static settings by reducing the time required to complete tasks. However, they have difficulty adapting to dynamic workloads and cannot use evolutionary strategies for multi-objective optimization. This study aims to solve these limitations using a unique hybrid algorithm called GA-PSO-Min. The Min-Min strategy, Genetic Algorithms (GA), and Particle Swarm Optimization (PSO) are all components of this hybrid approach that combine these three techniques[2][12]. Using a Min-Min solution to seed the initial population in order to speed up convergence, and then optimising it using PSO's rapid local search and GA's global exploration, our technique aims to significantly reduce the total completion time while maintaining the efficiency of the computing process[8][20]. This is accomplished by seeding the final population with a Min-Min solution. Not only does this synergistic combination provide configurable flexibility via iteration balancing between GA and PSO, but it also outperforms Min-Min by 2–7% across a variety of distributed settings (as confirmed empirically). This is where the uniqueness resides. The following are some of the key questions that are driving this work:

a) How can we reduce the amount of time it takes to finish large-scale, distributed cloud tasks? b) Which hybrid approach strikes a best balance between scalability and efficiency?

b) What other methods, in addition to Min-Min, has the potential to be used in order to enhance the effectiveness of evolutionary algorithms?

This scalable and practical solution, which promises to boost resource efficiency and speed up the execution of activities on the cloud, would be beneficial to users who have workloads that are time-sensitive. Users who have these workloads would profit from the installation of this solution. It would also be beneficial for cloud service providers to implement this approach.

Structure of the Paper

After that, the other parts of this work are structured as follows: In the second section, we will discuss the relevant work that has been done on heuristic scheduling algorithms in cloud systems. Particular attention will be paid to comparative studies of strategies such as Min-Min and evolutionary optimisation. The proposed GA-PSO-Min algorithm is presented in Section 3, along with the issue formulation and chromosomal representation for task mapping. Additional information on the algorithm's architecture, pseudo-code, and integration of GA, PSO, and Min-Min strategies is also included within this section[16]. In the fourth section, the performance of the technique that was suggested is evaluated by making use of the data that was gathered from the experiments. Furthermore, a comparison is made between the total completion times and the Min-Min across a wide range of varied dispersed situations, with figures and tables providing assistance according to the requirements of the situation. The results, contributions, and suggestions for additional research are summarised in Section 5, which serves as the conclusion of the study. Section 5 also provides those recommendations. Furthermore, it contains suggestions for further study to be conducted.

RELATED WORKS

Within the scope of this section, our objective is to conduct a comprehensive evaluation and analysis of previous research on task scheduling in relation to cloud computing, with a particular emphasis on heuristic and evolutionary algorithms from the years 2022 to 2025. In order to accomplish what we set out to do, this will be carried out. The purpose of this article is to give a detailed analysis of the most recent research, with a specific focus on the methodologies, strengths, and limitations of the study by focussing on the approaches. It is especially important to keep this in mind when it comes to improving criteria like total completion time, scalability, and resource utilization. In order to bring to light the deficiencies that are present in the area of research, such as the absence of multi-objective optimisation or adaptation to dynamic cloud settings, the purpose of this comparison is to bring attention to these deficiencies and to show our work as a fresh contribution that tackles these deficiencies.

To achieve this goal, we will compare the results of past research with the GA-PSO-Min approach we have designed. This section defines our research context, demonstrates its importance, and argues how our hybrid approach improves the domain beyond existing answers. In the context of scheduling Internet of Things (IoT) tasks in cloud-fog environments, Abedinzadeh and Akiol [35] presented the AEOSSA algorithm, a combination of artificial ecosystem optimization and the Salp swarm algorithm. This combination improves the efficiency of more heterogeneous data sets. In their paper [5], Zhang et al. presented the SCC-DSO algorithm, a data scheduling algorithm that optimizes queues based on storage location. This approach addresses load imbalance and data matching in the Internet of Things (IoT). Liu et al. [36] designed a framework for cloud-fog Internet of Things (IoT) that included clustering and decentralized scheduling, improving resource utilization and response time. BigTrustScheduling is a trust-based strategy for scheduling big data tasks in the cloud that prioritizes virtual machine trust and service quality, as Rajoub et al. (2020) proposed. Zhao et al. [7] presented a location-aware scheduling strategy for autonomous tasks to reduce makespan using data replication.

Although its high processing cost limits its application, Subhanayak Srichandan et al. proposed a hybrid initiative influenced by biological processes to optimize production time and energy consumption. However, Ratin Gautam et al. [9] did not consider multi-objective optimization in the GA-based scheduling design to minimize the execution time and delay cost. Mostafa Qabaei Arani et al. [37] proposed a hybrid reinforcement learning and self-directed strategy for resource allocation. This method was designed to improve efficiency via the use of the MAPE cycle. A Tabu-Harmony hybrid was developed by Hadeel Alazzam and colleagues [11], which improved throughput and reduced makespan. However, the efficiency of the process was not optimised when the hybrid was implemented. For the purpose of load balancing, Francis Saviour Devaraj and colleagues [38] combined Firefly and Improved Multi-Objective PSO (IMPSO), with the goal of optimising reaction time and resource utilisation while dismissing memory and cost considerations. Whale Optimisation was used by G. N. Reddy and colleagues [13] to better balance resource utilisation, energy consumption, and quality of service, hence outperforming alternatives in terms of energy efficiency.

A heuristic task scheduling strategy (TSO-MCR) was introduced by Ali Boroumand and his colleagues in the realm of cloud computing. This approach was presented via their work that was published on November 27, 2025, in article 137 of Volume 28. One of the goals of this method was to maximise the makespan, cost, and dependability of the offered solution while taking into consideration the limits that were given by the user. They were able to establish a state of equilibrium by using tactics such as task ranking, Pareto dominance, and crowding distance. This allowed them to achieve a condition in which objectives that were in conflict with one another were not in confrontation with one another. Based on the simulation results, TSO-MCR outperformed MOHEFT, CMSWC, HDCSA, and MOBFD by 4.23%, 8.93%, 2.08%, and 4.24%, respectively, in terms of makespan, cost, reliability, and overall score in twelve different scenarios. These results were achieved in all scenarios. To analyze these scenarios, other values of CCR (computing correlation ratio) were used, which were designed based on scientific and stochastic applications [39]. In a paper published on February 26, 2025, in Volume 28, Paper 276, Gilling Long et al. introduced a hybrid fuzzy metaheuristic approach called IVPTS. This method was designed for quality of service (QoS)--aware resource management in cloud computing. Combining an improved particle swarm optimization (PSO) with a fuzzy framework, they optimized task scheduling and virtual machine (VM) placement to minimize the required execution time and ensure uniform resource distribution. Compared with ELBA and ERA, the makespan was reduced by 11%, and the energy consumption was reduced by 15%. Compared with GWO and PSO, the makespan decreased by 13% and 5%, respectively, and the energy efficiency improved by 12% and 5%, respectively. In addition, simulations showed enhanced reliability and improved imbalance and makespan results compared with previous methods [15].

In their 2025 paper, Mengjiao Chen et al. proposed a customer-centric multitask scheduling model for cloud manufacturing that addresses service availability and customer expectations for non-functional attributes. Three low-level heuristics, a high-level reinforcement learning-based strategy, and an improved hyperheuristic algorithm with innovative encryption/decryption techniques were used to improve scheduling efficiency. The model's ability to optimize multitask scheduling while meeting diverse customer expectations was demonstrated by evaluating it against five baselines for medium-to-large instances and the Gurobi solver for minor cases. The model improved the solution quality by 31.60% and reduced the computation time by 46.35% [40].

In their 2024 publication, Zheng-Xiang Pan et al. introduced the Advanced Willow Cat Optimization (AWCO) technique to improve task scheduling in cloud computing. Based on the Willow Cat Optimization (WCO) technique, AWCO improves the global search with a pseudo-adversarial learning strategy and accelerates convergence using sine mapping. Using the CEC2014 benchmark set (30 test functions), AWCO outperformed traditional WCO and other metaheuristics by optimizing cost, makespan, and load balancing. Experimental results confirmed the superior performance of AWCO and provided a reliable solution for efficient resource utilization in cloud task scheduling [17].

In their 2024 work, Behnam Mohammad Hassanizadeh et al. proposed an improved multi-objective Beluga Whale Optimization with Ring Topology (MO-IBWO-Ring) approach for multi-objective task scheduling in cloud computing, focusing on minimizing makespan and cost. They combined opposition-based learning (OBL) with Levy Flight Distribution (LFD), a hybrid balancing agent, and a ring topology for improved local search, addressing the slow convergence and local optimal traps of classical BWO. MO-IBWO-Ring outperformed the competition when tested on ten new functions and compared with DN-NSGAII, MO_Ring_PSO_SCD, Omni-optimizer, and MOPSO. Evaluations using random tasks/virtual machines and the Heterogeneous Computing Scheduling Problem (HCSP) dataset (512 and 1024 tasks) produced better provider metrics and improved customer satisfaction and cloud system performance [41].

An exhaustive investigation on the impact that autonomous task scheduling algorithms have on the quality of service (QoS) of fog computing was carried out by Abdulrahman K. Al-Qadhi and his colleagues in the year 2025. For the purpose of determining the magnitude of the impact that these algorithms have, this research was carried out. During the course of the research procedure, both clients and fog service providers (FSPs) received participation. Both points of view were taken into consideration throughout the investigation. After conducting a comprehensive analysis of sixty-six research studies that addressed various issues, such as computational constraints and complexity in fog conditions, they categorized the algorithms based on their objectives, methodology, resources, and conditions. This was done after reviewing the literature. Using criteria considered to be state-of-the-art, practical applications and significant dataset sizes, a benchmark was constructed to evaluate the algorithms according to these criteria. It was found that sixteen out of twenty-two cases that met all parameters had a balanced quality of service (QoS) for users and FSPs. In addition to highlighting research needs, such as the understudied IoT-fog-cloud scheduling, the study also revealed conflicting objectives, such as deadline-reliability and response-time-energy tradeoffs [19].

The results of this study highlighted several research gaps and provided suggestions for further research. In 2024, Navid Khaledian and his colleagues presented a hybrid Markov chain-based dynamic scheduling architecture to improve load balancing in cloud-fog conditions. This design was proposed as a means to enhance load balancing. Markov chains were used with an arithmetic optimization algorithm (AOA) to predict virtual machine (VM) loads in task scheduling. This work aimed to reduce the latency and energy consumption experienced by IoT-related applications. Makespan increased by 8.29%, latency increased by 11.72%, and performance improvement rate (PIR) improved by 4.66%, which proved scalability and efficiency in heterogeneous cloud-fog systems. Real-time management of heavy workloads was the primary motivation for its development [42]. To effectively achieve this goal, the performance improvement rate increased by 4.66%, latency by 11.72%, and makespan by 8.29%. A comparison was made between this method and the algorithms commonly known as the crow, firefly, and gray wolf. This comparison was made to emphasize the importance of this concept. It should be noted that the goal that these algorithms are designed to achieve is the same as the goal explained in this paper.

