Journal on Artificial Intelligence

Tech Science Press

DOI: 10.32604/jai.2024.056259

ARTICLE

Hybrid Task Scheduling Algorithm for Makespan Optimisation in Cloud Computing: A Performance Evaluation

Abdulrahman M. Abdulghani[*]

Department of Communication Technology and Network, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang, 43400, Malaysia

*Corresponding Author: Abdulrahman M. Abdulghani. Email: abdulrahman.m.abdulghani@gmail.com

Received: 18 July 2024 Accepted: 04 September 2024 Published: 16 October 2024

ABSTRACT

Cloud computing has rapidly evolved into a critical technology, seamlessly integrating into various aspects of daily life. As user demand for cloud services continues to surge, the need for efficient virtualization and resource management becomes paramount. At the core of this efficiency lies task scheduling, a complex process that determines how tasks are allocated and executed across cloud resources. While extensive research has been conducted in the area of task scheduling, optimizing multiple objectives simultaneously remains a significant challenge due to the NP (Non-deterministic Polynomial) Complete nature of the problem. This study aims to address these challenges by providing a comprehensive review and experimental analysis of task scheduling approaches, with a particular focus on hybrid techniques that offer promising solutions. Utilizing the CloudSim simulation toolkit, we evaluated the performance of three hybrid algorithms: Estimation of Distribution Algorithm-Genetic Algorithm (EDA-GA), Hybrid Genetic Algorithm-Ant Colony Optimization (HGA-ACO), and Improved Discrete Particle Swarm Optimization (IDPSO). Our experimental results demonstrate that these hybrid methods significantly outperform traditional standalone algorithms in reducing Makespan, which is a critical measure of task completion time. Notably, the IDPSO algorithm exhibited superior performance, achieving a Makespan of just 0.64 milliseconds for a set of 150 tasks. These findings underscore the potential of hybrid algorithms to enhance task scheduling efficiency in cloud computing environments. This paper concludes with a discussion of the implications of our findings and offers recommendations for future research aimed at further improving task scheduling strategies, particularly in the context of increasingly complex and dynamic cloud environments.

KEYWORDS

Makespan; multi-objective optimisation; task scheduling; cloud computing; hybrid algorithms

1 Introduction

As cloud computing industries flourish, offering diverse applications in fields like education, healthcare, and telecommunications, the integration of task scheduling algorithms for cloud resources becomes essential [1–5]. This requirement arises from user demands, with cloud service providers responsible for service delivery and maintenance [6–8]. Cloud computing continues to evolve, especially with advancements in virtualisation, resource management, security, energy consumption, and task scheduling [8–11]. The growth in cloud computing and increasing user numbers introduce greater complexity in resource management and task execution, adhering to Service Level Agreements (SLA) [12]. An SLA defines the agreement between service providers and consumers [13]. In this dynamic environment, organising cloud systems to ensure user satisfaction and efficient demand management under the 'Pay as You Go' model becomes vital [14]. The organisation involves a resource manager, task manager, and scheduler, with the scheduler acting as an intermediary to distribute tasks using various scheduling techniques [15,16]. Fig. 1 depicts the cloud computing scheduling architecture, detailing the function of each component.

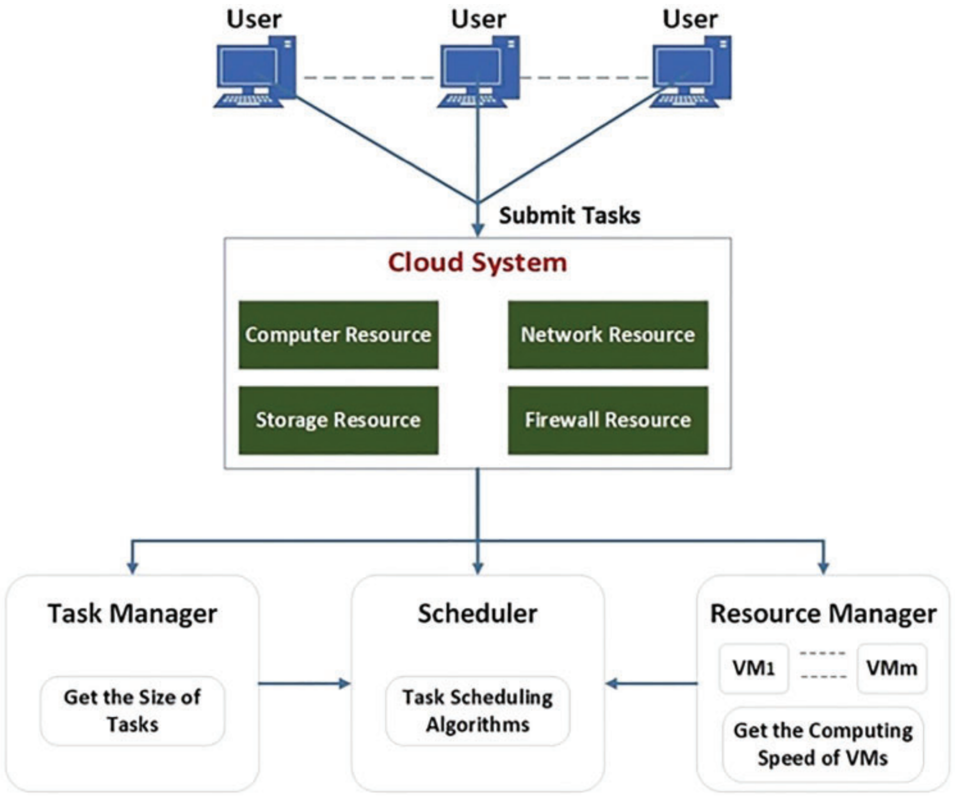


Figure 1: Scheduling mechanism in cloud computing

The role of the Scheduler in cloud computing is to effectively match incoming tasks with appropriate cloud resources or virtual machines (VMs) [17,18]. This process involves the Scheduler receiving task descriptions and resource manager details, and then assigning tasks based on quality of service (QoS) parameters. The Scheduler also determines the total number of tasks in the schedule. Task schedulers often utilise a range of optimisation algorithms to achieve different goals such as maximising resource utilisation, improving energy efficiency, balancing workloads, and minimising completion time. Tasks can have a wide range of requirements, including different hardware or software resources, as explained in [17]. The main goal of this paper is to investigate multi-objective task scheduling algorithms in cloud computing, specifically examining the impact of Makespan on load balancing, resource utilisation, cost, and energy efficiency.

The structure of this paper is outlined below: Sections 2 discusses related fieldwork. Sections 3 and 4 investigate hybridisation techniques found in the literature, discussing both the simulated results and the viable solutions they offer. Section 5 addresses the challenges of multi-objective solutions. Finally, Section 6 summarises the paper's findings.

2 Related Works

2.1 Single-Objective Optimisation

This section delves into the complexities of cloud computing task scheduling, critiquing the reliance on single-objective optimisation methods like Min-Min and Max-Min [16]. These methods, though efficient in certain aspects such as minimising job completion times, often result in significant drawbacks like resource underutilisation and load imbalances. Highlighting the necessity for a multi-faceted approach, it points out the limitations of focusing on a single parameter at the expense of others [2,13]. The passage advocates for multi-objective optimisation in cloud computing, emphasising the need to balance various goals and consider the wide-ranging needs of different stakeholders to achieve effective and efficient task management in cloud environments.

