DOI: 10.15849/IJASCA.251130.14

Int. J. Advance Soft Compu. Appl, Vol. 17, No. 3, November 2025
Print ISSN: 2710-1274, Online ISSN: 2074-8523

Enhancing E-Government Cloud Service Composition with PSO: A Comparative Study on Resource Allocation and Energy Efficiency Against ACO

Issam AlHadid[1], Evon Abu-Taieh[2,*], Mohammad Al Rawajbeh[3] , Suha Afaneh[4] , Mohammed E. Daghbosheh[5], Hamed S. Albdour[6], Rami S. Alkhawaldeh[7], Ala'aldin Alrowwad[8] , Sufian Khwaldeh[9]

[1] Department of Computer Information Systems, Faculty of Information Technology and Systems, The University of Jordan, Jordan. mail: i.alhadid@ju.edu.jo
[2] Department of Computer Information Systems, Faculty of Information Technology and Systems, The University of Jordan, Jordan. mail: e.abutaieh@ju.edu.jo
[3] Computer Science Department, Faculty of Science and Information Technology, Al-Zaytoonah University of Jordan, Jordan. mail: m.rawajbeh@zuj.edu.jo
[4] Department of Cybersecurity, Faculty of Information Technology, Zarqa University, Jordan. mail: S.afaneh@zu.edu.jo
[5] Department of Artificial Intelligence, The Faculty of Science and Information Technology, Irbid National University, Jordan.
mail: m.daghbousheh@inu.edu.jo
[6] Department of Information Technology, Faculty of King Abdullah II School for Information Technology, The University of Jordan, Jordan; mail: h.bdour@ju.edu.jo
[7] Department of Computer Information Systems, Faculty of Information Technology and Systems, The University of Jordan, Jordan. mail: r.alkhawaldeh@ju.edu.jo
[8] Department of Public Administration, School of Business, The University of Jordan, Amman, Jordan. mail: a.alrowwad@ju.edu.jo
[9] Department of Computer Information Systems, Faculty of Information Technology and Systems, The University of Jordan, Jordan; mail: sf.khwaldeh@ju.edu.jo

* Corresponding Author: e.abutaieh@ju.edu.jo

Abstract

The composition of electronic government (E-government) cloud services is essential for optimizing resource allocation, load balancing, and energy efficiency, thereby ensuring seamless digital service delivery. optimizing cloud service composition remains a massive challenge because of dynamic workloads and varying aid needs. This research investigates the implementation of Particle Swarm Optimization (PSO) to improve the cloud service composition in e-government platforms by effectively distributing workloads, minimizing execution time, enhancing Virtual Machine (VM) usage, and decreasing energy consumption. The performance of the proposed PSO-based technique was evaluated using CloudSim 5.0, and the results were compared with the Ant Colony Optimization (ACO) algorithm across key performance metrics, such as execution

Received 4 July 2025; Accepted 1 September 2025

time, energy consumption, load balancing efficiency, and Virtual Machines (VM) utilization. Experimental results show that the PSO-based technique outperforms ACO in optimizing cloud resource utilization. Specifically, PSO achieved higher VM utilization, with an average utilization rate 77.1%, compared to 69.1% for ACO, ensuring a better workload distribution. Additionally, PSO confirmed higher energy efficiency and decreased execution time, contributing to reduced operational costs. These findings highlight the potential of intelligent optimization algorithms in enhancing cloud service composition, paving the way for greater adaptive and efficient resource control in cloud environments.

Keywords: *Cloud Computing, PSO, ACO, CloudSim, E-government, E-Services, Optimization*

1 Introduction

Government e-services are considered as critical components of modern public administration, where the e-services leverage cloud computing to deliver efficient, accessible, and reliable digital services to citizens and businesses [1], [2]. These services encompass a wide range of applications that require robust, scalable, and secure infrastructures to handle varying user demands [3], [4], [5]. By adopting cloud-based solutions, governments can enable scalability, reduce operational costs, and ensure high availability infrastructures, even during peak usage periods [6], [7]. Where cloud adoption is essential for ensuring seamless public service delivery, optimizing resource utilization, and improving system performance [8], [9]. Scholars [10], [11], [12], [13] stated that one of the key challenges in cloud computing is service composition, which involves selecting and integrating multiple cloud services to meet users' requirements within the Service Level Agreement (SLA), optimizing critical performance metrics such as execution time, resource allocation, energy consumption, load balancing, response time, and service quality.