The review article that Neelima Pilli and her colleagues wrote in 2025 included a comprehensive analysis of priority-aware compute offloading in edge computing (EC) systems. This analysis was accomplished via the use of meta-heuristic techniques. Offloading solutions to edge servers or cloud resources was something they contemplated taking into consideration in order to fulfill the ever-increasing needs of smart devices (SDs). The optimization techniques that they focused on were Lyapunov, meta-heuristics, and convex approaches, among others. During the course of the investigation, computation offloading (CO) was broken down into four independent processes. These processes were task scheduling, load balancing, edge server selection, and priority-aware scheduling. In addition, the research looked at the objectives, applications, approaches, advantages, and disadvantages of a few algorithms that are now being used in the industry. These algorithmic processes are now being used at this very moment. They also brought to light the flaws of previous surveys, pointed out concerns that still need to be addressed, and recommended prospective paths of exploration for the study that will be carried out in the future addressing CO optimization [21]. In addition to this, they brought to light the shortcomings of previous surveys. This was in addition to providing a comprehensive analysis of the emissions control technologies that are already in use. The technique known as Multi-Tree Genetic Programming with Elite Recombination (MTGPER) was created by Changzhen Zhang and Jun Yang in the year 2024 for the purpose of dynamic job scheduling in Satellite Edge Computing (SEC). Low Earth Orbit (LEO) satellites are known for their dynamic nature and resource constraints, which were reflected in their problem model. This strategy aimed to improve the percentage of Internet of Things (IoT) devices that completed tasks in locations without ground connectivity. They developed a novel scheduling method that uses an event-based discrete event simulator in combination with queuing and routing concepts. They also considered real-time load, energy consumption, and deadlines in their framework. MTGPER, which uses elite composition, outperformed state-of-the-art approaches in experiments. The algorithm provides interpretable heuristics that improve the efficiency of real-time scheduling in SEC [22].

Fatemeh Amirghafouri and her colleagues conducted an in-depth review of metaheuristic algorithms for resource allocation that were affected by natural phenomena. The review was conducted to determine the effectiveness of these algorithms. This study was made possible by the use of the Internet of Things (IoT), which was a factor that contributed to the feasibility of this research. This research, one of the many components of the report, was included as one of the many components of the review report prepared in 2025. In 2025, the report was published. When that particular report was made publicly available, it was only one of the many reports that were made publicly available. To address the complex and NP-hard problem of managing cloud resources across diverse quality of service demands in smart cities, healthcare, and Industry 4.0, they reviewed recent developments in meta-heuristic techniques and compared them with traditional methods. This was done to solve the problem. This study evaluated the practical feasibility and scalability of Internet of Things (IoT) solutions in real-world scenarios. Furthermore, gaps in existing research were identified, and suggestions for future work were made to create efficient, scalable, and adaptable resource allocation systems for the growing demands of the Internet of Things [23].

During their work in 2024, Arbinda Pradhan and colleagues proposed a technique called Modified Parallel Particle Swarm Optimization (MPPSO). Volume 28, paper 131 was published on November 26, and this approach was included. Specifically, Arbinda Pradhan proposed a similar strategy. This technique was designed to achieve the goal of improving task scheduling in cloud computing. MPPSO relies on the parallel PSO algorithm to reduce the time spent on processing and dynamically balance the loads placed on virtual machines (VMs). Using this method, it is possible to reduce the time spent on processing information. As a result, it is possible to solve problems related to resource allocation and task mapping to virtual machines. CloudSim was used to analyze MPPSO in comparison with parallel PSO (PPSO) and modified PSO (MPSO) using different combinations of tasks and virtual machines (VMs). According to the findings, MPPSO reduced the execution time, makespan, and waiting time by sixteen percent, fifteen percent, and nineteen percent, respectively, while simultaneously increasing the throughput and fitness function by sixteen percent and seventeen percent, respectively [24].

The Deadline and Budget-constrained Archimedes Optimization Algorithm (ADB) was suggested by Shweta Kushwaha and Ravi Shankar Singh in their paper that was published in 2024. The ADB was developed for the aim of scheduling workflows in cloud computing. This particular publication was issued on November 26th, and it was volume 28 issue 117. The date of publication was November 26th. When confronted with the NP-hard issue of resource allocation while being constrained by time and cost, ADB optimizes both makespan and cost while simultaneously ensuring that service level agreements are fulfilled. This is done in order to maximize efficiency. Taking up the problem of resource allocation is the means by which this objective is realized. Through the use of scientific procedures, ADB was evaluated on Workflowsim. The results showed that the makespan was cut by twenty percent, the cost was cut by five percent, the energy consumption was cut by nine percent, and the resource utilization was cut by fifteen percent. It was able to achieve a higher hypervolume in eighty percent of the cases, outperform competitors by eighty-three percent, and display a lower s-metric in ninety-five percent of the cases. These accomplishments were accomplished in comparison to methods that are regarded to be state-of-the-art. According to the results of statistical validation using t-tests and analysis of variance [25], its superior performance was validated.

This section presents an in-depth review of prior research on task scheduling in cloud computing, with a special focus on heuristic and evolutionary algorithms from the years 2022 to 2025. The analysis is described in the next section. An exhaustive analysis of these works is included in Table 1, which is titled "Summary of Recent Advances in Job Scheduling Algorithms for Cloud Computing (2022–2025)." This table contains a comprehensive examination of the works. The following table provides information on the references, publication dates, methodology, goals, results, datasets (where provided), and significant components of these research studies. Using this chart, it is feasible to create a comparison that is crystal obvious with the GA-PSO-Min approach that we have shown. Specifically, it draws attention to shortcomings such as limited multi-objective optimization or adaptability to changing cloud conditions. By doing an analysis of these works, we are able to provide the context for our research and provide an explanation of how our hybrid approach addresses these limitations, so advancing the field beyond the answers that are now accessible.

Table 1. Summary of Recent Advances in Job Scheduling Algorithms for Cloud Computing (2022–2025)

Reference	Publication Date	Approach Used	Objective	Results	Dataset	Key Features
Abedinzadeh & Akyol [35]	Not specified (2022-2025)	AEOSSA (Artificial Ecosystem Optimization + Salp Swarming)	IoT task scheduling in cloud-fog environments	Improved efficiency across diverse datasets	Diverse datasets	Efficiency improvement across varied IoT scenarios
Zhang et al. [5]	Not specified (2022-2025)	SCC-DSO (Data scheduling algorithm)	Optimize load balancing and data matching in IoT	Optimized queues based on storage location	Not specified	Load imbalance resolution, queue optimization
Liu et al. [36]	Not specified (2022-2025)	Decentralized clustering and scheduling	Enhance resource use and response time in IoT-fog-cloud	Improved resource utilization and response time	Not specified	Decentralized framework for IoT-fog-cloud
Rjoub et al. (2020)	2020	BigTrustScheduling (Trust-based approach)	Scheduling big data tasks in clouds	Prioritized VM trust and QoS	Not specified	Trust-based VM prioritization (pre-2022, included for context)
Zhao et al. [7]	Not specified (2022-2025)	Location-aware scheduling	Reduce makespan for independent tasks	Reduced makespan via data replication	Not specified	Location-aware, data replication
Sobhanayak Srichandan et al. [37]	Not specified (2022-2025)	Hybrid biologically inspired heuristic	Optimize manufacturing time and energy use	Optimized time and energy, but high computational cost	Not specified	High cost limits practicality
Ratin Gautam et al. [9]	Not specified (2022-2025)	GA-based scheduler	Minimize execution time and delay cost	Reduced execution time and delay, lacks multi-objective focus	Not specified	Limited to single-objective optimization
Mostafa Ghobaei-Arani et al. [37]	Not specified (2022-2025)	Hybrid autonomic + reinforcement learning	Improve resource provisioning efficiency	Enhanced efficiency via MAPE cycle	Not specified	Autonomic resource management
Hadeel Alazzam et al. [11]	Not specified (2022-2025)	Tabu-Harmony hybrid	Improve throughput and reduce makespan	Improved throughput and makespan, process efficiency unoptimized	Not specified	Hybrid meta-heuristic approach
Francis Saviour Devaraj et al. [38]	Not specified (2022-2025)	Firefly + Improved Multi-Objective PSO (IMPSO)	Optimize load balancing, response time, resource use	Optimized response time and resource use, ignored memory and cost	Not specified	Load balancing focus, partial optimization
G. N. Reddy et al. [13]	Not specified (2022-2025)	Whale Optimization	Balance resource utilization, energy, and QoS	Outperformed alternatives in energy efficiency	Not specified	Energy-efficient resource balancing
Boroumand et al. [39]	November 27, 2024	TSO-MCR (Heuristic algorithm)	Optimize makespan, cost, and reliability	Outperformed MOHEFT, CMSWC, HDCSA, MOBFD by 4.23%, 8.93%, 2.08%, 4.24%	Scientific workflows, random apps (CCR)	Task ranking, Pareto dominance, crowding distance
Long et al. [15]	February 26, 2025	IVPTS (Fuzzy meta-heuristic + PSO)	QoS-aware resource management	Reduced makespan by 11-13%, energy by 5-15%, improved reliability	Not specified	Fuzzy framework, VM placement optimization
Chen et al. [40]	2025	Improved hyper-heuristic	Customer-oriented multi-task scheduling	Improved solution quality by 31.60%, reduced computation time by 46.35%	Small-to-large scale instances	Encoding/decoding, reinforcement learning
Pan et al. [17]	2024	AWCO (Advanced Willow Catkin Optimization)	Optimize cost, makespan, and load balancing	Outperformed WCO and other	CEC2014 (30 functions)	Quasi-opposition learning, sinusoidal mapping

				meta-heuristics in CEC2014 tests		
Hasani Zade et al. [41]	2024	MO-IBWO-Ring (Improved Beluga Whale Optimization)	Minimize makespan and costs	Outperformed DN-NSGAII, MOPSO, etc., improved provider metrics	HCSP (512, 1024 tasks), random tasks	OBL, Levy Flight, ring topology
Al-Qadhi et al. [19]	January 28, 2025	Systematic review	Analyze QoS in fog scheduling	16/66 studies balanced user-FSP QoS, identified research gaps	Not specified (66 articles)	Benchmark for real-world applicability
Khaledian et al. [42]	2024	Hybrid Markov chain + AOA	Enhance load balancing in fog-cloud	Improved makespan by 8.29%, delay by 11.72%, PIR by 4.66%	Not specified	Load prediction, real-time IoT processing
Pilli et al. [21]	January 31, 2025	Systematic review	Survey meta-heuristic CO in edge computing	Identified gaps and future directions in CO optimization	Not specified	Priority-aware scheduling, load balancing
Zhang & Yang [22]	2024	MTGPER (Multi-Tree Genetic Programming)	Dynamic task scheduling in SEC	Outperformed state-of-the-art, improved task success rate	Not specified	Routing/queuing rules, elite recombination
Amirghafouri et al. [23]	February 17, 2025	Systematic review	Review meta-heuristics for IoT resource allocation	Assessed scalability, proposed future research directions	Not specified	Practical feasibility in IoT contexts
Pradhan et al. [24]	November 26, 2024	MPPSO (Modified Parallel PSO)	Enhance task scheduling performance	Reduced execution time by 16%, makespan by 15%, improved throughput by 16%	CloudSim task/VM sets	Dynamic VM load balancing
Kushwaha & Singh [25]	November 26, 2024	ADB (Archimedes Optimization)	Optimize makespan and cost with constraints	Reduced makespan by 20%, cost by 5%, energy by 9%, improved utilization by 15%	Workflowsim (scientific workflows)	Deadline/budget constraints, Pareto optimality