2.2 Multi-Objective Optimization

The paper introduces multi-objective optimisation algorithms to overcome the limitations of single-objective optimisation in task scheduling. These algorithms are designed to balance a variety of parameters such as task numbers and available VMs. The authors in [16] conducted a comprehensive literature review covering 2015–2018, analysing methods, parameters, and strategies in this field. Fig. 2 displays the number of research papers published on task scheduling solutions. Key algorithms identified include Ant Colony Optimisation (ACO), Particle Swarm Optimisation (PSO), and Genetic Algorithm (GA). The review emphasises the need for innovative approaches in machine learning, with some studies exploring hybrid methods for new task scheduling solutions in cloud computing.

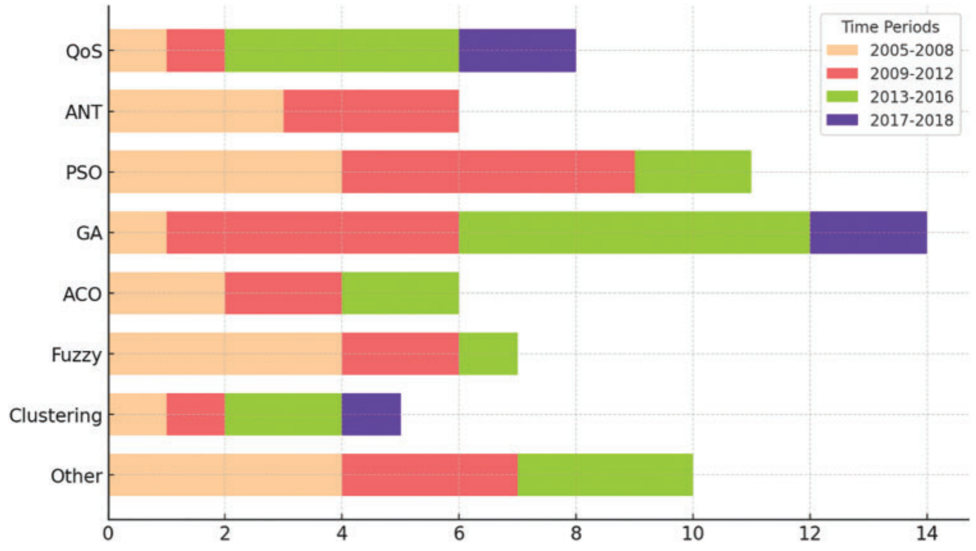


Figure 2: Survey summary about task scheduling methods

In the research summarised from [16], various studies in cloud computing and resource management are reviewed, with a focus on diverse parameters. One notable approach is the Adaptive Genetic Algorithm (AGA) discussed in [1], which employs binary coding for chromosomes to optimise resource scheduling. AGA's key feature, crossover mutation, is instrumental in generating new populations and has superior results in reducing Makespan, thereby improving load balancing. However, this approach tends to overlook the aspect of resource utilisation, highlighting a potential area for further development in task scheduling strategies.

In the study referenced in [2], researchers applied a novel hybrid EDA-GA algorithm, combining the Estimation of Distribution Algorithm (EDA) and Genetic Algorithm (GA). This method focused on improving task scheduling in cloud computing, targeting parameters like Makespan and load balancing. EDA enhanced mapping possibilities, while GA deepened the search for optimal solutions. The hybrid approach showed noticeable differences in performance compared to using EDA and GA separately and aligned with the findings in [1]. However, this study did not account for the cost implications of resource utilisation.

In [3], the focus is on Makespan as a critical factor influencing cloud application performance. The study explores energy conservation in cloud systems by examining job run times on virtual machines and resource availability. A key development is the multi-model EDA (mEDA), which calculates permutations and Voltage Levels of Supplies (VLSs) for task processing. This approach aims to reduce execution time and energy use, with enhancement operators improving the range of optimal solutions. While the study is effective in some areas, it fails to address the cost component or the complexities of heterogeneous systems in real-world settings. In their studies, researchers in [2] and [4] explored the use of Genetic Algorithm (GA) combined with other methods for job scheduling, highlighting it as an effective solution. A notable development is the efficient Hybrid Genetic Algorithm-Ant Colony Optimisation (HGA-ACO) [19], which applies a job allocation methodology to manage the demands of numerous cloud users. This approach merges top solutions from GA and ACO in a crossover, creating a new chromosome for optimal resource allocation. Their findings showed improvements in minimising response time and Makespan while increasing system throughput. However, aspects like load balancing and security were not addressed in their research. In [5], researchers introduced the Load balancing and Cost reduction Genetic Algorithm (LCGA), which combines load balancing GA and Cost GA to simultaneously address both parameters. This algorithm employs double fitness operations for load balancing and cost optimisation regarding completion time. LCGA outperformed standalone load balancing GA (LGA) and Cost GA (CGA), successfully demonstrating optimal task-resource mapping in CloudSim simulations. However, the approach did not consider task priority, which is important in scheduling and has an impact on energy consumption.

In [6], the Power-Aware and Real-Time Scheduling (PRTS) algorithm is introduced, aimed at reducing workflow costs and energy consumption. This algorithm comprises three key components: selecting cost-effective VMs while adhering to deadlines, tracking and utilising dynamic slack, and implementing the energy-efficient Dynamic Voltage/Frequency Saving (DVFS) methodology. The PRTS algorithm demonstrated up to 12.3% energy savings compared to the basic Energy Storage Systems (ESS) algorithm. However, its performance is somewhat limited as it does not fully consider the dynamics of real-world frequency variations. On the other hand, researchers [7] tackled task scheduling challenges using a modified Grey Wolf Optimisation (GWO) algorithm. This approach involves adjusting the fitness function to address multiple objectives, focusing on Makespan and cost reduction. According to their findings, the Modified Grey Wolf Optimisation (MGWO) algorithm outperforms the classic GWO and the Whale Optimisation Algorithm (WOA) regarding cost efficiency and Makespan. Nevertheless, their study is limited by the absence of practical application in real-world situations. In addition, this paper introduces a Memetic Algorithm (MA) based on the Genetic Algorithm (GA) for workflow scheduling. The main focus is on minimising completion time and resource costs. This method integrates hill-climbing optimisation with GA, improving individual solutions while conducting a global search. Consequently, the MA algorithm outperforms both GA and PSO in lowering Makespan. However, the study has a notable limitation in not addressing load balancing, which is considered a significant oversight in this context. In their study, researchers in [9] applied Particle Swarm Optimisation (PSO) to task scheduling, treating tasks as particles within swarms. The study focused on parameters like completion time and resource cost. They introduced an Improved Discrete Particle Swarm Optimisation (IDPSO) algorithm, utilising a sinusoidal strategy-based dynamic inertia weight for adaptive optimisation. Even though it was tested in a virtual environment with a small population size, IDPSO outperformed DPSO and First Come First Served (FCFS) algorithms in terms of completion time and convergence. In addition, the study by [10] explores the application of the Hybrid Genetic-Gravitational Search Algorithm (HG-GSA) in cost management for load scheduling, which is a crucial aspect of high-performance computing systems such as cloud systems. They used a GA with crossover and mutation for scheduling, with the goal of optimising the velocity and location of particles. The study discovered that this approach significantly decreased computation expenses by making full use of VMs, evaluating fitness function values and forces within the solution space. However, load balancing and Makespan were not included as objectives. In contrast, Reference [11] presented the Best Minimum (BMin) algorithm, which focuses on load balancing and includes completion time and throughput. In their experiments, the algorithm demonstrated limited performance when dealing with 80 cloudlets, despite its efficiency improvement over the Min-Min algorithm. In addition, the authors in [12] have developed an Enhanced Round Robin (ERR) approach, which builds upon the strategy used in [11]. This method was designed to enhance efficiency while maintaining the core functionality of the traditional Round Robin (RR) technique. The efficacy of ERR was confirmed through the utilisation of the CloudSim toolbox, showcasing a decrease in the total waiting time for tasks when compared to the RR method in similar circumstances. Despite some variability in the number of cloudlets, ERR showed superior results over algorithms like ACO, GA, Min-Min, and PSO in terms of Makespan and energy consumption. Yet, in [13], the Efficient Resource Allocation with Score (ERAS) algorithm was introduced for task scheduling in cloud data centres, focusing on VMs' operational availability and employing Earliest Finish Time (EFT) for standardised scoring. ERAS, which also takes into account throughput, demonstrated superior efficiency and reliability when compared to systems that solely rely on EFT. In [14], researchers proposed a dynamic fusion mission planning method that combines GA and ant-colony techniques. This method aims to minimise energy consumption in cloud data and storage facilities. Their approach, implemented using CloudSim, showcased remarkable improvements in task completion time and energy usage. In [15], researchers studied scientific workflow applications utilising the Distributed Grey Wolf Optimiser (DGWO) algorithm. Their approach treated task scheduling as an optimisation problem, emphasising a more detailed mapping of dependent tasks and strategies to minimise Makespan. The results indicated that DGWO significantly outperformed both the Grey Wolf Optimiser (GWO) [7] and Particle Swarm Optimisation (PSO) [9] when used independently.

