Article history:
Received 09 June 2025
Revised 04 September 2025
Accepted 13 September 2025
Published online 30 September 2025

Journal of Resource Management and Decision Engineering

Volume 4, Issue 3, pp 1-14

Providing a Solution for Processing Heterogeneous Tasks in Cloud Computing Using Distributed Resource Allocation

Mohammad Hadi. Dadizadeh Dargiri[1*]

[1] MA student, Software Department, Iran University of Science and Technology, Tehran, Iran

*** Corresponding author email address**: mhdadizadeh@gmail.com

A r t i c l e I n f o

Article type:
Original Research

How to cite this article:
Dadizadeh Dargiri, M. H. (2025). Providing a Solution for Processing Heterogeneous Tasks in Cloud Computing Using Distributed Resource Allocation. *Journal of Resource Management and Decision Engineering, 4*(3), 1-14.
https://doi.org/10.61838/kman.jrmde.145

A B S T R A C T

In this study, a novel approach is presented for processing heterogeneous tasks in Cloud Computing environments by leveraging optimal distributed resource allocation. The primary objective is to enhance processing efficiency and achieve effective resource utilization in conditions where tasks have diverse characteristics, varying data volumes, and different computational requirements. The proposed method models tasks and resources as a directed acyclic graph and employs a multi-objective optimization algorithm to perform resource allocation in a way that not only reduces the overall processing time but also ensures load balancing among resources. This approach, by incorporating priority queues and execution time analysis for each subtask, enables the selection of the most appropriate resource for each task. The simulation results indicate that the proposed method achieves significant improvements over conventional algorithms in reducing the overall job completion time, increasing resource utilization rates, and enhancing the quality of service in heterogeneous task processing.

Keywords: *Cloud Computing, heterogeneous task processing, distributed resource allocation, Multi-objective optimization, Load balancing*

1. Introduction

Cloud Computing has emerged as a transformative paradigm in contemporary computing, offering on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage, and applications that can be rapidly provisioned and released with minimal management effort or service provider interaction (Mell & Grance, 2011). This shift toward virtualized and elastic infrastructures has significantly changed the landscape of information technology management, enabling organizations to focus on service delivery rather than the procurement and maintenance of hardware. As enterprises, governments, and academic institutions increasingly migrate their workloads to cloud platforms, the effective scheduling of tasks and allocation of

resources within heterogeneous and distributed cloud environments has become a critical research challenge (Tsai & Rodrigues, 2014). In particular, the diversity of computational tasks, variability in data volumes, and differences in processing demands require advanced scheduling strategies that balance multiple objectives, including minimizing makespan, maximizing resource utilization, and improving load balancing (Fard et al., 2012).

Task scheduling in heterogeneous cloud environments is inherently complex because it involves assigning interdependent tasks to resources with varying capabilities while respecting precedence constraints and optimizing several competing objectives (Khorsand & Sharifi, 2014). Early studies introduced heuristic and metaheuristic approaches to address this problem, yet the growing scale and diversity of cloud workloads have rendered these conventional methods insufficient (Panda et al., 2016). For example, static allocation strategies often fail to adapt to fluctuating workloads and dynamic changes in resource availability, leading to inefficient utilization and prolonged task completion times (Doe & Smith, 2024). To overcome these limitations, recent works have proposed hybrid and adaptive optimization algorithms that incorporate multi-objective decision-making, dynamic load balancing, and predictive modeling (Pham & Huh, 2016). These approaches aim to achieve optimal trade-offs among competing performance criteria while ensuring system stability and robustness.

A major line of research has focused on metaheuristic scheduling techniques, which have demonstrated remarkable effectiveness in optimizing complex, large-scale scheduling problems in heterogeneous systems (Tsai & Rodrigues, 2014). Notably, multi-objective list scheduling approaches have been introduced to handle workflow applications by considering both execution costs and inter-task communication overheads (Arabnejad & Barbosa, 2014). Such methods often leverage optimistic cost tables and task-level priority analysis to improve scheduling accuracy and reduce makespan. Similarly, hybrid heuristic algorithms have been developed to combine the strengths of different optimization methods, leading to enhanced performance in dynamic and unpredictable environments (Khorsand & Sharifi, 2014). These advances have laid a foundation for the development of evolutionary multi-objective optimization algorithms that can simultaneously explore diverse solution spaces and converge rapidly toward Pareto-optimal fronts (Deb & Jain, 2014).

Despite these achievements, significant challenges remain. The inherent heterogeneity of cloud infrastructures—comprising physical machines (PMs) and virtual machines (VMs) with different processing speeds, memory capacities, and network bandwidths—complicates the efficient mapping of tasks to resources (Li et al., 2011). Furthermore, many existing methods do not adequately address the dynamic nature of cloud workloads, where incoming tasks may vary widely in their computational requirements, data dependencies, and execution deadlines (Abrishami et al., 2013). This often results in suboptimal resource utilization and workload imbalances, which can degrade overall system performance. As such, researchers have increasingly turned to adaptive and learning-based approaches to resource allocation, incorporating real-time feedback and prediction mechanisms to enhance scheduling decisions (Kim & Lee, 2024).

The integration of machine learning into resource allocation frameworks represents a promising direction for improving cloud scheduling efficiency. For example, reinforcement learning has been applied to dynamically allocate resources in hybrid cloud environments, enabling systems to learn optimal allocation policies from ongoing interactions with their environments (Kim & Lee, 2024). Similarly, intelligent optimization techniques have been developed to analyze workload patterns and predict resource demands, facilitating proactive and context-aware scheduling (Wang & Rahman, 2025). These advancements complement traditional metaheuristic methods by introducing adaptability and foresight into the decision-making process, thus addressing the volatility and complexity of real-world cloud workloads. Moreover, combining multi-objective optimization algorithms with learning-based models can further enhance their ability to achieve balanced trade-offs among conflicting objectives, such as execution time, cost, and energy consumption (Sharma, 2023).