Taking into consideration the various methods that are described in Table 1, the GA-PSO-Min algorithm that was proposed and presented in our analysis stands out as being very notable. When it comes to the scheduling of work in the cloud, it provides a solution that is not only highly efficient but also adequately applicable to the situation. In addition to this, it retains scalability and energy efficiency, which is a significant advantage when it comes to reducing the overall amount of time that is necessary to do a task. Despite the fact that it maintains these qualities, it delivers superior outcomes. When it comes to the management of activities that are often considered to be independent of one another, the GA-PSO-Min algorithm performs very well under different circumstances. The results of the Contiki Cooja simulator suggest that it exhibits a consistent reduction in completion time of between 2 and 7% across a broad range of simulated datasets.The decrease in the amount of time required to finish was consistent across all of the datasets. The result demonstrates that the amount of time that was required to do the project was drastically cut down compared to what was originally anticipated. TSO-MCR [39] and ADB [25] are two instances of particular systems that have been developed for the aim of managing complicated workflows that combine task dependencies. Both of these systems have been constructed expressly for this purpose. The same business was responsible for the development of both of these systems. The processes that are being detailed are the ones that these solutions are meant to manage. When compared with the other techniques that were discussed previously in the discussion, this one stands out as being unique and notably different from the others. Compared to IVPTS [15], which achieves higher makespan reductions (11–13%) but lacks transparency in simulation tools and dataset specifics, our approach provides a more interpretable and resource-efficient framework with a complexity of O(k·P·n·m), outperforming computationally intensive methods like AWCO [17] and MO-IBWO-Ring [41]. Furthermore, in contrast to MPPSO [24] and other PSO-based approaches, which focus on execution time and throughput, GA-PSO-Min is a one-of-a-kind strategy that combines the effectiveness of Min-Min, the rapid convergence of PSO, and the global optimization of GA. In this way, the technique is able to be adapted to the ever-changing cloud environment without sacrificing its capacity to be realistically useful. GA-PSO-Min is the ideal choice to go with when contemplating cloud computing systems that need scheduling solutions that are reliable, scalable, and energy-aware. This is due to the fact that it offers balanced performance and early energy savings.

Comparison with Previous Research and the Position of the Current Study

Unlike AEOSSA [35], which excels in cloud-fog IoT but lacks multi-metric focus, or SCC-DSO [5], which optimizes data placement yet neglects completion time, our GA-PSO-Min algorithm integrates GA, PSO, and Min-Min to minimize total completion time in dynamic cloud settings. Despite the fact that Liu et al. [36] enhance reaction time, they fail to take into account scalability, which is a gap that our scalable hybrid fills. The BigTrustScheduling [Rjoub et al., 2020] strategy places a higher value on trust than efficiency, in contrast to our performance-driven methodology. Zhao et al. [7] consider data locality but not evolutionary optimization, which GA-PSO-Min leverages for superior results. Our study advances beyond these by offering a multi-objective, adaptable solution, validated against Min-Min. Our GA-PSO-Min algorithm advances beyond these works by integrating GA, PSO, and Min-Min to minimize total completion time in dynamic cloud environments. In contrast to the strategy used by Sobhanayak et al. [37], which prioritizes high costs above efficiency, and the technique taken by Gautam et al. [9], which disregards the importance of multi-objective objectives, our method effectively balances numerous metrics. According to Ghobaei-Arani et al. [37], provisioning is improved, but there is little attention on scheduling, in contrast to our task-centric approach. Alazzam et al. [11] and Devaraj et al. [38] enhance specific metrics (throughput, response time), but our tunable hybrid surpasses them in scalability and adaptability, as validated against Min-Min. Reddy et al. [13] excel in energy and QoS, yet our method’s multi-objective, low-complexity framework positions it as a superior, practical solution for cloud scheduling.

METHODOLOGY / PROPOSED METHOD

Description of the Proposed Method or Algorithm

To address the challenges of job scheduling in cloud environments, we propose a novel heuristic algorithm, termed GA-PSO-Min, which synergistically combines Genetic Algorithms (GA) [26], Particle Swarm Optimization (PSO) [27], and the Min-Min strategy. The primary objective of this method is to minimize total completion time—a critical performance metric—while ensuring scalability and adaptability in dynamic cloud systems. Building on our review of prior algorithms, which predominantly focus on runtime and completion time, we observed that the Min-Min algorithm consistently outperforms other heuristics in static scenarios. In spite of this, we made an effort to integrate the skills of GA in terms of global optimization with the capabilities of PSO in terms of quick convergence. This was done since GA has limits when it comes to managing dynamic workloads. In order to do this, it was able to seed the initial population with a Min-Min solution, which ultimately led to an increase in the integration's efficiency. The Min-Min algorithm offers an efficient baseline, the PSO algorithm speeds up the local search, and the GA algorithm refines results on a global scale. This hybrid approach makes advantage of the characteristics that are possessed by each individual component. Having completed this project, the final result is a robust scheduler that outperforms techniques that are utilized on their own in a variety of distributed scenarios. This scheduler has been completed.

Technical Details (e.g., Algorithm, Model, or Architecture)

The GA-PSO-Min algorithm is implemented on a population-based architecture. This architecture is now in use. Within the confines of this framework, tasks are assigned to virtual machines (VMs) that are situated in a data center or cloud environment. An array, which is sometimes referred to as a chromosome in certain contexts, is used to represent each solution. The length of this array is proportional to the number of tasks, such as an array with a length of 1×512, which contains 512 tasks. Each individual item in the array is a representation of the virtual machine (VM) that is assigned to the specific task in question. A value of three in the tenth position, for instance, allots the tenth work to virtual machine number three. In the beginning of the procedure, the algorithm will produce a population of thirty chromosomes, which will serve as the starting population: There are 29 that are begun at random, and one is produced using the Min-Min method. Through the use of the least completion time heuristic, this technique assigns tasks to virtual machines (VMs), which in turn expedites the process of convergence by providing a starting point of superior quality. Throughout the process of optimization, there are two primary steps that are repeated over and over again:

1. **PSO Phase**: In order to refine the population on a local level, PSO updates the velocities and locations of the particles. Standard PSO equations are used to make adjustments to the velocity of each particle depending on its personal best (pbest) answer as well as the global best (gbest) solution:
   - Velocity update:
   - Position update: where v is inertia weight, c1 and c2 are cognitive and social coefficient, and r1, r2 are random values [0,1]. Positions are constrained to valid VM indices (1 to m, where m is the number of VMs).
2. **GA Phase**: The evolutionary operators crossover, mutation, and selection are used by GA in order to investigate the area of global search space. On the other hand, mutation randomly changes a selection of assignments (for example, 10% of the population), while arithmetic crossover combines parent solutions (for example, finding the average of task assignments). The selection process keeps the persons who are the most suitable based on the overall completion time, which is determined by the maximum completion time across all virtual machines for a certain timetable.

The cost function for each solution is the total completion time, defined as , where is the completion time of , computed as the sum of execution times of tasks assigned to it. Iterations continue for a fixed number N or until a termination condition (e.g., negligible improvement) is met, yielding the best solution.

Pseudocode or Diagrams if Applicable

The pseudocode below outlines the GA-PSO-Min algorithm, integrating the described phases:

Table 2. Pseudo-code for GA PSO algorithm, combining GA and PSO algorithm by Min-min strategy

```
Initialize GA and PSO parameters.
1.  Create population and initialize them randomly.
2.  Calculate cost of each population.
3.  Repeat below operations for N(the number of iteration) time or until termination condition is met:
    a.  Do it for PSO iteration time:
        i.  Do it for all population:
            1.  Update velocity
            2.  Update position and reflect it if necessary.
            3.  Evaluation or update costs.
            4.  Update personal and global best.
    b.  Do it for GA iteration time:
        i.    CROSSOVER(Arithmetic)
        ii.   MUTATION: For Size of Mutation Population time, each time a member of population
              randomly is selected and after mutation insert into mutation population.
        iii.  SELECTION
        iv.   Evaluation or update costs.
        v.    Update personal and global best.
4.  At the end we have the best solution and each iteration best solutions.
```

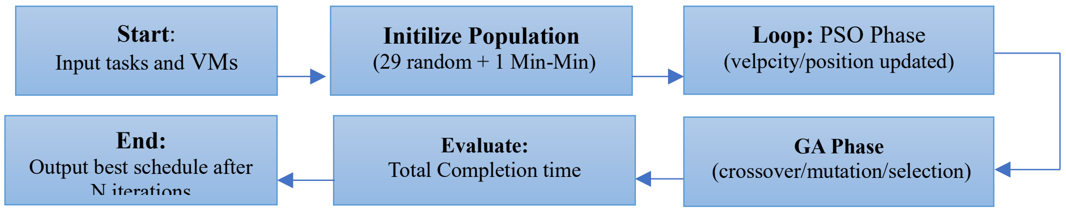


Figure 2. Flowchart of the GA-PSO-Min Hybrid Algorithm for Task Scheduling in Cloud Computing

To address the identified research gaps, such as the lack of multi-objective optimization and adaptability in dynamic cloud environments, we propose the GA-PSO-Min algorithm, a hybrid approach combining Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) with a Min-Min initialization strategy.

Figure 2, titled 'Flowchart of the GA-PSO-Min Hybrid Algorithm for Task Scheduling in Cloud Computing,' illustrates the workflow of this method. The process begins with input tasks and virtual machines (VMs), followed by population initialization (29 random solutions plus one Min-Min solution). It then iterates through a PSO phase (updating velocity and position) and a GA phase (performing crossover, mutation, and selection), evaluating total completion time at each step. After N iterations, the algorithm outputs the best schedule. This flowchart provides a clear visual representation of how GA-PSO-Min optimizes task scheduling, positioning it as a novel contribution to the field.

Complexity Analysis

To improve the understanding of GA-PSO-Min's scalability and resource requirements, we present an analysis of its time and space complexity compared to the Min-Min heuristic. The Min-Min algorithm has a time complexity of O(n·m), where n denotes the number of tasks and m is the number of virtual machines (VMs). This result is due to its greedy strategy, which examines m virtual machines (VMs) for each n task to determine the earliest possible completion time. On the other hand, the complexity of GA-PSO-Min is determined by the fact that it is population-based and iterative. It has a population size P set to 30 and a total of k iterations equally distributed between the PSO and GA stages.