2.3 Literature Analysis

From the previous section, most studies ditch the employment of traditional highlighted algorithms GA, ACO, and PSO that are used to orchestrate the task scheduling process in cloud computing. As a result, the researchers turned to algorithm hybridisation.

Table 1 presents the summary of the research findings regarding hybridisation in this review. It is worth noting that the hybrid approaches yield significant outcomes in comparison to the stand-alone algorithms when considering the objective functions. In cloud computing task scheduling missions, GA ranked first place as the most used hybridisation technique.

Table 1: Summary of hybrid techniques in task scheduling

Author	Method	GA	ACO	PSO	Other
[1]	AGA	√	—	—	—
[2]	EDA-GA	√	—	—	EDA
[3]	mEDA	—	—	—	EDA
[4]	HGA-ACO	√	√	—	—
[5]	LCGA	√	—	—	—
[6]	PRTS	—	—	—	PRTS
[7]	GWO	—	—	—	MGWO
[8]	MAGA	√	—	—	MA
[10]	IDPSO	—	—	√	—
[11]	HG-GSA	√	—	—	GSA
[12]	BMin	—	—	—	BMin
[13]	ERR	—	—	—	ERR
[14]	ERAS	—	—	—	ERAS
[15]	DGWO	√	√	—	—

Table 2 provides a summary of the objectives that enhance the significance of reducing completion time (Makespan) and its impact on other objectives and the output for each method. However, it is worth noting that the Makespan has consistently been the primary objective of focus in the literature. Other objectives are also taken into account and optimised simultaneously using a hybrid multi-objective algorithm. The majority of studies primarily concentrate on hybridisation techniques, which are considered a crucial solution for reducing makespan when compared to using the combined algorithms separately. References [2,4,10] employed various algorithms by combining them, and the hybridised approaches yielded improved outcomes in their studies.

Table 2: Summary of task scheduling techniques based on objective functions

Author	Makespan	Load balancing	Resources utilisation	Resources cost	Energy
[1]	√	√	—	—	—
[2]	√	√	—	—	—
[3]	√	—	—	—	√
[4]	√	—	√	—	—
[5]	—	√	—	√	—
[6]	—	—	—	√	√
[7]	—	√	—	√	—
[8]	√	—	—	√	—
[9]	√	√	—	—	—
[10]	—	—	—	√	—

(Continued)

Table 2 (continued)

Author	Makespan	Load balancing	Resources utilisation	Resources cost	Energy
[11]	√	√	—	—	—
[12]	√	—	—	—	√
[13]	√	—	—	—	—
[14]	√	—	—	—	√
[15]	√	—	—	—	—

3 Method and Experiment

In the experimental methodology section, the performance of algorithms from [2,4,10] is compared using CloudSim Software. The study uses ten virtual machines to test both hybrid (EDA-GA, HGA-ACO, IDPSO) and standalone algorithms (GA, ACO, PSO) under varying task loads (50, 100, and 150).

3.1 Algorithm Combination Explanation

This section details the formation of the three hybrid algorithms (EDA-GA, HGA-ACO, and IDPSO) used in this study, providing a detailed explanation of how they are derived from other heuristic methods. The selection of EDA, GA, ACO, and PSO as components for our hybrid algorithms was driven by their proven effectiveness in addressing complex optimisation problems, particularly in the context of cloud computing task scheduling. Each algorithm brings unique advantages that, when combined, create a more robust and efficient hybrid method. EDA is known for its ability to model the probability distribution of promising solutions and sample new solutions based on this model, effectively capturing and exploiting the underlying structure of the problem. GA excels in global search capabilities with its operators—selection, crossover, and mutation—ensuring diversity in the solution population, preventing premature convergence, and exploring a wide range of potential solutions. ACO, inspired by the foraging behaviour of ants, is highly effective in solving combinatorial optimisation problems with its pheromone-based learning mechanism, making it suitable for scheduling tasks where the order and allocation of tasks significantly impact performance. PSO simulates the social behaviour of birds flocking or fish schooling, known for its simplicity and ability to quickly converge to optimal or near-optimal solutions, with velocity and position update mechanisms that enable efficient exploration of the solution space, making it suitable for dynamic and large-scale optimisation problems.

3.2 EDA-GA

In this Hybridisation which combines the capabilities of Estimation of Distribution Algorithm (EDA) and Genetic Algorithm (GA), is employed to address the intricate task scheduling problem in cloud computing. The hybridisation process involves several key steps that collectively contribute to its effectiveness. First, we initialise the population of solutions, representing different task-to-virtual Machine (VM) assignments. Next, we employ EDA to create probability distribution models that capture the relationships between tasks and VMs, allowing us to gain insights into the problem structure. Subsequently, we use GA's operators, including selection, crossover, and mutation, to evolve the population of solutions, ensuring diversity and exploration of the solution space. Throughout this process, we iteratively refine the assignment of tasks to VMs, aiming to minimise Makespan, achieve load balancing, and optimise resource utilisation. The hybrid EDA-GA algorithm shows potential to improve task scheduling efficiency in cloud computing environments.