According to [14], [15], [16] cloud service composition complexity is related to the heterogeneous nature of cloud environments, unstable workload demands, and the need to balance multiple optimization objectives.. Wajid et al. [16] and Nezafat Tabalvandani et al. [17] stated that green workload distribution and resource allocation are vital for service reliability, sustainable cloud operations, and reducing operational fees.

Researchers [18], [19], [20], [21] argue that traditional approaches may not to achieve optimal service composition due to the dynamic nature of cloud resources. As a result, metaheuristic algorithms were widely employed to discover near-optimal solutions within a reasonable and inexpensive computational time and resources [14], [19], [20], [22], [23]. Among these, Particle Swarm Optimization (PSO) has gained significant attention due to its simplicity, fast convergence, and adaptability in complex optimization problems [24], [25], [26]. Inspired by swarm intelligence in nature, PSO dynamically adjusts service selection based on both individual and collective experience, making it effective for optimizing cloud resource allocation and load balancing [27], [28]. However, despite its advantages, PSO's performance must be evaluated against other well-established optimization techniques to determine its suitability for cloud service composition [29].

This study aims to compare PSO with Ant Colony Optimization (ACO) the popular bio-inspired algorithm that simulates the pheromone-based foraging behavior of ants [30], [31]. Where ACO demonstrated effectiveness in cloud optimization, its reliance on iterative pheromone updates can lead to higher computational overhead [32], [33]. The comparison between PSO and ACO is conducted by analyzing key Cloud computing

performance metrics, including execution time, energy consumption, load balancing, response time, and service quality.

The performance metrics of cloud computing are important for evaluating the efficiency and cost-effectiveness of cloud-based services [34], [35]. Where execution time represents the time required for a task or a set of tasks (cloudlets) to be processed and completed in a cloud computing environment [36], [37]. Execution time in the cloud environment includes the required time for resource allocation, data transfer, computation, and result retrieval, which directly impacts the user experience and application performance [35], [38]. Cost is considered as one of the critical metrics in the cloud computing environment [39], Cost encompasses the direct financial charges related to the use of cloud resources but in addition to the other indirect costs associated with data transfer, storage, and potential over-provisioning [37], [40]. Power Consumption has gained increasing attention due to its environmental implications, according to Katal et al. [41], where the Power Consumption is the total energy consumed by a data center's physical and virtual machines (VMs), network infrastructure, and other resources. Researchers [36], [41], [42] stated that optimizing energy can reduce operational costs and carbon footprints, making it a key for sustainable cloud computing.

Service Level Agreement (SLA) is a formal contract between a cloud service provider and a client, specifying overall performance metrics [43], [44], SLA violations arise when cloud vendors fail to fulfill predefined performance guarantees, such as availability, throughput, response time, or resource allocation, leading to potential consequences, penalties and customer dissatisfaction [45], [46]. To reduce SLA violations, advanced load balancing strategies have evolved to distribute workloads lightly across servers and virtual machines, ensure optimal performance, avoid bottlenecks, and enhance utilization of cloud resources[47], [48], [49]. Shafiq et al. [50] and Milani and Navimipour [51] identify load balancing as the process of distributing workloads across multiple virtual machines (VMs) to prevent resource overutilization or underutilization. Response Time is the time taken for a cloud service to respond to a user request [22], [48], Sreeramulu et al. [23] and Kadarla et al. [52] stated that response time directly impacts user experience and application efficiency, where minimizing response time is essential for ensuring high-quality service delivery minimizing SLA violations.

Response Time includes network latency, data retrieval and processing times, in addition to any delays in communication between different cloud components [51], [53]. Together, these metrics provide a comprehensive view of cloud performance, helping organizations make informed decisions about resource allocation, workload management, service quality and optimization. Scholar [19] stated that response time is "the total time needed for a cloud service to successfully respond to a request", including network latency, data retrieval and processing times, in addition to other delays in communication between the cloud components [51], [53], [54]. Accordingly, the mentioned metrics provide a comprehensive view of cloud performance, which supports organizations in making informed decisions about resource allocation, workload management, service quality, and optimization.