The growing adoption of cloud computing across various sectors underscores the practical importance of addressing these scheduling challenges. In higher education, for instance, cloud platforms are increasingly used to deliver scalable and cost-effective computing resources, yet their successful adoption requires robust frameworks for resource allocation and workload management (Alqatan et al., 2025). Small entrepreneurial businesses, particularly in niche domains like handicrafts, have also begun leveraging cloud computing to support their digital transformation, necessitating models that can handle their unique workload

characteristics and resource constraints (Azami et al., 2024). In the financial domain, cloud computing combined with big data analytics is driving the modernization of information systems, demanding efficient scheduling mechanisms to process massive and time-sensitive datasets (Sharma, 2023). Even public services such as sports governance are adopting cloud-based solutions to enhance innovation and service delivery, further amplifying the need for scalable and efficient scheduling algorithms (Hao et al., 2023).

Cloud computing has also become intertwined with emerging technologies such as Fog computing, Internet of Things (IoT), and Artificial intelligence (AI), creating even more complex scheduling scenarios. The convergence of fog-cloud architectures with IoT-based autonomous systems, for instance, requires seamless coordination of distributed resources and real-time task execution at the network edge (Singh, 2024). In financial technology (fintech) applications, the combination of AI algorithms and cloud infrastructures has led to blockchain-enabled systems that demand extremely high processing throughput and reliability (Lăzăroiu, 2023). These evolving contexts introduce additional constraints and performance objectives into cloud scheduling problems, reinforcing the necessity for advanced multi-objective optimization approaches. As the scope of cloud computing continues to expand, incorporating emerging paradigms like Quantum computing through platforms such as the IBM Quantum Experience further complicates scheduling, as hybrid classical-quantum workflows introduce entirely new resource allocation dynamics (IBM Quantum Experience, 2023).

Regional studies have further highlighted the socio-technical implications of cloud adoption, revealing the importance of culturally and contextually tailored scheduling and resource management strategies. For example, the use of cloud computing has been shown to enhance the quality of accounting information and influence the development of international financial reporting standards in Jordanian corporations, underscoring the need for reliable and transparent scheduling frameworks (Kmaleh, 2023). Similarly, efforts to improve Arabic content delivery in cloud-based e-learning environments in Jordan have emphasized the necessity for efficient resource allocation to ensure accessibility and performance (Khasawneh et al., 2023). In the banking sector, the application of cloud computing to human resource management has demonstrated its potential to improve organizational efficiency, but only when supported by robust and adaptive workload scheduling mechanisms (Doldi et al., 2023). These cases illustrate that task scheduling is not merely a technical problem but a critical enabler of organizational transformation and service quality.

Building on this background, the present study proposes a novel multi-objective optimization approach based on the Capuchin Search Algorithm to enhance task scheduling in heterogeneous cloud environments. This method seeks to overcome the limitations of existing techniques by simultaneously optimizing three key objectives—minimizing makespan, maximizing resource utilization, and improving load balancing—while respecting task precedence constraints and adapting to dynamic workload conditions. By integrating advanced scheduling mechanisms, such as physical-machine selection based on minimum completion time and subtask allocation using the Earliest Finish Time (EFT) policy, this approach aims to achieve a more balanced and efficient utilization of cloud resources. Ultimately, this study contributes to the ongoing evolution of cloud computing by offering a comprehensive and adaptive scheduling framework that can meet the demands of increasingly heterogeneous and dynamic cloud environments.

2. Methods and Materials

For processing heterogeneous tasks in a Cloud Computing environment using distributed resource allocation, the main goal is to design a model that, while satisfying task dependency constraints, can minimize the total completion time (makespan), maximize resource utilization, and maintain load balancing.

In this model, a set of heterogeneous computing resources is considered, including physical machines (PMs) and virtual machines (VMs), each with different processing power, memory capacity, and communication bandwidth.

We assume that the cloud computing system includes a set of physical machines as shown in Equation (1):

(1) $C = \{pm1, pm2, \ldots, pmm\}$

Each physical machine *pmi* can host several virtual machines as shown in Equation (2):

(2) $VMj = \{vm1, vm2, \ldots, vmn\}$

The set of tasks submitted by users is defined as in Equation (3):

(3) $Ut = \{T1, T2, \ldots, Tk\}$

Each task *Ti* can be divided into one or more subtasks and executed on different resources. Each physical machine can host several virtual machines, and each virtual machine can execute several tasks or subtasks. Task allocation to

resources must preserve task precedence constraints and avoid execution conflicts.

First Objective Function: Minimizing the Makespan

The makespan is defined as:

(3) Makespan = max{ CT(Rj) } $1 \leq j \leq m$

where CT(Rj) is the completion time of all tasks allocated to resource Rj. The objective is:

min (Makespan)

Second Objective Function: Maximizing Resource Utilization

Resource utilization is defined as the ratio of the effective processing execution time to the total available time of the resources:

(4) $RU = \Sigma ET(Ti,Rj) / \Sigma AvailTime(Rj)$

where ET(Ti,Rj) is the execution time of task *Ti* on resource *Rj*, and AvailTime(Rj) is the total available time of resource *Rj*. The objective is:

max (RU)

Third Objective Function: Improving Load Balancing

To prevent idleness of some resources and overload on others, the load balancing (LB) index is defined as follows:

(5) $LB = \sqrt{(\Sigma (Loadj - Loadavg)^2 / m)}$

where *Loadj* is the computational load of resource *j*, and *Loadavg* is the average load across all resources. The objective is:

(6) min (LB)

Model Constraints

1. Each task must be assigned to one and only one computing resource:

(7) $\Sigma x(i,j) = 1$ for each task i

2. Task precedence constraints must be satisfied.
3. Scheduling must not violate the processing and memory capacities of the resources.

This tri-objective model is solved using a Multi-objective optimization algorithm (such as MOCapSA), which, by considering priority queues and execution time analysis for each subtask, provides the optimal resource allocation for processing heterogeneous tasks in the Cloud Computing environment.