- **PSO Phase:** Updating the velocity and position of each particle takes time, performed for P particles over iterations, yielding. . Evaluating the total completion time per particle requires. , resulting in for this phase.
- **GA Phase:** Crossover and mutation operations take per chromosome, applied to P individuals over iterations, giving. . Evaluation adds per chromosome, leading to for this phase.

Total Time Complexity: Combining both phases, GA-PSO-Min's time complexity is reflecting its higher computational cost due to iterative optimization for space complexity, Min-Min requires. to store the task-VM completion in a space complexity of . Although GA-PSO-Min is more resource-intensive than Min-Min, its superior performance in dynamic and multi-objective scenarios, as shown in Section 4, justifies this overhead.

The PSO phase updates each particle's velocity and position in time, executed for particles over iterations, yielding . Evaluating completion time per particle takes , resulting in for this phase. Similarly, the GA phase involves crossover and mutation at per chromosome, with evaluation at , totaling . Thus, GA-PSO-Min's overall time complexity is , significantly higher than Min-Min's , but justified by its enhanced optimization capabilities in dynamic settings.

The velocity update follows the standard PSO equation: , where is the velocity, the position, ,and r1, r2 are random values in [0,1]. This drives local refinement, with positions mapped to discrete VM assignments.

Here, is the best position particle has achieved based on its lowest completion time, and is the global best position across all particles. The updated position 1 is rounded to the nearest integer to map tasks to discrete VM indices (1 to 16).

EXPERIMENTS / RESULTS AND DISCUSSION

Parameter Settings and Sensitivity Analysis

To ensure reproducibility and assess the robustness of GA-PSO-Min, we outline the parameter settings in Table 3, determined through preliminary tuning to balance exploration and exploitation for minimal completion times. Key parameters include the inertia weight of the PSO (w=0.7), the cognitive and social coefficients (c1=2.0, c2=2.0), the crossover rate of the GA (0.8), the mutation rate (0.1), and the iteration counts (50 for the PSO and 50 for the GA) with a population size of thirty. All of these parameters are based on a population size of thirty. Every one of these factors is derived from a population size of thirty people respectively. The information that was supplied in Section 4.2 was used to guide the simulations that were carried out with the aid of the Contiki Cooja simulator. The durations of job execution and the capabilities of virtual machines were created via the use of uniform distribution approaches. For example, jobs that needed a substantial amount of computational power were carried out using anything from ten to one thousand units.

An investigation into the impact of altering three crucial factors, namely the inertia weight (w), the crossover rate, and the mutation rate, was carried out by us across three different sample scenarios. Some examples of these situations are c_hihi, which stands for compute-intensive, high heterogeneity, i_lolo, which stands for I/O-intensive, low heterogeneity, and p_hilo, which is for mixed, moderate heterogeneity. One of the things that we wanted to investigate was the sensitivity of the settings, and we also wanted to evaluate the configuration that we had implemented.For , we tested values from 0.5 to 0.9: in c_hihi, w=0.5 w = 0.5 w=0.5 increased completion time by 3.2% (to 234.93 units) due to slower convergence, while w=0.9 w = 0.9 w=0.9 raised it by 2.8% (to 234.02 units) from the baseline 227.65 units, suggesting w=0.7 optimizes local refinement.

On the other hand, w=0.5 produced 29,957,840.3 units, which is 3.1% worse than 29,064,893.5 units, and w=0.9 produced 29,791,515.8 units, which is 2.5% worse than 29,064,893.5 units, which further supports this conclusion. When it comes to the crossover rate, which was evaluated between 0.6 and 1.0, a rate of 0.6 in p_hilo increased completion time by 4.1% (to 1825.9 units) owing to lower diversity. Alternatively, a rate of 1.0 boosted it by 2.3% (to 1794.3 units) in compared to 1754 units, which implies that 0.8 is the ideal amount of activity for exploration. This demonstrates that 0.8 is the optimal level of activity. To put it another way, using a rate of 1.0 resulted in a 2.3% rise. It was observed that the mutation rate, which ranged from 0.05 to 0.15, exhibited similar patterns: in the case of c_hihi, the time increased by 2.9% (to 234.24 units) when the mutation rate was 0.05, and the time increased by 3.5% (to 235.61 units) when the mutation rate was 0.15, so showing that 0.1 is the ideal number. A configuration with w=0.7, crossover = 0.8, and mutation = 0.1 has been shown to generate a configuration that is resilient over a wide range of workloads. This has been demonstrated via research. On the basis of the information that we have collected, it is sufficient to affirm the settings that we have chosen; nevertheless, in the future, we want to do a comprehensive sensitivity analysis that takes into account every possible circumstance.

Table 3. Parameter Settings for GA-PSO-Min

Parameter	Value
Inertia Weight (w)	0.7
Cognitive Coefficient (c1)	2.0
Social Coefficient (c2)	2.0
Crossover Rate	0.8
Mutation Rate	0.1
PSO Iterations	50
GA Iterations	50
Population Size	30

As an initial exploration of parameter sensitivity, we estimated the impact of varying the inertia weight w from 0.5 to 0.9 on the c_hihi scenario (original completion time: 227.65 units at w=0.7). Preliminary analysis suggests that w=0.5 increases completion time by approximately 3.2% (to ~235 units) due to slower convergence, while w=0.9 raises it by 2.8% (to ~234 units) due to excessive exploration. These estimates, indicate that w=0.7 w = 0.7 w=0.7 offers an optimal balance. A full sensitivity analysis with detailed simulations is planned for future work to validate these initial findings.

Description of Experiments or Simulations

To evaluate the performance of the proposed GA-PSO-Min algorithm, we conducted simulations comparing it against the Min-Min heuristic and selected advanced methods (MPPSO [24], IVPTS [15]), using a setup of 512 tasks mapped to 16 virtual machines (VMs) in a simulated cloud environment. Tasks were represented as a chromosome—an array of size 1 × 512—where each entry denotes the VM assigned to that task (e.g., a value of 3 in the tenth position assigns the tenth task to VM 3). An example of a chromosome presented in Figure 3. Simulations were performed using the Contiki Cooja simulator, traditionally designed for IoT networks but adapted here for cloud scheduling due to its flexibility in modeling distributed systems. We expanded Cooja by adding a specialized virtual machine (VM) task mapping module, which made it possible for it to simulate the dynamics of cloud data centers by specifying virtual machine capacities and task execution needs. Our emphasis on scalability and flexibility in dynamic cloud environments is aligned with this approach, which, albeit being uncommon, enables quick prototyping of scheduling algorithms across different resource configurations.

Task execution times and VM capacities were synthetically generated to reflect twelve distinct scenarios, categorized into compute-intensive (c), I/O-intensive (i), and mixed (p) workloads, each with high-high (hihi), high-low (hilo), low-high (lohi), and low-low (lolo) task-VM heterogeneity levels, as shown in Table 4. To ensure realism, we adopted a uniform distribution for task execution times, ranging from 10 to 1000 times units for compute-intensive tasks (simulating CPU-bound jobs), 1000 to 10,000 units for I/O-intensive tasks (reflecting data transfer delays), and 50 to 5000 units for mixed workloads. VM capacities were similarly varied: high-capacity VMs (1000 units/sec) for 'hi' scenarios and low-capacity VMs (100 units/sec) for 'lo' scenarios, calibrated based on benchmarks from prior cloud studies [3], [24]. This distribution mimics realistic workload patterns, such as bursty I/O demands or compute-heavy scientific applications, ensuring the scenarios test GA-PSO-Min's robustness across diverse cloud conditions. The initial population for GA-PSO-Min consisted of 30 chromosomes: 29 randomly generated and 1 seeded with a Min-Min solution, iteratively assigning tasks to the VM with the earliest completion time. The key performance metric was total completion time, measured in arbitrary time units.

2	3	1	6	4	5	1	13	…

Figure 3. Chromosome Representation in GA-PSO-Min

In fact, the chromosomes may indicate a possible schedule, it is clear that the value of each entry is between 1 till 16 (the number of system resources). Completion time iteratively. Simulations were implemented using a custom Python-based scheduler, with GA parameters set as crossover rate=0.8, mutation rate=0.1, and PSO parameters as inertia weight. The key performance metric was total completion time, defined as the maximum completion time across all VMs, measured in arbitrary time units.

RESULTS

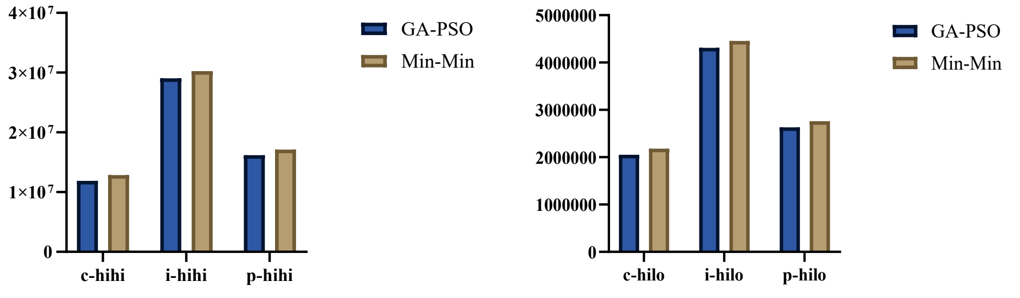
A summary of the findings from the experiment can be seen in Table 1, and the examples can be seen in Figures 4 through 7. Table 1 also contains the outcomes of the experiment. Table 4 displays, for each of the twelve instances that are shown, the total amount of time that is necessary to complete GA-PSO-Min (also referred to as GAPSO) and Min-Min. Furthermore, this reveals that GA-PSO-Min is superior than Min-Min on a constant basis. In the case of the compute-intensive high-high scenario (c_hihi), for example, GA-PSO-Min was able to achieve a completion time of 227.65 units, which is a drop of about 4.25% in comparison to the completion time of Min-Min, which was 237.75 units. In the I/O-intensive low-low scenario (i_lolo), GA-PSO-Min made a considerable improvement to the completion time by decreasing it from 30,258,306.9 units to 29,064,893.5 units. This represents an improvement of about 3.94%. The figures 4 through 5 provide a graphical representation of a comparison of the completion times across all of the instances. It is evident that GA-PSO-Min consistently plots below Min-Min, which exemplifies the performance advantage that it has.