This study evaluates the efficiency of PSO in improving e-government cloud service composition. The following are the study objectives and key contributions:

- Develops a PSO-based cloud service composition model that provides an effective and efficient and resource allocation and power management.

- Compare the performance of PSO with ACO, highlighting the performance of swarm intelligence-based methods in cloud optimization.
- Evaluates important cloud performance metrics, such as execution time, power consumption, response time, costs, and SLA violations.
- Demonstrates the applicability of PSO in E-Gov. cloud services, presenting insights into its capability for actual, real-world cloud service control and management.

By addressing those objectives, this study contributes to the advancement of cloud service optimization, allowing greater efficiency and sustainable E-Gov. service delivery.

2 Related Work

Cloud computing has played a transformative role in e-government services by enabling scalable, cost-effective, and on-demand access to critical public sector applications [1], [2]. Efficient cloud service composition is essential for ensuring high availability, security, and performance in e-government infrastructures [6], [7], [55]. Metaheuristic algorithms have been widely used in optimizing resource allocation, improving service quality, and minimizing Power Consumption in cloud-based e-government systems [14], [19], [20], [22], [23]. Qazi et al. [45], Sharma et al. [56] and Jula et al. [57] discussed the challenges and advancements in optimizing cloud service composition using the heuristic and metaheuristic techniques. Kadhim et al. [58] investigated the limitations of traditional rule-based and exact optimization methods, which often fail to scale with the increasing complexity of cloud environments. Scholars [45], [59], [60], [61] argue that traditional methods often fail to manage the dynamic, heterogeneous, and large-scale nature of modern cloud systems, resulting in suboptimal performance. Several studies [14], [62], [63] claimed that Bio-inspired optimization algorithms have emerged as viable alternatives, these techniques provide enhanced flexibility, scalability, and efficiency in addressing complex optimization challenges.

Several scholars have studied heuristic and metaheuristic approaches and techniques for optimizing cloud service composition. Bei et al. [33] introduced an enhanced ACO algorithm for service composition in multi-cloud networks, utilizing a multi-pheromone mechanism to optimize QoS parameters. they incorporated a genetic algorithm-inspired mutation operation to avoid local optima, shao et al. [64] proposed a hybrid heuristic algorithm known as PGSAO, which integrates the GA and PSO for scheduling data-intensive tasks in a heterogeneous cloud to avoid local optima and enhance exploitation capabilities. Cheikh and Walker [65] introduce a hybrid particle swarm optimization approach to solve the task scheduling problem in the cloud. The algorithm focuses on optimizing the critical metric for scheduling tasks across multiple VMs. Kurdi et al. [66] introduced a multi-cloud service composition technique using a cuckoo algorithm, employing the MultiCuckoo to efficiently determine the optimal service composition in a multi-cloud environment (MCE). Similarly, Tarawneh et al. [14] proposed an intelligent cloud service composition optimization using spider monkey and multistage forward search algorithms. The researchers claimed that the proposed solution maintains an optimal balance between VMs' resources, processing capacity, and service composition capabilities. Wei [67] proposed a task scheduling optimization strategy using an improved ant colony optimization algorithm in cloud computing, researchers focused on avoiding local optima and optimizing resource load balancing. The proposed approach integrates a satisfaction function combining waiting time, resource load balance, and task completion cost.

In their study, Chahal et al. [68] utilized the Grey Wolf Algorithm (GWA) to analyze and repair the process of failed virtual machines in cloud computing. The proposed approach provides a framework for enhancing the reliability and cost-effectiveness of cloud computing systems. Behera and Sobhanaya [69] proposed a hybrid GA-GWO algorithm, combining Genetic Algorithm (GA) and Grey Wolf Optimization (GWO), to address multi-objective task scheduling challenges in cloud computing. Yu et al. [70] introduced an Advanced Willow Catkin Optimization (AWCO) algorithm to enhance task scheduling in cloud computing by improving global search capability and convergence speed. The AWCO algorithm incorporates quasi-opposition-based learning and sinusoidal mapping to optimize resource utilization and load balancing.