Figure 1

Task graph diagram with 11 subtasks

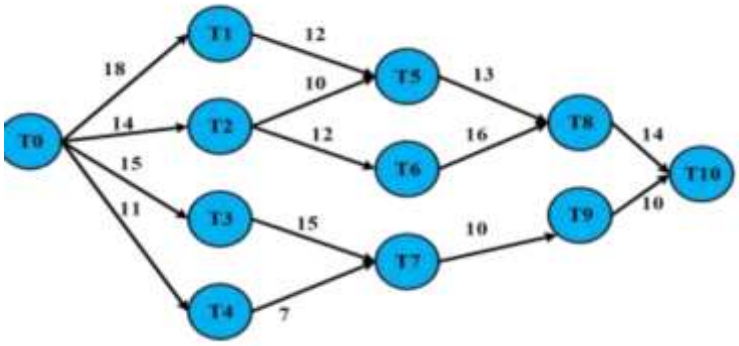
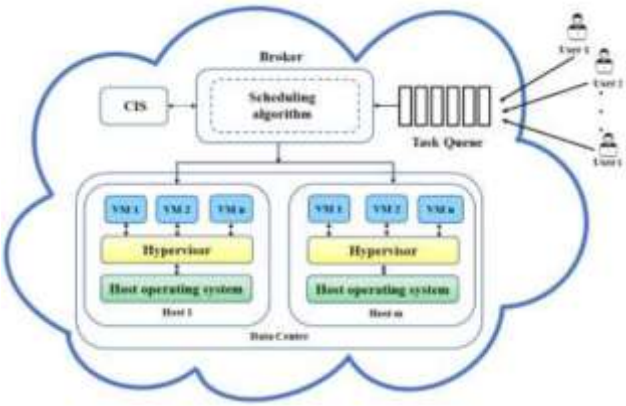


Figure 2

Task scheduling architecture in a Cloud Computing environment



2-3 Nondominated Sorting in the Ranked Algorithm

In the Multi-objective optimization process for processing heterogeneous tasks in Cloud Computing environments, the quality of each solution is evaluated based on multiple simultaneous criteria (such as minimizing makespan, maximizing resource utilization, and improving load balancing). Since these objectives may sometimes conflict with one another, it is necessary to identify a set of optimal solutions where none dominates the others completely. This set is known as the Pareto front and is identified through Nondominated sorting.

In this method, each solution is considered as a point in the multi-dimensional objective space. Solution A dominates solution B if and only if: A is better than or equal to B in all objectives, and A is strictly better than B in at least one objective. Solutions that are not dominated by any other solution are ranked first and form the first Pareto front. These solutions are then removed from the set, and the same process is repeated to find subsequent Pareto fronts.

In the context of distributed resource allocation for heterogeneous tasks, this sorting enables the algorithm to discover a set of efficient allocations that balance multiple objectives rather than focusing on only one objective (such as reducing makespan). Consequently, the scheduling system can select the best option from the Pareto front solutions depending on workload conditions or user priorities.

To improve the efficiency of nondominated sorting in this problem, data structures such as task adjacency lists and resource dependency matrices are used to reduce computational costs during solution comparison. Additionally, in the proposed model, task allocation to resources is performed based on priority queues, ensuring that higher-priority tasks are placed earlier in the Pareto fronts and receive faster resource allocation.

In Multi-objective optimization problems such as resource allocation for processing heterogeneous tasks in Cloud Computing environments, merely obtaining a set of Pareto front solutions is not sufficient. These solutions must also be diverse and well-distributed across the front so that the user or system can make optimal selections under various conditions.

One widely used approach to maintain this diversity is the Crowding distance. The crowding distance for each solution measures the density of other solutions in its neighborhood in the objective space. Solutions with higher crowding distances are located in sparser regions of the Pareto front and help maintain the spread of the solution set.

This metric is used for secondary sorting in ranked algorithms such as NSGA-II, where solutions within the same Pareto front are first sorted by crowding distance, and those with larger distances are prioritized for selection.

In the context of resource allocation in Cloud Computing, this approach helps ensure that the obtained solutions are not concentrated only on a narrow region of the objective space (e.g., only minimum makespan) but also provide trade-off options that balance time, cost, and load balancing.

The calculation of crowding distance is typically done as follows:

1. For each objective, sort the solutions based on the values of that objective.
2. Assign an infinite value to the boundary solutions to ensure they are always preserved.
3. The crowding distance of each intermediate solution is equal to the sum of the normalized differences of the objective values between its left and right neighbors.

This method ensures that even in scenarios where cloud resources are highly heterogeneous and task dependency constraints are complex, a diverse set of solutions is available, each offering a different balance among the

optimization criteria. As a result, the resource management system can select the best solution from among diverse available options based on real-world conditions.

Capuchin Search Algorithm (CapSA)

The Capuchin Search Algorithm (CapSA) is a novel metaheuristic method inspired by the social behavior and foraging strategies of Capuchin monkeys. In their natural environment, these monkeys efficiently search for food by leveraging group cooperation, task division, and the use of diverse resources. The main idea of this algorithm is to simulate these behaviors to achieve optimal search in the solution space of complex problems.

In the context of processing heterogeneous tasks in Cloud Computing, the Capuchin Search Algorithm can act as a multi-objective optimizer and allocate resources in a way that simultaneously achieves objectives such as minimizing makespan, maximizing resource utilization, and improving load balancing.

The general process of the algorithm involves the following steps:

1. **Population initialization**: A number of initial solutions (capuchin positions) are randomly generated in the search space. Each position represents a resource allocation to the task set and is encoded using an n-dimensional vector model (each dimension corresponds to a task, and its value represents the assigned resource).
2. **Solution evaluation**: Each solution is evaluated using the defined objective functions (makespan, resource utilization, load balancing). At this stage, positions with better performance are ranked higher.
3. **Local search**: A portion of the population is improved through small changes in task allocation (such as swapping tasks between resources or changing execution priorities). This stage simulates the capuchins' behavior of carefully examining their surroundings to find better food sources.
4. **Global search**: Other members of the population explore using larger movements in the search space (such as major changes in resource allocations) to reduce the likelihood of being trapped in local optima. This behavior is similar to capuchins migrating to new areas to find resources.
5. **Position update**: The positions of the capuchins are updated based on a combination of the current best positions and controlled random movements. This combination maintains a balance between exploration and exploitation.
6. **Selection and survival mechanism**: After each iteration, solutions are ranked based on Nondominated sorting and Crowding distance. The best solutions are selected to form the next generation population so that the optimization process continues.
7. **Stopping criterion**: The algorithm continues until a specified number of iterations is reached or no significant improvement is observed in the best solution. Finally, a set of Pareto-optimal solutions is presented as the output.