The steps are as follows:

1-EDA Phase

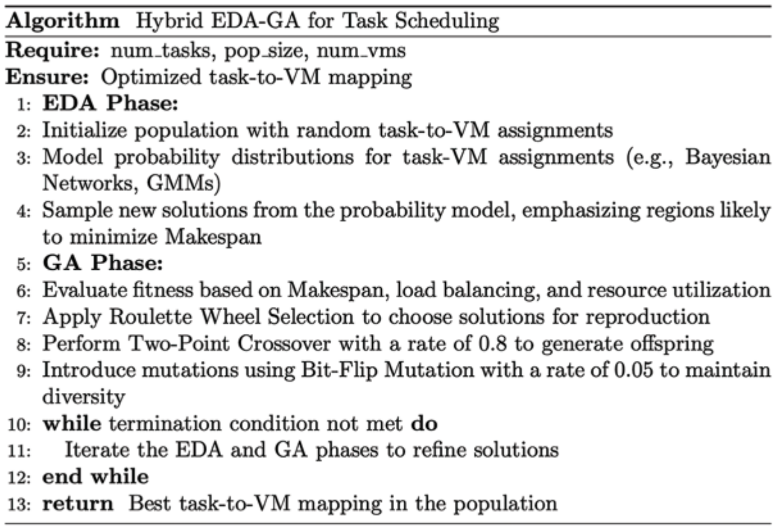
- *Initial Population*: The first population is made up of solutions in which virtual machines (VMs) are randomly allocated jobs. To simulate the initial population in our studies, we used the CloudSim simulator to create 50, 100, and 150 jobs of random sizes. Ten virtual machines (VMs) with predetermined computing capacities make up the second population.
- *Probability Distribution Modelling*: To estimate the probability distributions of task-to-VM assignments, the chosen solutions are examined. If tasks are thought of independently, this might entail creating simpler univariate distributions or multivariate distributions that describe interdependence between several activities. For example, the likelihood of allocating a given set of tasks to a certain virtual machine (VM) increases if the VM regularly produces shorter Makespans for a given set of jobs. Methods that represent the likelihood of different task-VM assignments, like as Bayesian Networks and Gaussian Mixture Models (GMMs), can be used to do this.
- *Sampling New Solutions*: The process of generating new task-to-VM allocations involves sampling from the probability distribution model. By concentrating the search on the most promising areas of the solution space, this phase takes advantage of the knowledge gathered from the previous iteration. The procedure of sampling guarantees a diversity of solutions, although with a bias towards configurations that are expected to provide lower Makespan measurements.

2-GA Phase:

- *Fitness Function*: A fitness function, in this case intended to decrease Makespan while also taking into account considerations like load balance and resource usage, is used to evaluate each solution in the population.
- *Selection Process*: By using Roulette Wheel Selection technique, solutions are probabilistically chosen according to how fit they are. According to each solution's fitness, a chance of selection is allocated. This guarantees that solutions that are more suited will be more likely to be chosen for replication.
- *Crossover*: To create offspring, a crossover operation joins pairings of chosen parent solutions. This process is essential for recombining the parent features to explore new regions of the solution space. Two-Point crossover used which represented by two parents switch the section that separates the two crossover spots they have chosen. This technique enables more intricate combinations of work assignments, which could result in more creative solutions. Crossover operations are used with a crossover rate of 0.8, indicating that 80% of the chosen pairings undergo crossover and 20% stay unmodified to maintain population variety, in order to prevent early convergence on poor solutions.
- *Mutation Operation*: Through crossover, mutation adds arbitrary alterations to the progeny. In order to preserve variety within the population and investigate areas of the solution space that crossover alone would miss, this procedure is crucial. Bit-Flip Mutation, in which the assignment is reversed with a modest probability, might be employed for binary-encoded solutions (where a job is either allocated or not assigned to a virtual machine). An equilibrium between exploitation and exploration is achieved by using a mutation rate of 0.05 (5%). In order

to keep the search process from being too random, a low mutation rate makes sure that only a tiny percentage of the population experiences mutation.

The aforementioned stages are repeated by the EDA-GA hybrid algorithm until a termination condition is satisfied, such as a certain number of generations or the point at which the Makespan improvement between generations is no longer significant. The best solution in the population is used to map tasks to virtual machines (VMs), minimizing Makespan and guaranteeing effective resource use throughout the cloud system. The details are displayed in Fig. 3 and Table 3 below.



```
Algorithm Hybrid EDA-GA for Task Scheduling
Require: num_tasks, pop_size, num_vms
Ensure: Optimized task-to-VM mapping
 1: EDA Phase:
 2: Initialize population with random task-to-VM assignments
 3: Model probability distributions for task-VM assignments (e.g., Bayesian
    Networks, GMMs)
 4: Sample new solutions from the probability model, emphasizing regions likely
    to minimize Makespan
 5: GA Phase:
 6: Evaluate fitness based on Makespan, load balancing, and resource utilization
 7: Apply Roulette Wheel Selection to choose solutions for reproduction
 8: Perform Two-Point Crossover with a rate of 0.8 to generate offspring
 9: Introduce mutations using Bit-Flip Mutation with a rate of 0.05 to maintain
    diversity
10: while termination condition not met do
11:     Iterate the EDA and GA phases to refine solutions
12: end while
13: return Best task-to-VM mapping in the population
```

Figure 3: Hybrid algorithm EDA-GA

Table 3: EDA-GA parameters settings

Algorithm	Parameter	Value
EDA	Population Size (PS)	50, 100, 150
GA	Crossover rate	0.8
GA	Mutation rate	0.05
EDA-GA	Elite population size (E)	30% of PS
EDA-GA	Learning rate (λ)	0.1

3.3 HGA-ACO

In our investigation of task scheduling optimisation in cloud computing, we introduce the Hybrid Genetic Algorithm-Ant Colony Optimisation (HGA-ACO) algorithm, a powerful hybridisation of Genetic Algorithm (GA) and Ant Colony Optimisation (ACO). This hybrid algorithm is designed to address multiple objectives, including Makespan reduction, load balancing, and efficient resource allocation. The hybridisation process involves a series of well-defined steps that enable us to harness the strengths of both GA and ACO.

Firstly, we start by creating a population of candidate solutions, each representing a potential assignment of task-to-Virtual Machines (VMs). GA is used to carry out selection, crossover, and mutation operations, enabling us to effectively explore and exploit the solution space. Meanwhile, ACO plays a crucial role in creating and updating pheromone trails that guide the search for optimal task-to-VM assignments. The pheromone trails imitate the foraging behaviour of ants, emphasising potential paths to pursue in the solution space. During the optimisation process, we continuously improve the task assignments, considering factors such as minimising Makespan, balancing the workload, and ensuring cost-effectiveness. The combination of GA and ACO allows for a comprehensive exploration and utilisation of the search space, resulting in enhanced task scheduling solutions that can fulfil a range of objectives. This hybrid HGA-ACO algorithm showcases its ability to improve task scheduling efficiency in cloud computing environments, offering a well-rounded and comprehensive approach to task assignment.