The Particle Swarm Optimization (PSO) algorithm has been widely used in cloud computing for load balancing, task scheduling, and resource allocation, where PSO has demonstrated high efficiency in minimizing execution time and optimizing virtual machine (VM) utilization while maintaining a balance between convergence speed and solution quality [22], [24], [71], [72]. In contrast, Ant Colony Optimization (ACO) is another swarm intelligence-based algorithm inspired by the pheromone-laying behavior of ants [31], [73]. ACO has been successfully applied in cloud environments for task scheduling and load balancing, demonstrating robustness in discovering optimal paths but suffering from slow convergence and local optima stagnation [74], [75], [76].

While several studies have applied metaheuristic techniques and approaches for cloud service structure, many focus on traditional or single-algorigar methods with limited emphasis on hybrids or adaptive adaptation. [56], [64], [67], [76]. However, some studies directly evaluate these methods in terms of e-government systems or include SLA violations and energy efficiency in the form of core performance matrix. This difference highlights the need for targeted research that compares not only to a well -established algorithm such as PSOs and ACOs, but also benchmark them against emerging hybrid and adaptive models to ensure practical relevance for public sector cloud infrastructure.

This study proposes a framework based on Particle Swarm Optimization (PSO) to optimize the cloud environment resource allocation, VM utilization, and energy consumption, unlike conventional approaches that prioritize either execution time or cost reduction independently. In addition, this study integrates multiple performance metrics, including SLA violations, latency, load balancing efficiency, and task completion rate, providing a more comprehensive evaluation. Also, this study contributes to existing literature by implementing and evaluating PSO for cloud service composition, with a direct performance comparison against ACO. Furthermore, by utilizing CloudSim 5.0 for rigorous simulation and direct comparative analysis between PSO and ACO, this research establishes a more scalable and efficient approach to cloud service composition. This holistic optimization strategy addresses key limitations in prior work, contributing significantly to the advancement of intelligent cloud resource management, where the findings will help cloud service providers and researchers select the most suitable optimization technique for efficient cloud resource management.

3 Methodology

This research methodology involves several steps to efficiently analyze and evaluate PSO [26] and ACO algorithms [73] using the CloudSim 5.0 [77]. First, the CloudSim simulation environment is set up by defining and configuring data centers, virtual machines (VMs), cloudlets (tasks), services, and the PSO and ACO optimization.

Figure (1) illustrates the PSO algorithm pseudo-code, which begins with Initializing algorithm constants, where positions and velocities are initialized randomly and assigned to particles (solutions) from the search space, each particle's fitness is evaluated based on the optimization objective, and later on the algorithm iteratively updates the personal best (pBest) and global best (gBest) solutions, if the stopping criterion is not satisfied, the algorithm keeps changing the velocities and positions of the particles iteratively to find better solutions. Once the Stopping condition is met or the ideal solution is discovered, the algorithm deploys the optimized solution. This iterative procedure enables efficient and adaptable problem-solving across diverse optimization scenarios.

```
ALGORITHM PSO_Cloud_Service_Composition
    INPUT:
      Cloud resources (Data centers, Virtual Machines (VMs), Cloudlets)
      PSO parameters (Swarm size, Inertia weight w, Acceleration coefficients c1, c2)
      Performance metrics (Execution time, Power consumption, Load balancing efficiency, etc.)
    OUTPUT: Optimized service composition with efficient resource allocation and energy
    management.
    STEP 1: INITIALIZATION
      1.1 Initialize a population (swarm) of particles (service compositions).
      1.2 Assign random positions (initial service compositions) and velocities to particles.
      1.3 Evaluate each particle's fitness based on cost, execution time, energy, and other factors.
      1.4 Set each particle's initial best-known position as its personal best (pBest).
      1.5 Identify the global best position (gBest) from the swarm.
    STEP 2: ITERATIVE OPTIMIZATION
      FOR each iteration UNTIL stopping criterion is met (e.g., max iterations or convergence):
        2.1 FOR each particle in the swarm:
          2.1.1 Update velocity using the PSO velocity equation:
            Velocity = w * Velocity + c1 * rand() * (pBest - Position) + c2 * rand() * (gBest -
    Position)
          2.1.2 Update position of the particle based on its new velocity.
          2.1.3 Evaluate the fitness of the new position.
          2.1.4 Update pBest if the new position has better fitness.
          2.1.5 Update gBest if any particle has found a better solution.
        2.2 Reduce inertia weight (w) gradually to balance exploration and exploitation.
      END FOR
    STEP 3: DEPLOY OPTIMIZED SERVICE COMPOSITION
      3.1 Assign cloudlets to VMs based on gBest solution.
      3.2 Monitor execution and resource utilization.
    RETURN gBest (Optimal service composition)
END ALGORITHM
```