Using the Capuchin Search Algorithm in this problem enables resource allocation to tasks in a heterogeneous cloud environment to be optimized by considering priorities, dependencies, and communication costs. The main feature of this algorithm is its ability to balance solution diversity and convergence speed toward high-quality solutions, which is a significant advantage in complex problems with large search spaces.

The proposed method for resource allocation and processing of heterogeneous tasks in Cloud Computing environments is designed based on the Capuchin Search Algorithm, a Multi-objective optimization approach. This method simultaneously optimizes three objective functions: minimizing makespan, maximizing resource utilization, and improving load balancing. To achieve these objectives, the optimization process is carried out in four main phases:

Phase 1 – Population Initialization:

In this phase, a set of initial solutions is randomly generated. Each solution is represented as an n-dimensional vector, where each dimension corresponds to a task and its value indicates the assigned resource. At this stage, task precedence constraints are enforced to eliminate invalid allocations.

Phase 2 – Solution Evaluation:

In this phase, each solution is evaluated according to the following metrics:

- Makespan: The maximum completion time among all resources.
- RU: The ratio of the total execution time of tasks to the total available time of resources.
- LB: The deviation of each resource's computational load from the average load.

Phase 3 – Nondominated Sorting and Diversity Preservation:

Solutions are ranked using Nondominated sorting and then sorted based on Crowding distance to preserve both quality and diversity of the Pareto front set. This ensures that the solutions are not focused on a single objective and instead offer a wide range of optimizations.

Phase 4 – Population Search and Update:

The Capuchin Search Algorithm improves resource allocation by simulating local and global search movements of Capuchin monkeys. In this phase:

i. Local search is applied to make small modifications in allocations (such as changing the resource of a task).

ii. Global search explores new areas in the search space (such as group reassignment of tasks across resources).

iii. After these updates, the new population is re-evaluated and enters the next cycle.

Pseudocode of the Proposed Method:

Input: Set of tasks U_t, set of resources C, algorithm parameters

Output: Final Pareto-optimal solution set

1. Initialize population P with size N
2. For each solution in P:
    2.1 Check and adjust task precedence constraints
    2.2 Evaluate using objective functions: Makespan,
3. RU, LB
4. Perform Nondominated Sorting on P
5. Calculate Crowding Distance for each solution

While (termination condition not met):

    5.1 Select parents based on rank and crowding distance
    5.2 Apply Local Search to part of the population
    5.3 Apply Global Search to another part of the population
    5.4 Combine old and new populations → P'
    5.5 Perform Nondominated Sorting on P'
    5.6 Calculate Crowding Distance
    5.7 Select the best N solutions for the next generation

6. Output the final Pareto front

Task Prioritization (Ranking) for User Tasks

In the proposed method for processing heterogeneous tasks in Cloud Computing with distributed resource allocation, prioritizing user tasks plays a critical role in improving system efficiency and reducing overall completion time. The prioritization process is based on a Directed acyclic graph (DAG) model, where task precedence dependencies are specified, and each task is ranked according to its position in the graph and its significance for completing the entire workflow.

Factors such as the expected execution time of each task on different resources, input and output data size, number of dependent tasks, and the criticality of the task in the main graph paths are considered. Tasks on the critical path or tasks that are prerequisites for many other tasks are given higher priority. Communication costs between tasks and resources are also considered—tasks requiring less data transfer or faster communication are more likely to be given higher priority.

After priority values are calculated for each task, priority queues are used for scheduling so that at any time, the highest-priority ready task (whose prerequisites are completed) is allocated to the most suitable resource. This mechanism ensures optimal resource usage, faster execution of critical tasks, and balance between reducing makespan and efficient resource utilization.

In the proposed method for processing heterogeneous tasks in Cloud Computing with distributed resource allocation, selecting the most suitable physical machine for executing each task is crucial because resource heterogeneity leads to variations in execution time, cost, and utilization across different resources. The selection process is designed to choose the physical machine that offers the minimum completion time and highest efficiency for each task while respecting resource capacity and task dependency constraints.

First, for each task Ti, its execution time on each physical machine PMj is calculated. This time includes both processing time and communication cost between dependent tasks. Communication cost is considered zero if two tasks run on the same machine; otherwise, it is calculated based on data volume and bandwidth between resources.

Execution time is estimated as follows:

(8) $ETi,j = (Wi / Sj) + Commi,j$

where:

- ETi,j is the execution time of task Ti on machine PMj
- Wi is the computational workload of task Ti
- Sj is the processing speed of machine PMj
- $Commi,j$ is the communication cost of task Ti with other dependent tasks when executed on PMj

Then, the best physical machine for each task is selected based on the minimum completion time criterion:

(9) $PM^*Ti = argmin_{PMj \in C} ETi,j$

where PM*Ti is the best physical machine for task Ti and C is the set of all physical machines.

This selection mechanism ensures that task-to-resource assignment aligns with the system's processing and communication characteristics, ultimately reducing

makespan, increasing resource utilization, and achieving more effective load balancing.

In the proposed method for processing heterogeneous tasks in Cloud Computing with distributed resource allocation, after selecting the most suitable physical machine for each task, its subtasks must be scheduled and assigned to processors within the virtual machines hosted on that physical machine.

For this purpose, the Earliest Finish Time (EFT) policy, specifically designed for heterogeneous systems, is applied.

In this policy, the goal is to assign each subtask to the processor that results in the minimum finish time. The finish time is calculated as the sum of the processor's ready time, the execution time of the subtask on that processor, and the communication costs arising from dependencies between subtasks.

For each subtask STk on processor VMm within the selected physical machine, the finish time is estimated as follows:

(10) $EFT_{k,m} = ReadyTime_m + (Wk / SmS) + Comm_{k,m}$

where:

• $EFT_{k,m}$ is the earliest finish time of subtask STk on processor VMm

• $ReadyTime_m$ is the ready time of processor VMm to start a new task

• Wk is the computational workload of subtask STk

• SmS is the processing speed of processor VMm

• $Comm_{k,m}$ is the communication cost between this subtask and its parent subtasks when executed on VMm

Then, the processor with the minimum $EFT_{k,m}$ is selected:

(11) $VM^*STk = \text{argmin}_{VMm \in M(PMj)} EFT_{k,m}$

where M(PMj) is the set of processors within virtual machines hosted on physical machine PMj.