1. **Initialisation:** Initial Population: Potential task-to-VM allocations make up the initial population. When it comes to assigning duties to virtual machines, each person in this population offers a potential solution. In our trials, 10 virtual machines (VMs) with present computing capacity made up the second population, whereas 50, 100, and 150 jobs created with the CloudSim simulator made up the first population.
2. **GA Phase:** Same GA structured and used in Fig. 3 (above used).
3. **ACO Phase:**
    - *Pheromone Initialization*: During the Ant Colony Optimization (ACO) phase, pheromone levels are initialized for every possible task-to-VM assignment. The pheromone value represented as set to a small constant to indicate no prior preference for any specific assignment. This initial pheromone value represents the attractiveness of assigning a particular task to a VM, providing a baseline from which the search process begins.
    - *Ant Deployment*: Ants (agents) are deployed to construct solutions using the current pheromone levels and heuristic information such as task size and VM capacity. The decision-making process for each ant is influenced by both the pheromone strength τ and the heuristic attractiveness η. The combined influence of τ and η allows ants to explore and exploit promising regions of the solution space. The probability *Pij* of an ant assigning task *i* to VM j is typically calculated using (1):

    $$Pij = (\tau ij)\,\alpha\,(\eta ij)\,\beta / \sum k\,(\tau ik)\,\alpha\,(\eta ik)\,\beta \tag{1}$$

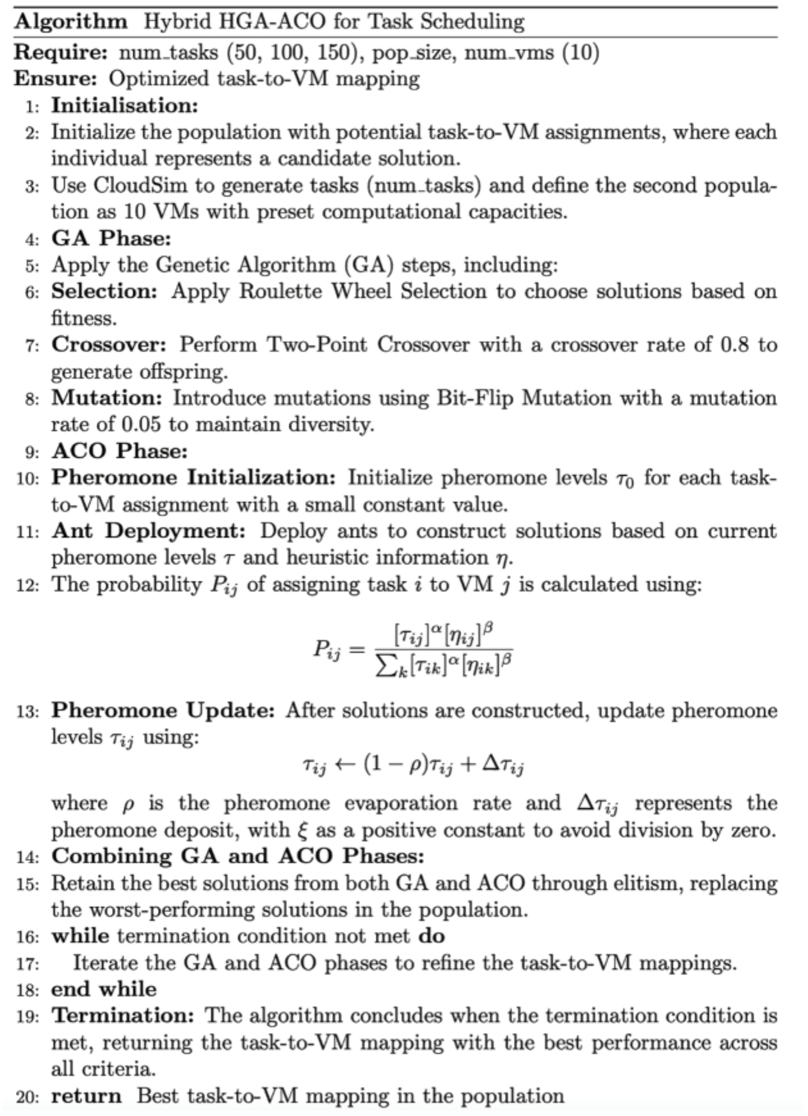
    where α and β are parameters that control the relative importance of the pheromone trail and the heuristic information.
    - *Pheromone Update*: After the ants have constructed their solutions, the pheromone levels are updated. The amount of pheromone deposited on each path is proportional to the quality of the solution, typically inversely related to the Makespan. For a path used in a higher-quality solution, more pheromone $\Delta\tau$ is deposited (2):

    $$\tau ij \leftarrow (1-\rho)\tau ij + \Delta\tau ij \tag{2}$$

    where ρ is the pheromone evaporation rate (with ρ between 0 and 1) that ensures the search remains dynamic and avoids premature convergence by gradually reducing the influence of previous pheromone levels. The positive constant ξ is applied as part of $\Delta\tau ij$ to prevent any division by zero during the update process.
4. **Combining the ACO and GA Phases:**
    - *Elitism and Iteration*: The best solutions identified by both GA and ACO are retained through elitism, replacing the worst-performing solutions in the population. This

step iterates, refining task-to-VM assignments to minimize Makespan, balance the workload, and optimize resource utilization.

Termination: The HGA-ACO algorithm continues iterating through these phases until a termination condition is met (e.g., maximum number of iterations or solution convergence). The final output is the task-to-VM mapping that achieves the best performance across all criteria. In accordance with Fig. 4, please refer to Table 4.



```
Algorithm  Hybrid HGA-ACO for Task Scheduling
Require: num_tasks (50, 100, 150), pop_size, num_vms (10)
Ensure: Optimized task-to-VM mapping
 1: Initialisation:
 2: Initialize the population with potential task-to-VM assignments, where each
    individual represents a candidate solution.
 3: Use CloudSim to generate tasks (num_tasks) and define the second popula-
    tion as 10 VMs with preset computational capacities.
 4: GA Phase:
 5: Apply the Genetic Algorithm (GA) steps, including:
 6: Selection: Apply Roulette Wheel Selection to choose solutions based on
    fitness.
 7: Crossover: Perform Two-Point Crossover with a crossover rate of 0.8 to
    generate offspring.
 8: Mutation: Introduce mutations using Bit-Flip Mutation with a mutation
    rate of 0.05 to maintain diversity.
 9: ACO Phase:
10: Pheromone Initialization: Initialize pheromone levels $\tau_0$ for each task-
    to-VM assignment with a small constant value.
11: Ant Deployment: Deploy ants to construct solutions based on current
    pheromone levels $\tau$ and heuristic information $\eta$.
12: The probability $P_{ij}$ of assigning task $i$ to VM $j$ is calculated using:
        $P_{ij} = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_k [\tau_{ik}]^\alpha [\eta_{ik}]^\beta}$
13: Pheromone Update: After solutions are constructed, update pheromone
    levels $\tau_{ij}$ using:
        $\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \Delta\tau_{ij}$
    where $\rho$ is the pheromone evaporation rate and $\Delta\tau_{ij}$ represents the
    pheromone deposit, with $\xi$ as a positive constant to avoid division by zero.
14: Combining GA and ACO Phases:
15: Retain the best solutions from both GA and ACO through elitism, replacing
    the worst-performing solutions in the population.
16: while termination condition not met do
17:     Iterate the GA and ACO phases to refine the task-to-VM mappings.
18: end while
19: Termination: The algorithm concludes when the termination condition is
    met, returning the task-to-VM mapping with the best performance across
    all criteria.
20: return Best task-to-VM mapping in the population
```

Figure 4: Hybrid algorithm HGA-ACO

Table 4: HGA-ACO parameters settings

Algorithm	Parameter	Value
HGA	Population Size (PS)	50, 100, 150
HGA	Iteration number	At max
HGA	Crossover rate	0.8
HGA	Mutation rate	0.05
ACO	Pheromone evaporation rate	$1 - \rho$
ACO	Constant (ξ)	Positive integer to avoid division by zero