Figure 1. Pseudocode of the PSO for Cloud Service Composition

The candidate solutions in the PSO implementation for cloud service composition are first represented by a swarm of particles, each of which represents numerous service compositions. Key performance measures, including execution time, cost, VMs' utilization, SLA, and energy usage, are used to assess each particle. To explore better solutions, the algorithm iteratively updates the particles that represent cloud service compositions by modifying their position and velocity. The PSO Algorithm improves cloud service efficiency by optimizing load balancing, resource allocation, and task scheduling. Similarly, the ACO Algorithm is used to determine the best service compositions based on the heuristic data and pheromone trails. Lastly, to evaluate the performance of PSO and ACO across various metrics, simulations are performed under varied workload scenarios.

Figure 1 shows the proposed PSO algorithm pseudocode for the cloud service composition method.

4 The Proposed Method

In this section, we will discuss the implementation of the PSO-based Cloud Service Composition, outlining the complete process from initialization to optimization and final deployment. This discussion will cover key aspects including particle representation, fitness evaluation, velocity and position updates, convergence criteria, and performance analysis. Additionally, we will explore how PSO dynamically adapts to workload variations, ensuring efficient task scheduling, resource allocation, and load balancing within cloud environments.

4.1 Initialization Step

The PSO method starts by creating a swarm of particles, each one representing a possible composition of cloud services. The effectiveness of service allocation in the cloud environment is determined by several attributes that comprise each particle, the key attributes include:

- Position (Xi): Indicates which services are assigned to which virtual machines (VMs) based on the current state of service composition. Every place in the search space represents a potential solution.
- Velocity (Vi): It manages the particle's motion including speed and direction in the solution space, impacting the evolution of the service composition in later iterations.
- Personal Best(pBest): is the local best position that Keeps track of each particle's optimal service composition, as determined by key performance metrics, during its search.
- Global Best (gBest): is the global best position among all the particles, it represents the overall best service composition found by any particle in the swarm and is considered a guiding reference for other particles.
- To guarantee variety in the initial population, each particle is assigned an initial position and velocity randomly within specific constraints. This randomization technique improves the PSO algorithm's capability to investigate and explore a large number of potential solutions and avoids premature convergence.

The initialization process is defined in Eq. (1) and Eq. (2) as follows:

$$X_i = X_{min} + rand() \times (X_{max} - X_{min}) \quad \text{…….(1)}$$

$$V_i = V_{min} + rand() \times (V_{max} - V_{min}) \quad \text{…….(2)}$$

In order to maintain possible solutions, the bounds Xmin, Xmax, Vmin, and Vmax are defined, and values within the range [0,1] are generated using the rand() function. After the PSO initialization process, each particle's fitness is evaluated using a predetermined objective function based on different factors, including execution time, cost, energy consumption, and SLA violations. By the end of the initialization process, the (pBest) and (gBest) are identified.

4.2 Iterative Optimization Step

Once the particles are initialized, the iterative optimization process begins. The algorithm refines service compositions over multiple iterations, dynamically adjusting service allocations based on previous performance evaluations. In each iteration, the following steps are applied:

4.2.1. Velocity Update

The velocity of each particle is updated using the PSO velocity equation:

$$V_i^{(t+1)} = \omega . V_i^{(t)} + c_1 . r_1 . (pBest_i - X_i^{(t)}) + c_2 . r_2 . (gBest - X_i^{(t)}) \quad \text{.....(3)}$$

where:

- ω is the inertia weight, controlling the balance between exploration and exploitation.
- c_1, c_2 are acceleration coefficients, influencing the impact of personal and global best solutions.
- r_1, r_2 are random values [0,1], to ensure stochastic behavior.

A higher inertia weight in the early iterations encourages global exploration, while a lower inertia weight in later stages promotes local exploitation for convergence [78].

4.2.2. Position Update

After updating the velocity, the new position of the particle is determined using:

$$X_i^{(t+1)} = X_i^{(t)} + V_i^{(t+1)} \quad \text{.....(4)}$$

This step refines the service allocation strategy by adjusting service compositions based on the updated velocity.