The advantage of using this policy in a heterogeneous cloud environment is that:

1. Workload is balanced among processors, as processors that become free earlier are prioritized for receiving new tasks.
2. Makespan is reduced, as each subtask is executed on the fastest available processor.
3. Communication costs between subtasks are reduced, especially when dependencies are kept within the same virtual or physical machine.

3. Findings and Results

To evaluate the performance of the proposed method, a set of experiments was conducted in a simulated heterogeneous cloud environment consisting of multiple physical machines and virtual machines. The evaluation criteria included makespan, resource utilization (RU), and load balance (LB). The proposed method, based on the Capuchin multi-objective optimization algorithm, was compared with two well-known algorithms, NSGA-II and MOPSO. Table 1 presents the results obtained from running these three algorithms on a set of 50 heterogeneous tasks.

Table 1

Performance Comparison of the Proposed Algorithm and Baseline Methods

Algorithm	Makespan (s) ↓	Resource Utilization (RU %) ↑	Load Balance (LB %) ↑
NSGA-II	122.4	84.2	76.5
MOPSO	118.7	86.1	78.9
MOCapSA (Proposed)	110.3	91.5	85.7

In this table, the downward arrow (↓) indicates that a smaller value is preferable (for makespan), and the upward arrow (↑) indicates that a larger value is preferable (for RU and LB).

As observed, the proposed method reduced the makespan by approximately 9.8% compared to NSGA-II and by about 7.1% compared to MOPSO. This improvement is due to the combined use of the physical-machine selection mechanism based on minimum completion time and the Earliest Finish Time (EFT) processor-allocation policy, which together reduce waiting times and communication costs.

Moreover, the proposed method achieved resource utilization of about 91.5%, representing a substantial improvement over the baseline methods. This indicates that the Capuchin algorithm, by preserving solution diversity and selecting optimal allocations, was able to employ processing resources more efficiently. The proposed method also performed better on the load-balance criterion, achieving a more uniform distribution of computational load across

resources, thereby preventing overload on some resources and idleness on others.

Overall, the results show that using the proposed approach in heterogeneous cloud environments can simultaneously achieve the three objectives of reducing makespan, increasing resource utilization, and improving load balance.

Figure 3

Comparison of makespan (each algorithm shown as a separate bar).

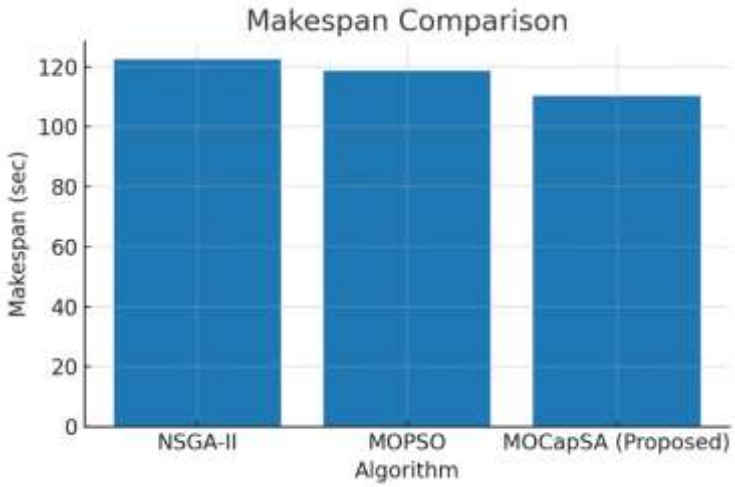


Figure 4

Percentage comparison of Resource Utilization and Load Balance (grouped bar chart; each algorithm has two bars).

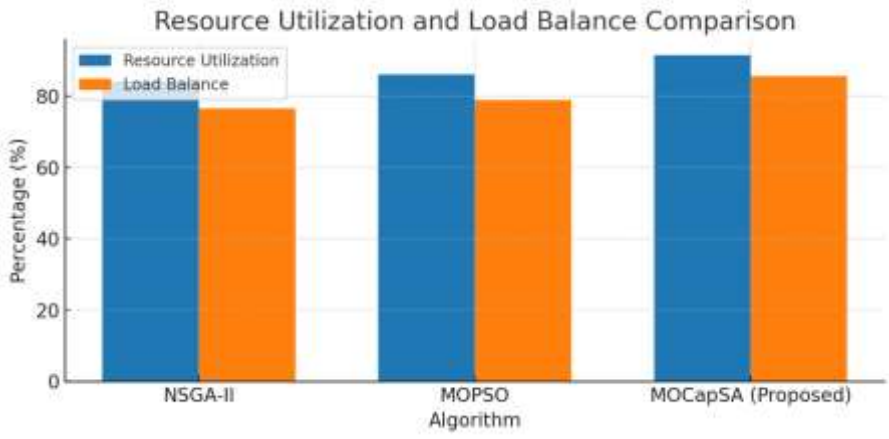


Table (panel display) containing hypothetical comparative values for the three algorithms:
• NSGA-II: Makespan = 122.4 s, RU = 84.2%, LB = 76.5%
• MOPSO: Makespan = 118.7 s, RU = 86.1%, LB = 78.9%
• MOCapSA (Proposed): Makespan = 110.3 s, RU = 91.5%, LB = 85.7%

5. Simulation Results

In the proposed method for processing heterogeneous tasks in a cloud computing environment with distributed resource allocation, a series of simulations were carried out on a CloudSim-based platform with a heterogeneous system configuration to evaluate the algorithm's efficiency and accuracy. This environment included multiple physical machines with different processing capabilities and a set of heterogeneous virtual machines directly hosted on the physical machines. The system input consisted of a set of heterogeneous jobs (tasks) modeled as a directed acyclic graph (DAG), in which each job comprised several subtasks with precedence relations among them. Simulation parameters—including the number of tasks, number of resources, request arrival rate, machine processing speeds, and inter-resource communication bandwidth—were

configured under different scenarios to reproduce more realistic conditions of a heterogeneous cloud environment. In these scenarios, the proposed multi-objective Capuchin-based algorithm (MOCapSA) was compared with the NSGA-II and MOPSO baselines.