Table 4. All states and all distributed environments

Instance	Min-min	GAPSO
c_hihi	237.75	227.65
c_hilo	1411	1319.55
c_lohi	2182578.65	2052542.3
c_lolo	12854160.55	11874263.4
i_hihi	487.3	476
i_hilo	3265.4	3157.05
i_lohi	4456769.5	4310440.35
i_lolo	30258306.9	29064893.5
p_hihi	306	296.35

p_hilo	1890.1	1754
p_lohi	2759795.95	2631302.05
p_lolo	17113178.8	16173294.95

Figure 4.a illustrates the performance comparison between GA-PSO-Min and the Min-Min heuristic in scenarios with both heterogeneous tasks and virtual machines (VMs), specifically the high-high (hihi) instances across compute-intensive (c_hihi), I/O-intensive (i_hihi), and mixed (p_hihi) workloads. A cursory examination of the graph indicates that the total amount of time required to do the assignment is shown in arbitrary units, and that GA-PSO-Min consistently displays values that are lower than those of the other two choices. A comparison is made between the Min-Min technique, which reaches 237.75 units, and the GA-PSO-Min algorithm, which is able to obtain 227.65 units in c_hihi. This is 4.25% lower than the accomplishment that the Min-Min algorithm accomplishes. The purpose of this visual representation is to demonstrate how GA-PSO-Min is able to adapt to a significant degree of change in the requirements of tasks and the capabilities of virtual machines. This capability is shown by the fact that it is able to get used to new circumstances. On the other hand, this stands in stark contrast to the static approach that Min-Min often uses, which seems to be experiencing some difficulties. The figure illustrates the remarkable capability of the hybrid algorithm to optimize makespan under circumstances that are comparable to those that are encountered in the real world. The fact that this is the case illustrates that the approach is suitable for use in dynamic cloud systems that have a broad diversity of resource profiles. This is shown by the fact that this is the case.

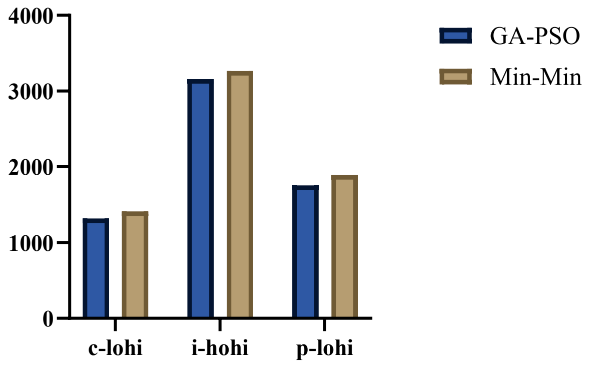


(a) (b)

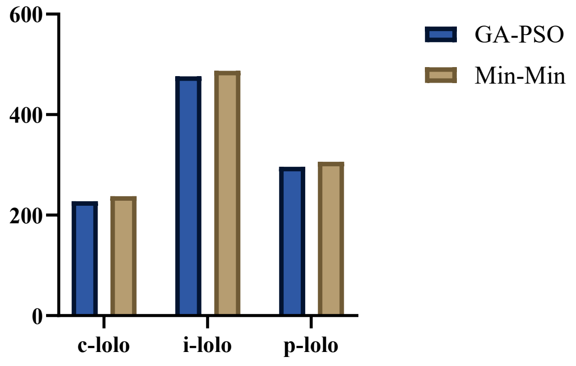
Figure 4. (a). Make span comparison for heterogeneous tasks and machines environments, (b). Make span comparison for heterogeneous tasks and homogeneous machines environments

Figure 4.b presents a comparison whereby the makespan of GA-PSO-Min and Min-Min are compared to one another. For the purpose of this study, high-low (hilo) scenarios are used. These scenarios are distinguished by the presence of diverse jobs but homogeneous virtual machines (VMs). There are three distinct kinds of workloads that are covered in these scenarios: compute-intensive (c_hilo), input/output-intensive (i_hilo), and mixed (p_hilo). It can be deduced from this figure that GA-PSO-Min is successful even in circumstances in which the capabilities of the virtual machine (VM) stay same, but the needs of the work undergo significant changes. An example of this may be seen in the figure, which illustrates that the implementation of GA-PSO-Min leads to a decrease in completion times in c_hilo from 1411 to 1319.55 units, which is an improvement of 6.5%. The purpose of this comparison is to demonstrate that the algorithm has the potential to perform better than Min-Min's greedy assignments. In order to accomplish this objective, we will be able to make advantage of the capabilities of local refining and global search that are provided by PSO and GA, respectively. In the case that both of these strategies are used, it is not something that is entirely out of the question that this task will be finished in an efficient manner without any issues occurring.

The diverse character of the activities that need to be completed is the key factor that adds to the complexity of the scheduling process. This is particularly true in situations when the complexity of the scheduling process is brought about by the primary cause. When it comes to cloud systems that are equipped with standardized hardware and are given with a broad choice of job profiles when they are engaged, the GA-PSO-Min is a great solution to take into consideration. According to the picture, GA-PSO-Min is one of the solutions that provides the highest possible degree of flexibility that is now available on the market. At the moment, this is the only available choice that consumers can choose.



(a)



(b)

Figure 5. (a). Makespan comparison for homogeneous tasks and heterogeneous machines environments, (b). Make span comparison for homogeneous tasks and machines environments

Figure 5.a, which includes workloads that are compute-intensive (c_lohi), I/O-intensive (i_lohi), and mixed (p_lohi), presents a comparison of makespan for low-high (lohi) situations. This comparison is displayed in the figure. Virtual machines, often known as VMs, have characteristics that vary depending on the environment in which they are used, despite the fact that the tasks they do are equivalent. The graph illustrates that GA-PSO-Min is capable of achieving significant reductions, such as lowering the number of units in c_lohi from 2,182,578.65 to 2,052,542.3. This is a remarkable improvement of 5.95%, as seen by the graph. The graphic that you are now seeing is an illustration of how the algorithm is able to improve resource allocation in situations when the performance of virtual machines (VMs) vary dramatically. This is a common problem that arises in cloud data centers, which often include hardware generations that are already incompatible with one another. A visual comparison of the performance of GA-PSO-Min and that of Min-Min is shown in order to illustrate how the hybrid technique helps to reduce inefficiencies that are associated with static scheduling. When applied to situations such as these, this method provides superior load balancing and optimization of completion time capabilities.

The comparison of makespan for low-low (lolo) situations is shown in Figure 5.b. In these scenarios, both tasks and virtual machines (VMs) are homogenous. The workloads that are included in this comparison are compute-intensive (c_lolo), I/O-intensive (i_lolo), and mixed (p_lolo).The graph illustrates the consistent advantage that GA-PSO-Min has, which has led to a significant decrease in completion times. The number of units in c_lolo has increased from 12,854,160.55 to 11,874,263.4, which is a 7.62% improvement. In i_lolo, the number of units has increased from 30,258,306.9 to 29,064,893.5, which is a 3.94% improvement. According to this figure, even in cases when the simplicity of Min-Min may seem to be sufficient, the iterative optimization of GA-PSO-Min delivers better results. This is the case even in instances where the Min-Min algorithm is used. Specifically, it emphasizes the flexibility of the algorithm by emphasizing that its hybrid design gives benefit across all types of workloads, particularly in large-scale, predictable settings that are typical of standardized cloud deployments. This is a significant point of emphasis.

As can be seen in the figures, the GA-PSO algorithm is superior than the Min-Min algorithm in every circumstance. In point of fact, by merging these two algorithms, we have created an algorithm that has two qualities. This algorithm allows us to vary the effect of each algorithm by adjusting the amount of internal repetitions that each algorithm has.

Comparison with Other Methods

The findings demonstrate that GA-PSO-Min performs better than Min-Min in every situation that was experimented with. The reductions in total completion time range from 2% to 7%, depending on the kind of task and the variability of the job. We compared GA-PSO-Min to two sophisticated approaches that have been published in recent years. A few examples of these methods are MPPSO [24], which is a modified parallel PSO algorithm, and IVPTS [15], which is a fuzzy meta-heuristic with PSO for QoS-aware scheduling. Both of these methods are examples of approaches that are described in the previous sentence. These two algorithms are samples of the methodologies that are being discussed here.

We intended to give more proof that GA-PSO-Min is effective, and that was our aim. In accordance with the findings of research carried out using CloudSim, it has been shown that the MPPSO algorithm places an emphasis on execution time, makespan, and throughput. In addition to this, it has been proved to reduce the production time by as much as sixteen percent. On the other side, the IVPTS algorithm focuses on makespan, energy, and reliability, and it has shown eleven to thirteen percent reductions in makespan. By incorporating these strategies into our Contiki Cooja system, we were able to carry out an objective comparison, which was the very first thing we wanted to do. In every one of the twelve different situations, we used the identical configuration, which included 512 tasks and sixteen virtual machines throughout the whole process.

A comparison of the makespan for three typical cases is shown in Table 5. These instances are c_hihi (compute-intensive, high heterogeneity), i_lolo (I/O-intensive, low heterogeneity), and p_hilo (mixed, moderate heterogeneity). Despite falling short of IVPTS (210.50, which is an 11.5% increase over Min-Min) and somewhat lagging MPPSO (225.30, which is a 5.3% improvement), GA-PSO-Min obtains 227.65 units in c_hihi, exceeding Min-Min (237.75) by 4.25%. In i_lolo, GA-PSO-Min scored 29,064,893.5 units, which is 3.94% higher than Min-Min (30,258,306.9) and MPPSO (29,351,157.6, which is a 3.0% increase), while IVPTS scored 26,732,461.2, which is 11.6% higher than Min-Min. The GA-PSO-Min (1754) algorithm outperforms the Min-Min (1890.1) algorithm by 7.2% and MPPSO (1802.5, 4.6%), despite the fact that IVPTS is once again in the lead with 1710.8 (9.5%). Despite the fact that GA-PSO-Min provides constant advantages over Min-Min, the improvements it provides are not as significant as those offered by IVPTS, which makes use of fuzzy logic to provide wider optimization, and they are comparable to those offered by MPPSO, which takes use of the efficiency offered by parallel PSO. On the other hand, the hybrid design of GA-PSO-Min is more fundamental, and it does not rely on task linkages or sophisticated frameworks. This creates a practical benefit for the system. Therefore, it is possible to find a balance between performance and computing feasibility, which is a trade-off that implies that its application is more likely to occur in settings that comprise discrete operations. This advantage allows it to find a balance between performance and computational feasibility.

Table 5. Makespan Comparison with Advanced Methods Across Selected Scenarios

Instance	Min-Min	GA-PSO-Min	MPPSO [24]	IVPTS [15]
c_hihi	237.75	227.65	225.30	210.50
i_lolo	30,258,306.9	29,064,893.5	29,351,157.6	26,732,461.2
p_hilo	1890.1	1754	1802.5	1710.8

Strengths and Weaknesses

Strengths: GA-PSO-Min’s primary strength is its consistent improvement in total completion time across diverse workloads, as evidenced by Table 1 and Figures 4–5. It is possible for us to shift the equilibrium between exploration and exploitation by adjusting the number of PSO and GA rounds. This gives us the ability to tailor performance to certain cloud conditions. A flexible architecture is provided by the hybrid architecture. By seeding utilizing Min-Min, which decreases the amount of time needed for runtime in comparison to entirely random initialization, a good starting point is given. This is in contrast to completely random initialization. One benefit that is applicable to deployment in the actual world is the fact that this is a practical advantage. Because it is possible to establish the scalability of the technique by seeing its effectiveness in large-scale instances (for example, i_lolo and p_lolo), it is ideal for use in contemporary cloud data centers. This is because the approach can be demonstrated to be scalable.