3.4 ID-PSO

The Improved Discrete Particle Swarm Optimisation (IDPSO) algorithm represents a significant advancement in the field of task scheduling for cloud computing. This algorithm combines the principles of Particle Swarm Optimisation (PSO) with innovative enhancements to tackle the complex problem of task-to-Virtual Machine (VM) assignment efficiently. In this essay, we looked into the key steps and features of the IDPSO algorithm, shedding light on its significance in optimising cloud-based task scheduling. The IDPSO algorithm initiates its journey with the careful initialisation of a population of particles. Each particle corresponds to a potential task-to-VM assignment, and crucial parameters such as population size, maximum iterations, and various weighting factors are defined to guide the optimisation process. The primary objective of the IDPSO algorithm is to minimise the Makespan, which quantifies the total time taken to complete all tasks while adhering to VM availability constraints. The fitness evaluation step is a crucial component of the IDPSO algorithm. During every iteration, the fitness of each particle's assignment is carefully calculated. The fitness measure acts as a guide, directing the algorithm towards solutions that minimise the Makespan and improve task scheduling efficiency. The IDPSO algorithm also incorporates a mechanism for updating personal and global best assignments. The personal best assignment of each particle is updated when the current fitness exceeds the previous one. In a similar vein, the top assignment on a global scale is updated whenever a particle uncovers a better assignment. This encourages particles to work together and effectively explore the solution space. The velocity and position update mechanism of the IDPSO algorithm is a key feature. This step captures the core of PSO, as particles fine-tune their velocities and positions to move toward improved solutions. In this case, the algorithm carefully considers both the exploration of new assignments and the exploitation of promising ones. The particles' movement is guided by a combination of factors, including the inertia weight, cognitive factor (based on personal experience), and social factor (influenced by global knowledge). Boundary constraints are strictly enforced to ensure the integrity of task-to-VM assignments. These constraints ensure that particles do not exceed acceptable boundaries, preventing tasks from being assigned to VMs with insufficient resources. The IDPSO algorithm employs an iterative optimisation process in which particles continuously modify their assignments. This iterative nature allows the algorithm to explore a wide solution space, seeking to converge towards an optimal task assignment that minimises Makespan while respecting VM resource constraints. In the culmination of its efforts, the IDPSO algorithm yields the best task-to-VM assignment it has discovered during the optimisation process. This assignment represents the global best solution, characterised by the lowest Makespan, and is the outcome of the algorithm's rigorous exploration and refinement. The steps are as follows:

1. **Initialisation:** In the ID-PSO (Improved Dynamic Particle Swarm Optimization) algorithm, the initial population consists of particles, where each particle represents a potential task-to-VM assignment. In our experiments, we generated 50, 100, and 150 tasks of random sizes using the CloudSim simulator, representing the first population. The second population consists of 10 VMs with predefined computational capacities. The initial position and velocity of each particle are randomly initialized.
2. **PSO Phase:**
   - *Fitness Evaluation*: Each particle's fitness is evaluated based on Makespan, load balancing, and resource utilization. The fitness function is designed to minimize the Makespan while ensuring efficient resource use across all VMs.
   - *Personal Best (pBest) and Global Best (gBest) Update*:
     1. Each particle keeps track of its personal best position (pBest)–the best task-to-VM assignment it has achieved so far.
     2. The algorithm also tracks the global best position (gBest)–the best task-to-VM assignment found by any particle in the swarm.
     3. The fitness of each particle is compared against its pBest and the gBest. If a particle's current position yields a better fitness, its pBest is updated. If this pBest is better than the current gBest, the gBest is updated accordingly.
   - *Velocity and Position Update*: The velocity vi of each particle is updated based on its current velocity, the distance to its pBest, and the distance to the gBest, using the Formula (3):

     $$vi\,(t+1) = \omega\, vi\,(t) + c1\, r1\,(pBesti\ - xi\,(t)) + c2\, r2\,(gBest - xi\,(t)) \tag{3}$$

     where
     - ω is the inertia weight controlling the influence of the previous velocity.
     - $c1$ and $c2$ are acceleration coefficients representing the cognitive (pBest) and social (gBest) components.
     - $r1$ and $r2$ are random numbers uniformly distributed in the range [0, 1].
     - The position xi of each particle is then updated using its new velocity as in (4):

       $$xi\,(t+1) = xi\,(t) + vi\,(t+1) \tag{4}$$

   - *Dynamic Inertia Weight Adjustment*:
     1. The inertia weight ω is dynamically adjusted during the iterations using a sinusoidal strategy, allowing the algorithm to balance between exploration (searching new areas) and exploitation (focusing on known good areas).
     2. The inertia weight is varied as follows (5):

        $$\omega(t) = \omega max\ - (\omega max\ - \omega min\,/itermax)t \tag{5}$$

     where ωmax and ωmin are the maximum and minimum inertia weights, respectively, and $itermax$ is the maximum number of iterations.
3. **Termination**:
   - *Convergence Check*: The ID-PSO algorithm continues iterating until a termination condition is met, such as a maximum number of iterations or when the improvement in the gBest fitness value becomes negligible.
   - *Output*: The final output is the task-to-VM mapping that corresponds to the gBest position, which minimizes the Makespan while ensuring efficient load balancing and resource utilization across the cloud system.

Ultimately, the IDPSO algorithm proves to be a powerful solution to the complex problem of task scheduling in cloud computing. By combining PSO principles with discrete assignment constraints and prioritising Makespan minimisation, it achieves a careful equilibrium between individual particle exploration and utilising global knowledge. This approach results in an optimal task assignment that improves Makespan and respects resource limitations. The IDPSO algorithm represents a notable advancement in enhancing task scheduling efficiency in cloud environments, positioning it as a valuable asset in the field of cloud computing. As demonstrated in Fig. 5 and Table 5.

```
Algorithm Hybrid ID-PSO for Task Scheduling
Require: num_tasks (50, 100, 150), pop_size, num_vms (10)
Ensure: Optimized task-to-VM mapping
 1: Initialisation:
 2: Initialize the population of particles, each representing a potential task-to-
    VM assignment.
 3: Randomly initialize the position and velocity of each particle.
 4: Set the initial personal best (pBest) for each particle and the global best
    (gBest) for the entire swarm.
 5: PSO Phase:
 6: for each iteration do
 7:   for each particle do
 8:     Evaluate the fitness of the particle based on Makespan, load balancing,
        and resource utilization.
 9:     Update pBest if the current fitness is better than the particle's previous
        best.
10:     Update gBest if the current pBest is better than the global best.
11:     Update the velocity $v_i$ using:
            $v_i(t+1) = \omega \cdot v_i(t) + c_1 \cdot r_1 \cdot (pBest_i - x_i(t)) + c_2 \cdot r_2 \cdot (gBest - x_i(t))$
12:     Update the position $x_i$ using:
            $x_i(t+1) = x_i(t) + v_i(t+1)$
13:     Dynamically adjust the inertia weight $\omega$ using:
            $\omega(t) = \omega_{max} - \left(\frac{\omega_{max} - \omega_{min}}{iter_{max}}\right) \cdot t$
14:     Apply mutation with a probability of 0.05 to maintain diversity.
15:   end for
16: end for
17: Termination:
18: The algorithm terminates when the maximum number of iterations (500) is
    reached, or the improvement in gBest becomes negligible.
19: return Best task-to-VM mapping corresponding to gBest, minimizing
    Makespan while ensuring efficient resource utilization.
```