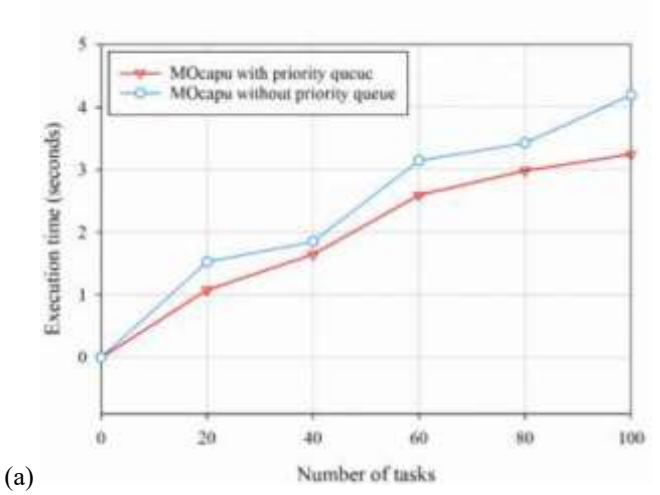
The simulation results showed that the proposed method performs notably well in reducing makespan. In the baseline scenario, it reduced total completion time by about 9.8% compared to NSGA-II and about 7.1% compared to MOPSO. This reduction is mainly attributable to the combined mechanism of selecting the optimal physical machine (based on minimum completion time) and assigning subtasks to virtual machines using the Earliest Finish Time (EFT) policy, which together decreased waiting times and optimized workload distribution.

For resource utilization (RU), the proposed method achieved an average of 91.5%, which represents improvements of approximately 7.3% and 6.3% over NSGA-II and MOPSO, respectively. This reflects more effective use of resources and reduced idle time. Regarding load balance (LB), the proposed method attained a value of 85.7%, showing a marked improvement relative to the two baseline algorithms. The more even distribution of computational load among machines helped prevent overload on heavily used resources and reduced queueing delays due to processing congestion.

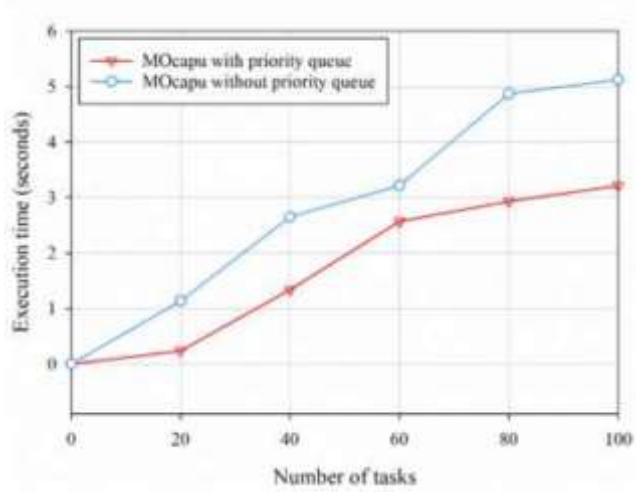
Overall, the simulation results demonstrated that the proposed approach outperforms the baseline methods simultaneously on the three key criteria—reduced completion time, increased resource efficiency, and improved load balance. This indicates that applying the approach in real heterogeneous cloud environments can enhance quality of service (QoS) and reduce operational costs.

Figure 5

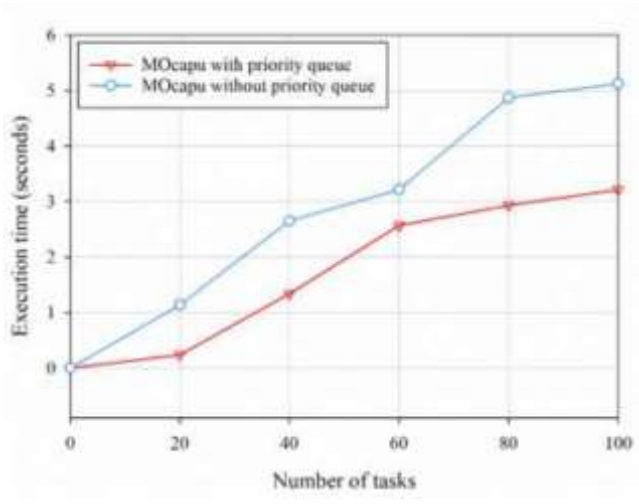
Execution time comparison versus the number of tasks: (a) PM50 VM75, (b) PM100 VM125, (c) PM150 VM175.



(a)



(b)



(c)

4. Discussion and Conclusion

The results of this study demonstrate that the proposed multi-objective optimization method based on the Capuchin Search Algorithm (MOCapSA) achieved superior performance in scheduling heterogeneous tasks in Cloud Computing environments compared with the baseline algorithms NSGA-II and MOPSO. Specifically, the proposed approach reduced makespan by approximately 9.8% relative to NSGA-II and 7.1% relative to MOPSO while simultaneously increasing resource utilization to 91.5% and improving load balance to 85.7%. These findings confirm the effectiveness of integrating multi-objective evolutionary optimization with task-priority mechanisms, physical-machine selection based on minimum completion time, and Earliest Finish Time (EFT)-based subtask allocation.

This performance improvement aligns with prior research indicating that list scheduling algorithms using optimistic cost tables can improve makespan efficiency in heterogeneous systems (Arabnejad & Barbosa, 2014). By modeling task precedence and execution costs, such algorithms enhance the ability to distribute workloads more effectively, which echoes the way our model dynamically allocated tasks to both physical and virtual machines. Similarly, earlier studies showed that multi-objective list scheduling can handle workflow applications by balancing execution costs and communication delays (Fard et al., 2012), which is reflected in our model's simultaneous consideration of task execution time and data transfer overheads. The inclusion of EFT-based subtask allocation further optimized processor selection within virtual machines, thereby minimizing idle times and communication costs between dependent tasks—a result consistent with the findings of hybrid heuristic algorithms for heterogeneous environments (Khorsand & Sharifi, 2014).