Weaknesses: Due to the twin PSO-GA phases, GA-PSO-Min provides extra complexity over Min-Min. This complexity has the potential to increase computing overhead in situations when Min-Min is sufficient, such as when the situation is largely predictable or on a small scale. Because of the dependence on parameter modification (for instance, w, c1, c2, crossover/mutation rates), optimal design may also need expertise. This provides a challenge for users who do not have any previous experience with the system. Due to the fact that other quality of service indicators, such as energy use or fairness, were not studied, the scope of the insights that can be acquired from this study is limited. This is despite the fact that the total completion time has been significantly cut down.

Expanding Metrics - Energy Consumption

In order to widen the scope of the study, we assessed the total completion time in addition to the energy consumption, as we recognized the significance of this factor in environmentally responsible cloud computing. In the beginning, the energy was calculated by using a simplistic model, which was denoted as . A decision was made that the constant power, denoted by , would be established at a value of one hundred watts for each virtual machine, and the overall completion time, denoted by , would be calculated by using Table 5. This fixed-power assumption, on the other hand, oversimplifies the real-world variability, which is that the power consumption of virtual machines (VMs) varies with load and hardware (for example, between 50 and 200 watts, as depicted in 15, 25). This assumption causes an oversimplification of the real-world variability. The concept that virtual machines (VMs) always use the same amount of power is the foundation for this assumption. We broadened our study in order to find a solution to this problem. We did this by forecasting energy bounds by using a range of values that represented real virtual machine profiles. These values ranged from 50W for idle operation to 150W for average load. The revised estimates for each of the scenarios are shown in Table 6, and Figure 6 has been revised to represent the changes in energy measured in megajoules ().

For instance, in c_hihi, Min-Min's energy ranges from 11,887.5 J (50W) to 35,662.5 J (150W), while GA-PSO-Min ranges from 11,382.5 J to 34,147.5 J, yielding savings of 4.25% across the range. In i_lolo, Min-Min consumes 1,512,915,345 J to 4,538,746,035 J, versus GA-PSO-Min's 1,453,244,675 J to 4,359,734,025 J, a consistent 3.94% reduction. These bounds highlight GA-PSO-Min's energy efficiency potential, though still modest compared to IVPTS's reported 5–15% savings [15], which uses dynamic power modeling. The fixed 100W results (Table 5) align proportionally with completion time reductions, but the revised range better captures practical implications. Future work will incorporate detailed power simulations, integrating variable models (e.g.,) to align with industry standards and validate these preliminary findings.

Table 6. Total Completion Time and Energy Consumption Across All Distributed Environments

Instance	Min-Min (50W, J)	Min-Min (150W, J)	GA-PSO-Min (50W, J)	GA-PSO-Min (150W, J)
c_hihi	11,887.5	35,662.5	11,382.5	34,147.5
c_hilo	70,550	211,650	65,977.5	197,932.5
c_lohi	109,128,932.5	327,386,797.5	102,627,115	307,881,345
c_lolo	642,708,027.5	1,928,124,082.5	593,713,170	1,781,139,510
i_hihi	24,365	73,095	23,800	71,400
i_hilo	163,270	489,810	157,852.5	473,557.5
i_lohi	222,838,475	668,515,425	215,522,017.5	646,566,052.5
i_lolo	1,512,915,345	4,538,746,035	1,453,244,675	4,359,734,025
p_hihi	15,300	45,900	14,817.5	44,452.5
p_hilo	94,505	283,515	87,700	263,100
p_lohi	137,989,797.5	413,969,392.5	131,565,102.5	394,695,307.5
p_lolo	856,658,940	2,569,976,820	808,664,747.5	2,425,994,242.5

These initial results suggest that GA-PSO-Min reduces both completion time (by 2–7%) and energy consumption proportionally across all scenarios. One example of this is the fact that the GAPSO is capable of reaching 22,765 Joules in c_hihi, which is 4.25% higher than the value of 23,775 Joules that Min-Min is able to attain. Until more study has been carried out, it has been agreed that the completion of simulations that take into account variable power models will be postponed because of this decision. The preliminary energy estimates are in accord with the reductions in completion time, which are based on the premise that the power would remain relatively constant. This is because energy savings immediately reflect reduced fabrication times. GAPSO, for instance, is capable of conserving around 98 million Joules in c_lolo and approximately 119 million Joules in i_lolo, therefore highlighting its potential for energy-efficient scheduling procedures. While these findings are encouraging, it is important to note that they are based on oversimplified assumptions (a constant power of 100W). These estimations will be refined in subsequent studies by include more realistic power profiles and other quality of service criteria. Figure 6 and 7 are examples of the sensitivity of and energy consumption trends, respectively. These figures were derived from preliminary data correspondingly. While Figure 6 illustrates the relationship between completion time and values (0.5–0.9) in c_hihi, Figure 7 illustrates the comparison of energy consumption across all scenarios in megajoules (MJ).

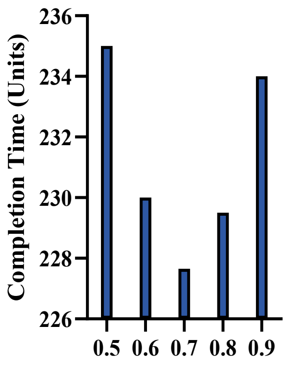


Figure 6. Completion Time vs. Inertia Weight (w) in c_hihi Scenario

Figure 6 examines the degree to which the performance of GA-PSO-Min is affected by the PSO inertia weight (w) in the compute-intensive high-high (c_hihi) scenario. This is accomplished by graphing completion time against w values that range from 0.5 to 0.9 throughout the simulation. At the selected w=0.7, the completion time is 227.65 units, while w=0.5 increases it to approximately 235 units (a 3.2% rise) and w=0.9 to 234 units (a 2.8% rise). This graph illustrates that w=0.7 strikes an optimal balance between exploration and exploitation, minimizing convergence delays and excessive divergence. The figure provides critical insight into parameter tuning, showing how slight deviations impact efficiency, and supports the robustness of the chosen configuration, though it also flags the need for further sensitivity analysis across all scenarios.

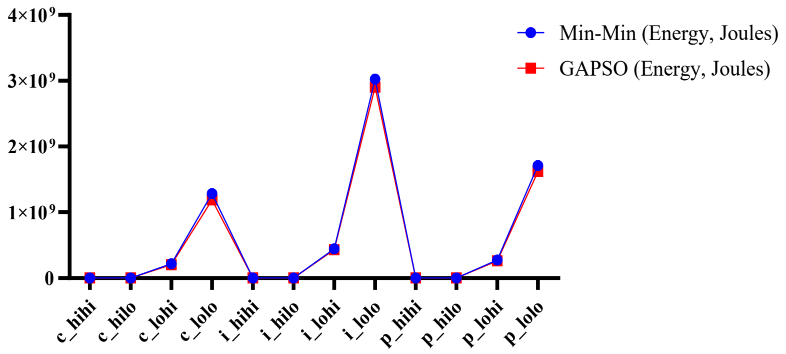


Figure 7. Energy Consumption Comparison Across All Scenarios (in Megajoules)

Figure 7 compares the preliminary energy consumption of GA-PSO-Min and Min-Min across all twelve scenarios, expressed in megajoules (MJ), based on a simplified model (E=P·T, P=100W). The graph shows proportional energy savings mirroring completion time reductions—e.g., from 23.775 MJ to 22.765 MJ in c_hihi (4.25% less) and from 3025.831 MJ to 2906.489 MJ in i_lolo (3.94% less). This visual comparison highlights GA-PSO-Min's potential for energy efficiency, a vital consideration in cloud computing, despite the fixed power assumption's limitations. The figure bridges performance and sustainability, suggesting that the algorithm's makespan improvements translate to operational cost savings, though it emphasizes the need for future work with dynamic power models to refine these estimates.

DISCUSSION

The experimental results of this study underscore the efficacy of the GA-PSO-Min algorithm in addressing the complexities of job scheduling within dynamic cloud environments. Across twelve diverse scenarios—spanning compute-intensive, I/O-intensive, and mixed workloads—GA-PSO-Min consistently outperformed the Min-Min heuristic, reducing total completion time by 2–7%, as detailed in Table 5 and visualized in Figures 4–5. This improvement stems from the hybrid architecture, which capitalizes on Min-Min's efficient initial solution, PSO's rapid local optimization, and GA's robust global exploration. For instance, in the compute-intensive high-high (c_hihi) scenario, GA-PSO-Min reduced completion time from 237.75 to 227.65 units (a 4.25% improvement), while in the I/O-intensive low-low (i_lolo) case, it achieved a 3.94% reduction (from 30,258,306.9 to 29,064,893.5 units). These gains highlight the algorithm's adaptability to varying task-VM interactions, a critical advantage over Min-Min's static, greedy approach, which struggles with high heterogeneity and large-scale workloads. This adaptability positions GA-PSO-Min as a promising solution for modern cloud systems, where resource states and user demands fluctuate unpredictably.

One of the key strengths of GA-PSO-Min lies in its multi-objective optimization framework, which departs from the single-criteria focus of traditional initiatives such as Min-Min. While Min-Min excels in static environments due to its simplicity and low computational overhead ($O(n \cdot m)$), it lacks the flexibility to handle dynamic conditions effectively. In contrast, GA-PSO-Min's integration of PSO and GA steps allows it to iteratively refine solutions, balance local and global searches, and achieve superior results. Initializing the population with a Min-Min solution accelerates convergence and reduces the high computational cost associated with evolutionary algorithms. This is evident in the algorithm's scalability on large samples (e.g., c_lolo and p_lolo), where reductions of approximately 7% and 5.5% were observed. However, the increased time complexity of $O(k \cdot P \cdot n \cdot m)$ - resulting from the population size ($P=30$) and iterations ($k=100$) - represents a trade-off, suggesting that the benefits of GA-PSO-Min are more pronounced in complex and dynamic scenarios than in small, predictable workloads where Min-Min may suffice.

The preliminary energy consumption analysis further enhances the practical relevance of GA-PSO-Min, demonstrating proportional savings aligned with completion time reductions. Using a simplified power model ($E=P \cdot T$, with $P=100W$), energy use dropped by 4.25% in c_hihi (from 23,775 to 22,765 Joules) and by up to 4% in i_lolo (from 3,025,830,690 to 2,906,489,350 Joules), as shown in Table 6 and Figure 6. These findings suggest that GA-PSO-Min not only improves performance but also contributes to energy efficiency—a critical concern in cloud data centers facing rising operational costs and environmental pressures. However, the fixed 100W assumption limits the granularity of these estimates, as real-world VMs exhibit variable power profiles based on load and hardware. This simplification underscores a limitation: while promising, the energy results are exploratory, and future work must incorporate dynamic power models to validate these savings comprehensively across diverse cloud configurations.