Figure 5: Hybrid algorithm ID-PSO

Table 5: ID-PSO parameters settings

Algorithm	Parameter	Value
IDPSO	Population Size (PS)	50, 100, 150
IDPSO	Iteration number	500
IDPSO	Maximum velocity	1.0
IDPSO	Mutation rate	0.05
IDPSO	Inertia weight (Dynamic)	Varies between 0.3 and 1.3 using a sinusoidal strategy

3.5 Task and VM Specifications

In this study, the task and virtual machine (VM) specifications are meticulously designed to reflect realistic cloud computing environments. The length of each task is randomly generated to introduce variability and simulate diverse computational requirements. Each task's input and output file sizes are randomly generated 100 to 300 MB, ensuring a consistent data flow that challenges the resource allocation efficiency of the algorithms. The VM parameters are configured as follows: a mirror size of 10,000 MB, a memory capacity of 512 MB, a computational speed of 100 Million Instructions Per Second (MIPS), and a bandwidth of 100 Megabits per second (Mbps). These parameters are selected to represent typical mid-range VMs used in contemporary cloud environments. Additionally, the host machine parameters include a MIPS rating of 1000, 2 GB of Memory a storage capacity of 1,000,000 MB, and a bandwidth of 10,000 Mbps, with a time-shared processor sharing mode to optimise the concurrent execution of tasks. This detailed specification ensures that the experimental setup is robust and reflective of real-world cloud infrastructure.

3.6 Objective Function Definition

The objective function is crucial for our comparative analysis, as it allows us to quantitatively evaluate the performance of the proposed scheduling algorithms. The main focus of this research is to reduce the Makespan, which refers to the total time needed to finish all scheduled tasks. The efficiency and responsiveness of cloud computing resources are directly impacted by the Makespan, which is a critical metric. The fitness function (FF) as stated in (6) is utilised to assess the quality of each solution.

$$FF = \frac{1}{Makespan} \tag{6}$$

A higher fitness value is achieved when the Makespan is lower. This formulation guarantees that the scheduling algorithm always gives priority to configurations that minimise overall completion time, thus improving the throughput and efficiency of the cloud system. Through the explicit definition and utilisation of this objective function, the study offers a clear and replicable framework for performance assessment. This framework enables rigorous comparative analysis across various scheduling strategies.

4 Results

Our experimentation results indicate a significant advantage for the hybrid models compared to the standalone algorithms, specifically in reducing Makespan. The experiment was carried out in three phases, where the number of tasks was gradually increased to 50, 100, and finally 150 tasks.

The simulation was performed using the CloudSim tool on a setup of 10 virtual machines (VMs). The simulations were conducted on a system equipped with Windows 10, 512 GB of RAM (Random Access Memory), a 2.6 GHz Core i7 CPU (Central Processing Unit), and a 500 GB hard drive. The hybrid models demonstrated exceptional performance in the context of scheduling 150 tasks, achieving Makespan values of 0.75, 0.77, and 0.64 milliseconds. In contrast, the Makespan values of 0.8, 0.84, and 0.79 milliseconds were slightly higher for the standalone algorithms. This discrepancy emphasises the hybridisation approach's effectiveness in improving task scheduling efficiency, as evidenced by the significantly shorter Makespan durations resulting from a comparable workload of 150 tasks. Fig. 6 illustrates that the hybrid models consistently outperformed the standalone algorithms in all phases of the experiment, thereby highlighting their efficiency and robustness in a variety of task scheduling scenarios.

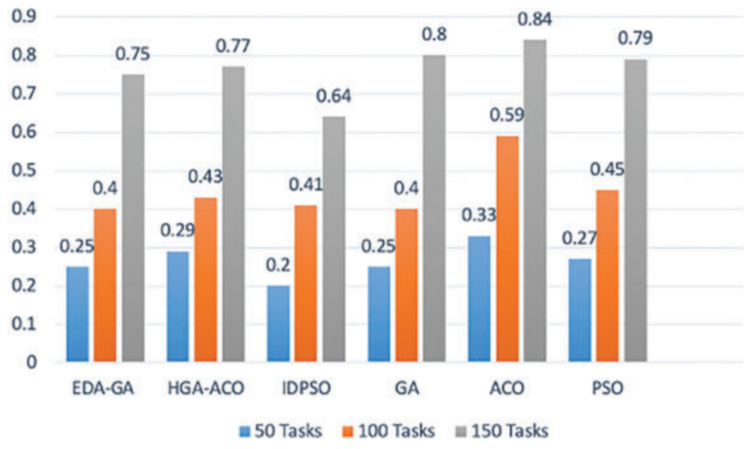


Figure 6: Comparison of Makespan reduction between hybrid and stand-alone algorithms

According to the findings presented, IDPSO multi-objective optimisation emerged as the most effective combination for task scheduling in cloud computing. This hybridisation strategy was designed to minimise makespan, enhance convergence, and improve load balancing throughout the scheduling and execution processes, particularly when dealing with a large number of tasks. The results indicate a noteworthy trend: as the number of input tasks increases, the overall completion time decreases, leading to improved load balancing. This promising outcome suggests the feasibility of implementing this approach in real-world scenarios, where optimising task scheduling efficiency is paramount.

5 Discussion and Challenges

The previous discussion clearly shows that researchers have used a wide range of algorithms to achieve their goals. These objectives can be accomplished by strategically applying hybridisation techniques, which involve combining multiple methods to achieve the desired performance levels. Table 1 presents a thorough analysis, highlighting the genetic algorithm as a frontrunner. This is primarily attributed to its key components, Crossover and Mutation. These components collaborate to create a large-scale population and practical solutions, effectively tackling the challenges of task scheduling in cloud computing. Nonetheless, it is prudent to conduct additional research into the specific goals that should be prioritised in the domain of cloud computing task scheduling. By tailoring objectives to the research focus, researchers can make more informed decisions about which aspects to highlight. Table 2 highlights that completion time (Makespan) is the primary objective in the vast majority of reviewed articles. Other objectives are important, but they are inextricably linked to and dependent on Makespan.

There are two significant challenges that arise in the field of multi-objective scheduling optimisation.

5.1 Mapping

The primary obstacle in multi-objective task scheduling in cloud computing is the development of mapping solutions that minimise Makespan, maintain load balancing, consider resource utilisation and cost-effectiveness, and integrate green energy data centres. A complex endeavour is the discovery of the optimal balance among these factors.

5.2 Integration

Future research must investigate the integration of task scheduling with virtual machine consolidation methodologies to improve the efficiency of task scheduling. To augment conventional scheduling strategies in cloud computing, it is imperative to develop innovative methods, including economic models, heuristic algorithms, and nature-inspired algorithms.

Notably, there exists a viable opportunity to devise sophisticated job scheduling techniques in cloud computing environments by integrating multiple methodologies and considering various input criteria, including costs and energy consumption. Additionally, the convergence of Time-Sensitive Networking (TSN) principles can play a pivotal role in ensuring timely and deterministic communication within cloud-based systems [20,21]. Further research should explore both single-and multi-objective task scheduling, using a combination of existing methods to drive further improvements. This comprehensive approach shows potential for advancing the field of cloud computing task scheduling. By drawing inspiration from various interdisciplinary work and the principles of communication progression, researchers in cloud computing can discover new methods to improve efficiency and resource allocation in cloud-based systems.