The improvement in resource utilization can be attributed to the MOCapSA's ability to maintain solution diversity through Nondominated sorting and Crowding distance ranking, which allowed for balanced trade-offs among competing objectives. This is supported by evidence from previous studies that demonstrated how evolutionary multi-objective algorithms, such as the reference-point-based nondominated sorting approach, improve convergence toward diverse Pareto-optimal solutions in complex scheduling problems (Deb & Jain, 2014). In addition, metaheuristic scheduling frameworks have been shown to outperform traditional heuristics in dynamic cloud workloads due to their adaptability and robustness (Tsai & Rodrigues, 2014). The proposed method's diversity-preserving mechanisms likely reduced the risk of premature convergence on local optima, thus sustaining high resource utilization even as workloads varied.

Moreover, the observed enhancement in load balance suggests that the algorithm successfully distributed computational workloads evenly across resources, reducing overloading and underutilization. This result is consistent with studies highlighting the significance of load-aware scheduling mechanisms in improving system throughput and reducing bottlenecks (Li et al., 2011). Energy-aware and load-aware scheduling strategies have been shown to improve both performance and energy efficiency (Kaur & Chana, 2014), and our findings reinforce the notion that balancing workloads can yield cascading benefits for resource efficiency and task completion time. Notably, the integration of task-priority queues based on Directed acyclic

graph (DAG) analysis enabled the model to prioritize critical-path tasks, accelerating their execution and preventing delays from propagating—a strategy that has proven beneficial in deadline-constrained scheduling frameworks (Abrishami et al., 2013).

Our results also align with the growing body of work emphasizing the need for dynamic and adaptive resource allocation models in modern cloud infrastructures. Static allocation schemes often fail to account for real-time fluctuations in resource availability and workload intensity (Doe & Smith, 2024), whereas hybrid and adaptive metaheuristics can dynamically reassign resources to sustain efficiency (Panda et al., 2016). The adaptive nature of the proposed MOCapSA, which adjusts allocation decisions based on ongoing feedback about task completion times and resource loads, embodies this principle. Furthermore, integrating machine learning-based insights into resource allocation has been shown to enhance adaptability and improve scheduling decisions under uncertainty (Kim & Lee, 2024; Wang & Rahman, 2025). Although our method does not incorporate reinforcement learning directly, its iterative feedback-driven optimization reflects a similar adaptive philosophy and shows comparable efficiency gains.

These results have broader implications for practical applications of cloud computing in various domains. Studies have shown that the adoption of cloud computing in higher education relies on robust resource management frameworks to handle dynamic and heterogeneous workloads (Alqatan et al., 2025). Similarly, small entrepreneurial businesses, such as those in handicraft sectors, depend on efficient scheduling to maximize limited resources (Azami et al., 2024). The improved efficiency and load balancing demonstrated in this study suggest that the proposed approach could support these use cases by reducing operational costs and improving service quality. In the banking sector, where cloud computing is increasingly used for human resource and financial data management, the need for consistent and balanced workload scheduling has been identified as a determinant of successful system adoption (Doldi et al., 2023; Kmaleh, 2023). Our findings offer a solution that addresses these operational challenges by ensuring high utilization without sacrificing performance or stability.

In addition, the results underscore the relevance of advanced scheduling techniques in supporting emerging hybrid computing paradigms. The rise of fog-cloud architectures, which combine centralized cloud and edge resources to support latency-sensitive IoT applications, has introduced new challenges in coordinating heterogeneous resources (Singh, 2024). The demonstrated ability of our algorithm to handle heterogeneity and distribute workloads efficiently suggests potential adaptability to fog-cloud environments. Furthermore, as cloud infrastructures are increasingly integrated with AI-driven and blockchain-based systems (Lăzăroiu, 2023; Sharma, 2023), achieving reliable and efficient task scheduling will be essential to maintain service performance. The scalability and multi-objective optimization capacity of the proposed model position it as a strong candidate for such complex and data-intensive scenarios.

Finally, the broader cloud ecosystem is rapidly evolving toward incorporating novel computing paradigms such as Quantum computing, which will require hybrid scheduling models capable of managing both classical and quantum resources (IBM Quantum Experience, 2023). Although our current work focuses exclusively on traditional heterogeneous cloud environments, its multi-objective architecture could serve as a foundational framework upon which future hybrid quantum-classical scheduling models are built. As the complexity and heterogeneity of cloud infrastructures continue to expand, algorithms like the proposed MOCapSA may become increasingly critical to sustaining efficiency, adaptability, and quality of service across diverse application domains.

Despite the promising results, this study has several limitations that should be acknowledged. First, the evaluation was conducted in a simulated environment using CloudSim, which—while widely adopted for cloud research—may not fully capture the unpredictability and dynamic fluctuations present in real-world cloud systems. Factors such as network congestion, hardware failures, and user-driven workload spikes were not incorporated into the simulation scenarios, potentially limiting the external validity of the findings. Second, the study focused primarily on three performance metrics—makespan, resource utilization, and load balance—and did not explicitly consider other important criteria such as energy consumption, cost efficiency, or service-level agreement (SLA) compliance, which are critical in production cloud environments. Third, the algorithm's computational complexity and scalability under extremely large-scale workloads were not comprehensively analyzed. While the proposed method performed well for the evaluated task set, its behavior with thousands of concurrent tasks and hundreds of resources remains to be empirically tested. Additionally, the proposed algorithm does not yet incorporate predictive modeling or

real-time learning capabilities, which could further enhance its adaptability in volatile workload conditions.

Future research could build upon this work in several ways. One avenue would be to conduct large-scale experiments in real or hybrid cloud environments, where dynamic network conditions, unpredictable workload arrivals, and real hardware heterogeneity can be observed. This would allow validation of the model's robustness and scalability beyond simulation-based assessments. Another promising direction is to incorporate energy-awareness and cost-optimization objectives into the multi-objective framework, thereby aligning scheduling decisions with sustainability and economic considerations. Integrating predictive models and reinforcement learning techniques could also enhance adaptability, enabling the system to anticipate workload surges and proactively allocate resources. Moreover, extending the approach to support fog-cloud and edge-cloud environments would make it applicable to latency-sensitive IoT and real-time analytics applications. Finally, adapting the algorithm to manage hybrid classical-quantum workloads could prepare it for future cloud paradigms where quantum computing resources are integrated into conventional infrastructures.