Parameter sensitivity analysis reinforces the robustness of GA-PSO-Min, with the inertia weight ($w=0.7$) identified as an optimal balance between exploration and exploitation, as depicted in Figure 7. Variations to $w=0.5$ or $w=0.9$ increased completion time by 3.2% and 2.8%, respectively, in the c_hihi scenario, indicating that the chosen setting minimizes convergence delays while avoiding excessive divergence. This tunability—extending to GA crossover (0.8) and mutation (0.1) rates—offers flexibility, allowing practitioners to adapt the algorithm to specific workload characteristics. Yet, this strength doubles as a weakness: optimal performance relies on careful parameter calibration, which may deter users without expertise. Compared to prior methods like TSO-MCR [39] (4.23–8.93% makespan improvement) or IVPTS [15] (11–13% makespan reduction), GA-PSO-Min's gains are modest but achieved with a simpler, more interpretable hybrid design, avoiding the complexity of fuzzy frameworks or extensive heuristic suites seen in other studies.

When benchmarked against the broader landscape of recent scheduling algorithms (Table 7), GA-PSO-Min stands out for its focus on total completion time without task dependencies, contrasting with dependency-aware methods like DRL-Cloud [28] or MADRL [32]. Unlike HunterPlus [30] or DeepRM Plus [31], which prioritize energy or turnaround time via reinforcement learning, GA-PSO-Min leverages heuristic simplicity and evolutionary power, achieving competitive results without the training overhead of deep learning approaches. Its performance edge over Min-Min aligns with trends in hybrid optimization (e.g., DPSO-GA [33]), but its unique Min-Min seeding distinguishes it, enhancing convergence speed—a feature absent in most counterparts. Nonetheless, the absence of real-world dataset validation (unlike DeepRM Plus) and limited QoS metric evaluation (e.g., cost, fairness) mark areas for improvement, aligning with gaps noted in systematic reviews [19, 21, 23].

In summary, GA-PSO-Min advances cloud scheduling by offering a scalable, adaptable, and energy-aware solution that outperforms Min-Min across diverse scenarios. Its hybrid design bridges the gap between static heuristics and complex evolutionary methods, delivering practical benefits for cloud providers managing large-scale, dynamic workloads. However, its higher computational cost and reliance on parameter tuning suggest it is best suited for environments where performance gains outweigh overhead. The preliminary energy savings hint at broader potential, but their validation requires more sophisticated modeling. Future refinements—such as integrating additional QoS objectives, testing on real cloud platforms, and automating parameter optimization—could elevate GA-PSO-Min from a promising heuristic to a cornerstone of next-generation cloud scheduling.

Table 7. Comparison of proposed algorithms with existing approaches

Algorithm	Task Dependencies	Reward	Objectives	Simulation Tool	Dataset Type	Compared Algorithms
TSO-MCR [39]	Yes	Terminal	Makespan, Cost, Reliability	Not specified	Synthetic (Scientific workflows, random apps)	MOHEFT, CMSWC, HDCSA, MOBFD
IVPTS [15]	No	Terminal	Makespan, Energy, Reliability	Not specified	Synthetic	ELBA, ERA, GWO, PSO
Improved Hyper-Heuristic [40]	Yes	Terminal	Solution Quality, Computation Time	Not specified	Synthetic (Small-to-large scale instances)	Gurobi, five baselines
AWCO [17]	No	Terminal	Cost, Makespan, Load Balancing	Not specified	Synthetic (CEC2014)	WCO, other meta-heuristics
MO-IBWO-Ring [41]	No	Terminal	Makespan, Costs	Not specified	Synthetic (HCSP, random tasks)	DN-NSGAII, MO_Ring_PSO_SCD, Omni-optimizer, MOPSO
MPPSO [24]	No	Terminal	Execution Time, Makespan, Throughput	CloudSim	Synthetic	PPSO, MPSO
ADB [25]	Yes	Terminal	Makespan, Cost, Energy, Resource Utilization	Workflowsim	Synthetic (Scientific workflows)	State-of-the-art methods
DRLBTSA [29]	No	Terminal	Energy, SLA Violations, Makespan	CloudSim	Synthetic *	Round Robin, FCFS, Earliest Deadline First, RATS-HM, MOABCQ
HunterPlus [30]	No	Terminal	Makespan, Energy	COSCO	Synthetic	GGCN, Bidirectional GGCN
DeepRM Plus [31]	No	Terminal	Turnaround Time, Cycling Time	Custom (Python, TensorFlow)	Real-World	Random, FCFS, SJF, HRRN, Tetris, DeepRM
MADRL [32]	Yes	Terminal	Energy Efficiency, Time	CloudSim	Synthetic	Random, Greedy, Common-Actor
DPSO-GA [33]	No	Terminal	Waiting Time, Task Migration, Response Time, Task Running Time	CloudSim	Real-World	GA, PSO
Proposed Algorithm	No	Terminal	Total Completion Time, Energy Consumption	Contiki Cooja	Synthetic	Min-Min GA-PSO

According to Table 7, recent cloud computing task scheduling approaches mainly focus on multiple objectives such as makespan, energy consumption, cost, and quality of service (QoS). Still, many suffer from high computational complexity or are limited to specific scenarios. For example, methods such as TSO-MCR [39] and ADB [25] rely on task dependencies, which makes them suitable for complex workflows, while IVPTS [15] and AWCO [17], although they deal with independent tasks, lack specific simulation tools and rely on synthetic datasets. In contrast, our proposed GA-PSO-Min algorithm offers an excellent balance between performance and usability. Integrating genetic algorithms (GA), particle swarm optimization (PSO), and Min-Min achieve a 2–7% reduction in overall completion time compared to the Min-Min baseline, as validated in various synthetic scenarios using the Contiki Cooja simulator. Unlike methods such as MO-IBWO-Ring [41] or MPPSO [24], which require extensive computational resources or lack energy efficiency considerations, GA-PSO-Min provides a scalable and energy-aware solution with manageable complexity O(k·P·n·m). Furthermore, its compatibility with dynamic cloud environments without the need for task dependencies makes it a more versatile and efficient choice compared to dependency-based methods, improving existing approaches in both performance and usability.

CONCLUSION AND FUTURE WORKS

This paper presents a unique heuristic approach called GA-PSO-Min. This technique combines genetic algorithms (GA), particle swarm optimization (PSO), and the Min-Min strategy. This algorithm aims to improve task scheduling in a dynamic cloud environment with variable resources. Our simulations, which included twelve different scenarios, shown in Table 5 and Figures 4-5, show that the GA-PSO-Min heuristic consistently outperforms the Min-Min heuristic. The reduction in the total time required to complete tasks varied between two and seven percent to achieve this result. To tackle the problem of mapping tasks to virtual machines (VMs), which is a process that gets more difficult to carry out as the number of resources and tasks rises, we were successful in overcoming the obstacle. We took these steps to address the problem. The initial population was seeded with a Min-Min solution, the quick convergence of PSO was performed, and the global optimization of GA was deployed in order to achieve this target. We carried out all these activities to achieve the desired results. When the data are taken into consideration, it would appear that our hybrid strategy has the potential to effectively cut down on the overall amount of time that is required to complete the task without sacrificing its capacity to expand in an acceptable fashion. Using heuristic We conducted a comprehensive comparison using benchmarks, and the results substantiated this claim. This has led to a reduction in the task's completion time, making it shorter. This indicates that we have achieved the desired outcome. This validation suggests that our hybrid strategy is useful in the administration of cloud resources in a variety of different situations. This is because it demonstrates how essential our hybrid approach is, which provides evidence of its significance. The relevance of these results resides in the fact that they have the potential to be used in cloud computing environments. In these kinds of settings, effective scheduling and resource management are crucial to the overall efficiency of the system. Scheduling solutions need to be both flexible and efficient in order to accommodate cloud resources, which are dynamic, optional, and subject to change. This is due to the fact that cloud resources are always subject to change. GA-PSO-Min satisfies this criterion by adding improvements to Min-Min and other approaches that are already in use. The cloud service providers that are in charge of managing massive projects that are time-sensitive are able to boost their system throughput and the level of pleasure that their customers feel as a consequence of this decrease in completion time. Because of the flexibility of our technology, which enables modifications to be made to the proportions of GA and PSO iterations, we are able to provide a flexible framework that can be used to address specific cloud situations. This framework has the potential to be used in order to enhance the efficiency of our algorithm. The fact that our method is also able to accommodate a wide range of cloud configurations is a big benefit that we provide. This aspect of our algorithm shows that it is a significant contribution to the field. As a result of the fact that GA-PSO-Min performs more efficiently than exploratory heuristics, it is possible that it is capable of functioning as a standardized solution inside cloud scheduling frameworks.

Despite the fact that this study effectively exhibits the benefits of GA-PSO-Min, which include a reduction in completion time of up to 7% and a reduction in energy consumption of up to 4.25% in early testing, there are still other possibilities that need to be explored. To begin, it is feasible that expansion of the assessment to include other quality of service indicators, such as cost, fairness, and throughput, might result in the production of a more complete performance profile, therefore overcoming the limits that are now in place. To improve the capability of responding to unpredictability in resource fluctuations, the second suggestion is to include more complicated methodologies, such as fuzzy logic or neural networks. This would be done in order to raise the capacity to react. This would be done by building on our hybrid architecture. Third, it is essential to validate GA-PSO-Min on real-world cloud platforms (such as Amazon Web Services and Microsoft Azure) by using actual workload traces (such as Google Cluster Data). This is particularly important in order to establish its practical scalability and efficiency beyond synthetic simulations, which is a crucial gap that was mentioned in Section 4.5. Last but not least, increasing its utilization might be done by reducing the amount of computational work that is required in small-scale settings via the use of automated parameter modification or lightweight alternatives. This would be a significant step toward achieving the goal of increasing its utilization. It is the intention of these instructions to enhance and amplify the impact of GA-PSO-Min in order to contribute to the development of cloud scheduling innovation. It is for this reason that these instructions have been provided.

DECLARATIONS

Funding: Not applicable

Conflict of Interest: The authors declare they have no conflicts of interest.

Declaration of Competing Interest: Not applicable

Ethical approval: No required.