6 Conclusion

In conclusion, researchers have extensively investigated the complex challenge of task scheduling in cloud computing. It is now firmly established as a non-deterministic polynomial (NP-Complete) issue characterised by stochastic behaviour. A multitude of methods and techniques have been proposed to address these issues, taking into account various factors impacting service providers and consumers. This paper has provided a comprehensive overview of task scheduling challenges, classifying them as NP-Complete difficulties based on an in-depth survey of prior research. Through an exhaustive review of established literature, we have identified effective methodologies for achieving optimal implementations, with optimisation algorithms emerging as key solutions. Furthermore, this research has heralded a new era of techniques by combining existing algorithms and refining operations and procedures to meet diverse objectives. These advancements have the potential to eliminate bottlenecks in cloud computing by taking into account the interests of both users and service providers. The IDPSO hybrid approach is the most promising solution in our assessment of pertinent studies, particularly in scenarios that involve a high volume of assigned tasks, while also ensuring effective load balancing within the dynamic landscape of cloud computing. This is a substantial stride toward resolving the intricate task scheduling issues that exist in the cloud and represents the continuous development of this essential field.

Acknowledgement: The author acknowledges the contribution and support of the Faculty of Computer Science and Information Technology at University Putra Malaysia (UPM).

Funding Statement: The author received no specific funding for this study.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author, Abdulrahman M. Abdulghani, upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The author declares that they have no conflicts of interest to report regarding the present study.

References

[1] X. Fu, Y. Sun, H. Wang, and H. Li, "Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm," *Cluster Comput.*, vol. 26, pp. 2479–2488, Oct. 2023. doi: 10.1007/s10586-020-03221-z.

[2] S. Pang, W. Li, H. He, Z. Shan, and X. Wang, "An EDA-GA hybrid algorithm for multi-objective task scheduling in cloud computing," *IEEE Access*, vol. 7, pp. 146379–146389, 2019. doi: 10.1109/ACCESS.2019.2946216.

[3] C. Wu and L. Wang, "A multi-model estimation of distribution algorithm for energy efficient scheduling under cloud computing system," *J. Parallel Distr. Comput.*, vol. 117, pp. 63–72, 2018. doi: 10.1016/j.jpdc.2018.02.009.

[4] A. S. Kumar and M. Venkatesan, "Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment," *Wirel. Pers. Commun.*, vol. 107, no. 4, pp. 1835–1848, 2019. doi: 10.1007/s11277-019-06360-8.

[5] C. Chandrashekar, P. Krishnadoss, V. K. Poornachary, B. Ananthakrishnan, and K. Rangasamy, "HWACOA scheduler: Hybrid weighted ant colony optimization algorithm for task scheduling in cloud computing," *Appl. Sci.*, vol. 13, no. 6, Mar. 8, 2023, Art. no. 3433. doi: 10.3390/app13063433.

[6] S. Yin, P. Ke, and L. Tao, "An improved genetic algorithm for task scheduling in cloud computing," in *13th IEEE Conf. Ind. Electron. Appl. (ICIEA)*, Wuhan, China, 2018, pp. 526–530.

[7] P. Zhu, J. Chen, and Y. Fu, "A power-aware scheduling algorithm for real-time workflow applications in clouds," in *3rd Int. Conf. Electron. Inform. Technol. Comput. Eng. (EITCE)*, Xiamen, China, 2019, pp. 1870–1873.

[8] A. Alzaqebah, R. Al-Sayyed, and R. Masadeh, "Task scheduling based on modified grey wolf optimizer in cloud computing environment," in *2019 2nd Int. Conf. New Trends Comput. Sci. (ICTCS)*, Amman, Jordan, 2019, pp. 1–6.

[9] A. Alsmady, T. Al-Khraishi, W. Mardini, H. Alazzam, and Y. Khamayseh, "Workflow scheduling in cloud computing using memetic algorithm," in *2019 IEEE Jordan Int. Joint Conf. Electric. Eng. Inform. Technol. (JEEIT)*, Amman, Jordan, 2019, pp. 302–306.

[10] S. Liu and Y. Yin, "Task scheduling in cloud computing based on improved discrete particle swarm optimization," in *2019 2nd Int. Conf. Inform. Syst. Comput. Aided Edu. (ICISCAE)*, Dalian, China, 2019, pp. 594–597.

[11] D. Chaudhary and B. Kumar, "Cost optimized hybrid genetic-gravitational search algorithm for load scheduling in cloud computing," *Appl. Soft Comput.*, vol. 83, 2019, Art. no. 105627. doi: 10.1016/j.asoc.2019.105627.

[12] Y. Shi, K. Suo, S. Kemp, and J. Hodge, "A task scheduling approach for cloud resource management," in *2020 Fourth World Conf. Smart Trends Syst., Secur. Sustainability (WorldS4)*, London, UK, 2020, pp. 131–136.

[13] M. S. Sanaj and P. M. Joe Prathap, “An Enhanced Round Robin (ERR) algorithm for effective and efficient task scheduling in cloud environment,” in *2020 Adv. Comput. Commun. Technol. High Perform. Appl. (ACCTHPA)*, Cochin, India, 2020, pp. 107–110.

[14] V. A. Lepakshi and C. S. R. Prashanth, “Efficient resource allocation with score for reliable task scheduling in cloud computing systems,” in *2nd Int. Conf. Innov. Mech. Ind. Appl. (ICIMIA)*, Bangalore, India, 2020, pp. 6–12.

[15] Z. Zong, “An improvement of task scheduling algorithms for green cloud computing,” in *15th Int. Conf. Comput. Sci. Edu. (ICCSE)*, Delft, Netherlands, 2020, pp. 654–657.

[16] B. Abed-alguni and N. Alawad, “Distributed grey wolf optimizer for scheduling of workflow applications in cloud environments,” *Appl. Soft Comput.*, vol. 102, 2021, Art. no. 107113. doi: 10.1016/j.asoc.2021.107113.

[17] A. Arunarani, D. Manjula, and V. Sugumaran, “Task scheduling techniques in cloud computing: A literature survey,” *Future Gener. Comput. Syst.*, vol. 91, pp. 407–415, 2019. doi: 10.1016/j.future.2018.09.014.

[18] M. Kumar and S. Sharma, “PSO-COGENT: Cost and energy efficient scheduling in cloud environment with deadline constraint,” *Sustainable Comput.: Inform. Syst.*, vol. 19, pp. 147–164, 2018.

[19] M. A. Tawfeek, A. El-Sisi, A. E. Keshk, and F. A. Torkey, “Cloud task scheduling based on ant colony optimization,” in *8th Int. Conf. Comput. Eng. Syst. (ICCES)*, Cairo, Egypt, 2013, pp. 64–69.

[20] B. O. Akram, N. K. Noordin, F. Hashim, M. F. A. Rasid, M. I. Salman and A. M. Abdulghani, “Joint scheduling and routing optimization for deterministic hybrid traffic in time-sensitive networks using constraint programming,” *IEEE Access*, vol. 11, pp. 142764–142779, 2023.

[21] B. O. Akram, N. K. Noordin, F. Hashim, M. A. F. Rasid, M. I. Salman and A. M. Abdulghani, “Enhancing reliability of time-triggered traffic in joint scheduling and routing optimization within time-sensitive networks,” *IEEE Access*, vol. 12, pp. 78379–78396, 2024. doi: 10.1109/ACCESS.2024.3408923.