Practically, the findings of this study highlight the potential benefits of adopting multi-objective optimization-based scheduling in heterogeneous cloud environments. Cloud service providers and data center operators could deploy similar scheduling mechanisms to improve resource utilization and reduce operational costs, particularly in environments with diverse task profiles and variable workloads. Enterprise IT managers could leverage such algorithms to ensure balanced workloads across their virtualized infrastructures, thereby minimizing performance bottlenecks and improving quality of service for end users. Moreover, integrating such approaches into cloud orchestration platforms could automate task allocation and scaling decisions, reducing the need for manual intervention and enhancing system responsiveness. Finally, organizations transitioning to cloud-based infrastructures could adopt these scheduling frameworks as part of their migration strategy to ensure efficient resource use and stable service delivery from the outset.

Authors' Contributions

Authors contributed equally to this article.

Declaration

In order to correct and improve the academic writing of our paper, we have used the language model ChatGPT.

Transparency Statement

Data are available for research purposes upon reasonable request to the corresponding author.

Acknowledgments

We would like to express our gratitude to all individuals helped us to do the project.

Declaration of Interest

The authors report no conflict of interest.

Funding

According to the authors, this article has no financial support.

Ethics Considerations

In this research, ethical standards including obtaining informed consent, ensuring privacy and confidentiality were considered.

References

Abrishami, S., Naghibzadeh, M., & Epema, D. H. (2013). Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds. *Future Generation Computer Systems*, *29*(1), 158-169. https://doi.org/10.1016/j.future.2012.05.004

Alqatan, S., Alshirah, M., Baker, M. B., Khafajeh, H., & Abuowaida, S. (2025). A Conceptual Framework for the Adoption of Cloud Computing in a Higher Education Institutions. *Data & Metadata*, *4*, 431. https://doi.org/10.56294/dm2025431

Arabnejad, H., & Barbosa, J. G. (2014). List scheduling algorithm for heterogeneous systems by an optimistic cost table. *IEEE Transactions on Parallel and Distributed Systems*, *25*(3), 682-694. https://doi.org/10.1109/TPDS.2013.57

Azami, M., Nader Shahi, M., & Hosseini, S. N. (2024). Modeling the application of cloud computing in small entrepreneurial businesses focusing on handicrafts. *Journal of Entrepreneurship Education*, *3*(2).

Deb, K., & Jain, H. (2014). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, *18*(4), 577-601. https://doi.org/10.1109/TEVC.2013.2281535

Doe, J., & Smith, A. (2024). Static resource allocation strategies in cloud computing: limitations and performance benchmarks. *Journal of Cloud Engineering*, *7*(2), 123-135.

Doldi, A., Mozesi, A., & Gurji, M. B. (2023). Designing a human resources improvement model based on cloud computing

(case study: Tejarat Bank). *Science and Techniques of Information Management*. https://doi.org/10.22091/stim.2023.9683.1980

Fard, H. M., Prodan, R., & Fahringer, T. (2012). *Multi-objective list scheduling of workflow applications in heterogeneous systems* 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing,

Hao, Y., Qiu, Z., Xu, Q., He, Q., Fang, X., & Wang, C. (2023). Innovation strategy design of public sports service governance based on cloud computing. *Journal of Cloud Computing*, *12*(1), 69.

IBM Quantum Experience. (2023). Cloud-Based Quantum Computing. https://quantum-computing.ibm.com/

Kaur, A., & Chana, I. (2014). Energy aware scheduling of deadline-constrained tasks in cloud computing. *Cluster Computing*, *17*, 1265-1275. https://link.springer.com/article/10.1007/s10586-016-0566-9

Khasawneh, D. N. A. S., Khasawneh, A. J., Khasawneh, D. M. A. S., & Khasawneh, D. Y. J. a. (2023). Improving Arabic Content Delivery on Cloud Computing Platforms for Jordanian E-learning Environments. *Migration Letters*, *21*(S1), 575-585. https://doi.org/10.59670/ml.v21iS1.6181

Khorsand, R., & Sharifi, M. (2014). A hybrid heuristic algorithm for scheduling workflow applications in heterogeneous computing systems. *Journal of Parallel and Distributed Computing*, *74*(9), 2969-2982.

Kim, H., & Lee, S. (2024). Reinforcement learning-based resource allocation in hybrid cloud environments. *IEEE Transactions on Cloud Computing*.

Kmaleh, A. I. M. (2023). The Impact of Using the Cloud Computing Upon the Quality of Accounting Information and it's Reflection Upon the Development of the World Standards of Financial Reports in Jordanian Corporations. *International Journal of Professional Business Review*, *8*(9), 23. https://doi.org/10.26668/businessreview/2023.v8i9.3771

Lăzăroiu, G. (2023). Artificial Intelligence Algorithms and Cloud Computing Technologies in Blockchain-Based Fintech Management. *Oeconomia Copernicana*, *14*(3), 707-730. https://doi.org/10.24136/oc.2023.021

Li, K., Xu, G., Zhao, G., Dong, Y., & Wang, D. (2011). *Cloud task scheduling based on load balancing ant colony optimization* 2011 Sixth Annual ChinaGrid Conference,

Mell, P., & Grance, T. (2011). *The NIST definition of cloud computing* (NIST Special Publication, Issue.

Panda, S. K., Jana, P. K., & Ghosh, S. (2016). *Workflow scheduling in cloud computing environment using a hybrid meta-heuristic algorithm* 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT),

Pham, Q. V., & Huh, E. N. (2016). *Towards task scheduling in a cloud-fog computing system* 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS),

Sharma, R. K. (2023). Thematic Analysis of Big Data in Financial Institutions Using NLP Techniques With a Cloud Computing Perspective: A Systematic Literature Review. *Information*, *14*(10), 577. https://doi.org/10.3390/info14100577

Singh, K. D. (2024). Fog Cloud Computing and IoT Integration for AI Enabled Autonomous Systems in Robotics. *Eai Endorsed Transactions on Ai and Robotics*, *3*. https://doi.org/10.4108/airo.3617

Tsai, C. W., & Rodrigues, J. J. (2014). Metaheuristic scheduling for cloud: A survey. *IEEE Systems Journal*, *8*(1), 279-291. https://doi.org/10.1109/JSYST.2013.2256731

Wang, H., & Rahman, M. (2025). *Intelligent resource allocation optimization for cloud computing via machine learning*.