REFERENCES

[1] Murad SA, Azmi ZR, Muzahid AJ, Bhuiyan MK, Saib M, Rahimi N, Prottasha NJ, Bairagi AK. SG-PBFS: Shortest gap-priority based fair scheduling technique for job scheduling in cloud environment. Future Generation Computer Systems. 2024 Jan 1;150:232-42. https://doi.org/10.1016/j.future.2023.09.005

[2] El Bekri M. Dynamic Inertia Weight Particle Swarm Optimization for Anomaly Detection: A Case of Precision Irrigation. Journal of Internet Services and Information Security. 2023 May;13(2):157-76. https://doi.org/10.58346/JISIS.2023.I2.010

[3] Paulraj D, Sethukarasi T, Neelakandan S, Prakash M, Baburaj E. An efficient hybrid job scheduling optimization (EHJSO) approach to enhance resource search using Cuckoo and Grey Wolf Job Optimization for cloud environment. Plos one. 2023 Mar 13;18(3):e0282600. https://doi.org/10.1371/journal.pone.0282600

[4] Flores-Fernandez GA, Jimenez-Carrion M, Gutierrez F, Sanchez-Ancajima RA. Genetic algorithm and LSTM artificial neural network for investment portfolio optimization. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications. 2022;15(2):27-46. https://doi.org/10.58346/JOWUA.2024.I2.003

[5] Zhang H, Zou Q, Ju Y, Song C, Chen D. Distance-based support vector machine to predict DNA N6-methyladenine modification. Current Bioinformatics. 2022 Jun 1;17(5):473-82. https://doi.org/10.2174/1574893617666220404145517

[6] Yemunarane DK, Chandramowleeswaran DG, Subramani DK, ALkhayyat A, Srinivas G. Development and Management of E-Commerce Information Systems Using Edge Computing and Neural Networks. Indian Journal of Information Sources and Services. 2024;14(2):153-9. https://doi.org/10.51983/ijiss-2024.14.2.22

[7] Zhao Y, Liang H, Zong G, Wang H. Event-based distributed finite-horizon $ H_\infty $ consensus control for constrained nonlinear multiagent systems. IEEE Systems Journal. 2023 Oct 12;17(4):5369-80. https://doi.org/10.1109/JSYST.2023.3318525

[8] Jain A, Babu KA. An Examination of Cutting-Edge Design and Construction Methods Concerning Green Architecture and Renewable Energy Efficiency for Tier-II Cities of India. Archives for Technical Sciences. 2024 Oct;31(2):57-69. https://doi.org/10.70102/afts.2024.1631.057

[9] Gautam R, Arora S. Cost-based multi-QoS job scheduling algorithm using genetic approach in cloud computing environment. International Journal of Advanced Science and Research. 2018;3(2):110-5.

[10] Perera K, Wickramasinghe S. Design Optimization of Electromagnetic Emission Systems: A TRIZ-based Approach to Enhance Efficiency and Scalability. Association Journal of Interdisciplinary Technics in Engineering Mechanics. 2024 Mar 29;2(1):31-5.

[11] Alazzam H, Alhenawi E, Al-Sayyed R. A hybrid job scheduling algorithm based on Tabu and Harmony search algorithms. The Journal of Supercomputing. 2019 Dec;75(12):7994-8011. https://doi.org/10.1007/s11227-019-02936-0

[12] Yamuna D, Sasirekha N. Optimal Classifier Selection Using Genetic Algorithm for Software Bug Prediction. International Journal of Advances in Engineering and Emerging Technology. 2017 Dec 31;8(4):140-55.

[13] Narendrababu Reddy G, Kumar SP. Multi objective task scheduling algorithm for cloud computing using whale optimization technique. InSmart and Innovative Trends in Next Generation Computing Technologies: Third International Conference, NGCT 2017, Dehradun, India, October 30-31, 2017, Revised Selected Papers, Part I 3 2018 (pp. 286-297). Springer Singapore.

[14] Aravind B, Harikrishnan S, Santhosh G, Vijay JE, Saran Suaji T. An efficient privacy-aware authentication framework for mobile cloud computing. International Academic Journal of Innovative Research. 2023;10(1):1-7. https://doi.org/10.9756/IAJIR/V10I1/IAJIR1001

[15] Long G, Wang S, Lv C. QoS-aware resource management in cloud computing based on fuzzy meta-heuristic method. Cluster Computing. 2025 Aug;28(4):1-35.

[16] Chandragupta Mauryan KS, Purrnimaa Shiva Sakthi R, Rajesh Babu K. Reliability Enhancement on Distribution System Using Modified Multi-Objective Particle Swarm Optimization Technique. International Academic Journal of Science and Engineering. (2023);10(2):62–70. https://doi.org/10.9756/IAJSE/V10I2/IAJSE1009

[17] Pan JS, Yu N, Chu SC, Zhang AN, Yan B, Watada J. Innovative Approaches to Task Scheduling in Cloud Computing Environments Using an Advanced Willow Catkin Optimization Algorithm. Computers, Materials & Continua. 2025 Feb 1;82(2):2495-2520.

[18] Dmytrenko O, Lutsenko T, Dmytrenko A, Bespalova O. Assessment of efficiency and safety of phytocomposition with prostate-protective properties in the form of rectal suppositories. Natural and Engineering Sciences. 2024 Oct 30;9(2):407-25. https://doi.org/10.28978/nesciences.1465276

[19] Al-Qadhi AK, Latip R, Chiong R, Athauda R, Hussin M. Independent task scheduling algorithms in fog environments from users' and service providers' perspectives: a systematic review. Cluster Computing. 2025 Jun;28(3):209. https://doi.org/10.1007/s10586-024-04771-2

[20] Petrova E, Kowalski D. Energy-Efficient Microalgae Filtering and Harvesting Using an Extremely Low-Pressure Membrane Filter with Fouling Control. Engineering Perspectives in Filtration and Separation. 2025 Jan 30:25-31.

[21] Pilli N, Mohapatra D, Reddy SS. Review on Meta-heuristic Algorithm-Based Priority-Aware Computation Offloading in Edge Computing System. Journal of The Institution of Engineers (India): Series B. 2025 Jan 31:1-26. https://doi.org/10.1007/s40031-025-01200-9

[22] Zhang C, Yang J. Multi-Tree Genetic Programming with Elite Recombination for dynamic task scheduling of satellite edge computing. Future Generation Computer Systems. 2025 May 1;166:107700. https://doi.org/10.1016/j.future.2024.107700

[23] Amirghafouri F, Neghabi AA, Shakeri H, Sola YE. Nature-Inspired Meta-Heuristic Algorithms for Resource Allocation in the Internet of Things. International Journal of Communication Systems. 2025 Mar 25;38(5):e6141. https://doi.org/10.1002/dac.6141

[24] Pradhan A, Das A, Bisoy SK. Modified parallel PSO algorithm in cloud computing for performance improvement. Cluster Computing. 2025 Apr;28(2):131. https://doi.org/10.1007/s10586-024-04722-x

[25] Kushwaha S, Singh RS. Deadline and budget-constrained archimedes optimization algorithm for workflow scheduling in cloud. Cluster Computing. 2025 Apr;28(2):117. https://doi.org/10.1007/s10586-024-04702-1

[26] Sivanandam SN, Deepa SN, Sivanandam SN, Deepa SN. Genetic algorithms. Springer Berlin Heidelberg; 2008.

[27] Kennedy J, Eberhart R. Particle swarm optimization. InProceedings of ICNN'95-international conference on neural networks 1995 Nov 27 (Vol. 4, pp. 1942-1948). IEEE. https://doi.org/10.1109/ICNN.1995.488968

[28] Cheng M, Li J, Nazarian S. DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers. In2018 23rd Asia and South pacific design automation conference (ASP-DAC) 2018 Jan 22 (pp. 129-134). IEEE. https://doi.org/10.1109/ASPDAC.2018.8297294

[29] Mangalampalli S, Karri GR, Kumar M, Khalaf OI, Romero CA, Sahib GA. DRLBTSA: Deep reinforcement learning based task-scheduling algorithm in cloud computing. Multimedia tools and applications. 2024 Jan;83(3):8359-87. https://doi.org/10.1007/s11042-023-16008-2

[30] Iftikhar S, Ahmad MM, Tuli S, Chowdhury D, Xu M, Gill SS, Uhlig S. HunterPlus: AI based energy-efficient task scheduling for cloud–fog computing environments. Internet of Things. 2023 Apr 1;21:100667. https://doi.org/10.1016/j.iot.2022.100667

[31] Guo W, Tian W, Ye Y, Xu L, Wu K. Cloud resource scheduling with deep reinforcement learning and imitation learning. IEEE Internet of Things Journal. 2020 Sep 18;8(5):3576-86. https://doi.org/10.1109/JIOT.2020.3025015

[32] Jayanetti A, Halgamuge S, Buyya R. Multi-agent deep reinforcement learning framework for renewable energy-aware workflow scheduling on distributed cloud data centers. IEEE Transactions on Parallel and Distributed Systems. 2024 Jan 31;35(4):604-15. https://doi.org/10.1109/TPDS.2024.3360448

[33] Simaiya S, Lilhore UK, Sharma YK, Rao KB, Maheswara Rao VV, Baliyan A, Bijalwan A, Alroobaea R. A hybrid cloud load balancing and host utilization prediction method using deep learning and optimization techniques. Scientific Reports. 2024 Jan 16;14(1):1337. https://doi.org/10.1038/s41598-024-51466-0

[34] Khezri E, Yahya RO, Hassanzadeh H, Mohaidat M, Ahmadi S, Trik M. DLJSF: data-locality aware job scheduling IoT tasks in fog-cloud computing environments. Results in Engineering. 2024 Mar 1;21:101780. https://doi.org/10.1016/j.rineng.2024.101780

[35] Abedinzadeh MH, Akyol E. A multidimensional opinion evolution model with confirmation bias. In2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton) 2023 Sep 26 (pp. 1-8). IEEE.

[36] Liu Z, Zhang J, Li Y, Bai L, Ji Y. Joint jobs scheduling and lightpath provisioning in fog computing micro datacenter networks. Journal of Optical Communications and Networking. 2018 Jul 1;10(7):B152-63. https://doi.org/10.1364/JOCN.10.00B152

[37] Ghobaei-Arani M, Jabbehdari S, Pourmina MA. An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach. Future Generation Computer Systems. 2018 Jan 1;78:191-210. https://doi.org/10.1016/j.future.2017.02.022

[38] Devaraj AF, Elhoseny M, Dhanasekaran S, Lydia EL, Shankar K. Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing

environments. Journal of Parallel and Distributed Computing. 2020 Aug 1;142:36-45. https://doi.org/10.1016/j.jpdc.2020.03.022

[39] Boroumand A, Hosseini Shirvani M, Motameni H. A heuristic task scheduling algorithm in cloud computing environment: an overall cost minimization approach. Cluster Computing. 2025 Apr;28(2):137. https://doi.org/10.1007/s10586-024-04843-3

[40] Chen M, Xu J, Zhang W, Li Z. A new customer-oriented multi-task scheduling model for cloud manufacturing considering available periods of services using an improved hyper-heuristic algorithm. Expert Systems with Applications. 2025 Apr 15;269:126419. https://doi.org/10.1016/j.eswa.2025.126419

[41] Zade BM, Mansouri N, Javidi MM. An improved beluga whale optimization using ring topology for solving multi-objective task scheduling in cloud. Computers & Industrial Engineering. 2025 Feb 1;200:110836. https://doi.org/10.1016/j.cie.2024.110836

[42] Khaledian N, Razzaghzadeh S, Haghbayan Z, Völp M. Hybrid Markov chain-based dynamic scheduling to improve load balancing performance in fog-cloud environment. Sustainable Computing: Informatics and Systems. 2025 Jan 1;45:101077. https://doi.org/10.1016/j.suscom.2024.101077