3.2.3. Fitness Evaluation

The fitness of the new solution is calculated based on multiple cloud performance metrics, such as:

- Execution Time: The total time required for task completion.
- Energy Consumption: The power utilized by VMs in executing cloudlets.
- Load Balancing Efficiency: The even distribution of workload across resources.
- SLA Violations: The number of instances where service delivery fails to meet predefined performance agreements.

4.2.4. Updating Personal and Global Best Solutions

- If the newly computed fitness value is better than the particle's previous pBest, it is updated.
- If the new fitness value outperforms the current gBest, the global best is updated accordingly.

4.2.5. Convergence Check

The iterative process continues until the termination condition is met. Common termination conditions include:

- A predefined maximum number of iterations is reached.
- Fitness improvement falls below a predefined threshold, indicating convergence.

The final global best solution (gBest) obtained at the end of the optimization process represents the optimal cloud service composition, ensuring efficient resource allocation, minimal execution time, and reduced power consumption.

4.3. Deploying the Optimized Service Composition Step

Following the optimization process, the (gBest) is chosen for deployment which represents the best service composition identified by the PSO algorithm. And the Cloudlets are efficiently mapped to virtual machines (VMs), ensuring optimal resource allocation while maintaining SLAs and system constraints.

4.3.1. Mapping Optimized Service Composition to Cloud Resources

The optimized composition is deployed by means of assigning tasks to VMs based on the PSO algorithm's most appropriate resource allocation. The deployment follows a subsequent-structured technique:

- Service Placement Strategy: services are allocated to virtual machines (VMs) according to their execution efficiency, to ensure low response times and lower power consumption.
- Resource-aware Task Scheduling: To optimize usage and avoid bottlenecks, cloudlets are scheduled on virtual machines (VMs), while carefully considering the availability of CPU resources, memory capacity, and network bandwidth
- Enforcement of Load Balancing: To avoid resource overuse and underuse, workloads are split equally across several virtual machines.

The optimized allocation is defined in Eq. (5). The equation is used to minimize the workload imbalance across VMs by dynamically selecting the VM with the least utilization, preventing overloading, and improving system efficiency.

$$VM_j = arg\ min_j \frac{\sum_{i=1}^{n} X_{i,j}}{C_j} \quad \text{…….(5)}$$

where:

- $X_{i,j}$ represents the task assignment matrix (1 if taski is assigned to VMj, otherwise 0).
- C_j is the processing capacity of VMj.
- $arg\ min_j$: selects the VM with the lowest workload utilization, ensuring a balanced task distribution across VMs.
- $\frac{\sum_{i=1}^{n} X_{i,j}}{C_j}$: represents the total workload assigned to VMj.

4.3.2. Dynamic Resource Monitoring and Adjustment

To ensure the effectiveness of the optimized deployment, a real-time tracking mechanism is carried out. This entails monitoring key overall performance indicators (KPIs) to maintain service efficiency:

- Execution Time Monitoring: Tracks the time taken for each task to complete and adjusts scheduling dynamically if delays are detected.
- Power Consumption Analysis: Evaluates energy efficiency in real-time, identifying VMs that may require load redistribution to minimize power usage.
- SLA Compliance Checks: Monitors service quality to ensure that SLA constraints (e.g., response time, availability, throughput) are met.

4.3.3. Performance Optimization through Auto-Scaling

An auto-scaling mechanism is adopted by the cloud computing environment to enhance cloud efficiency. The mechanism allows dynamic resource scaling based on workload requirements, including Vertical and Horizontal Scaling:

- Vertical Scaling (Scaling Up/Down VMs): Adjusting the CPU, RAM, or bandwidth of existing VMs, to ensure that a single VM can handle an increased workload without adding more VMs.
- Horizontal Scaling (Adding/Removing VMs): Provisioning or decommissioning VMs based on workload intensity, to ensure better load balancing and cost efficiency. Horizontal Scaling uses the predefined utilization thresholds θ_{high} and θ_{low} to determine when to add or remove resources

Accordingly, the system decides whether to scale up, scale down, or maintain the current configuration based on resource utilization thresholds. Eq. (6) represents the scaling decision function:

$$U_{avg} = \frac{1}{m}\sum_{j=1}^{m} U_j \qquad \text{……...…6}$$

Where:

- U_{avg}= Average VM utilization (CPU, memory, bandwidth).
- m: Total number of active VMs.
- U_j = Utilization of VM_j, which is calculated using eq. (7):

$$U_j = \frac{Current\ Workload_j}{VM\ Capacity_j} \times 100 \qquad \text{……..(7)}$$

If the average utilization exceeds a predefined upper threshold θ_{high} , additional resources are required to maintain performance, as shown in eq. (8):

$$\text{if } U_{avg} > \theta_{high} \text{ Scale Up (Add VM)} \qquad \text{……(8)}$$

Where a new VM is provisioned, eq. (9):

$$m_{new} = m + 1 \text{ …….(9)}$$

If the average utilization drops below a predefined lower threshold θ_{low}, excess VMs should be removed to reduce costs:

$$\text{if } U_{avg} < \theta_{low} \text{ Scale Down (Remove VM)} \quad \text{……(10)}$$

And a VM is decommissioned to improve cost efficiency by turning off underutilized VMs, eq. (11):

$$m_{new} = m - 1 \text{ .....(11)}$$

On the hand, if the system is within an optimal range shown in eq. (12), then No scaling action is performed, ensuring stability and efficiency:

$$\theta_{low} \leq U_{avg} \leq \theta_{high} \quad \text{……..(12)}$$

To enhance adaptability, thresholds ($\theta_{\text{low}}, \theta_{\text{high}}$) can be dynamically adjusted using historical data and predictive models, a dynamic threshold function can be defined using the equations (13) and (14):

$$\theta_{high} = \theta_{base} + \alpha \times \sigma \quad \text{……..(13)}$$

$$\theta_{low} = \theta_{base} - \beta \times \sigma \quad \text{…….(14)}$$

Where:

- θ_{base}=Base threshold (default value).
- σ = Standard deviation of recent utilization data.
- α, σ = Scaling sensitivity parameters.

The adaptive mechanism shown in eq. 13 and eq. 14 ensures that the system dynamically adjusts to workload fluctuations without excessive scaling actions. Accordingly, the auto-scaling model offers several key advantages, where it keeps SLA violations to a minimum by ensuring virtual machines (VMs) are used efficiently, reduces operational expenses by decommissioning underutilized VMs, enhances system stability by preventing unnecessary scaling actions, and finally, improves energy efficiency by optimizing resource allocation.

4.3.4. Fault Tolerance, Reliability, and Validation

To make the system more resilient, the deployment includes several fault-tolerant mechanisms, such as backup service replication, proactive failure prediction, and automatic task migration. These measures help ensure reliability and minimize disruptions. Before the deployment is finalized, the system goes through a thorough validation process to optimize performance. This includes benchmarking key metrics, assessing energy efficiency, and evaluating user satisfaction to ensure the system runs smoothly and effectively. The next section focuses on running simulations in CloudSim 5.0 [77] to evaluate PSO against ACO. Different workload scenarios are tested to evaluate performance metrics, including execution time, power consumption, load balancing, SLA violations, and response time.

5 Simulation, Results and Discussions

In this study, a cloud computing environment was modeled and conducted using the CloudSim 5.0 Framework [38], [77]. All simulation experiments were performed on a machine equipped with an Intel Core i7 processor (12th Gen) running at 3.6 GHz, 16 GB of DDR4 RAM, a 512 GB NVMe SSD, and the Windows 11 operating systems (64-

bit), and the simulation parameters were configured to reflect a realistic cloud computing scenario as shown in Table 1.

Table 1: CloudSim 5.0 Configuration and Settings.

Parameter	Value
Number of Data Centers	3
Number of Virtual Machines (VMs)	50
VM Configuration	2 GHz CPU, 4 GB RAM, 100 GB Storage
Number of Cloudlets (Tasks)	500
Cloudlet Length	1000 to 10000 MIPS
Number of Services in Composition	5 to 20 per composition
Scheduling Policy	Time-Shared & Space-Shared
Number of PSO Particles	30
PSO Parameters (w, c1, c2)	w=0.9 to 0.4, c1=2, c2=2
Number of ACO Ants	30
ACO Parameters (α, β, ρ)	α=1, β=2, ρ=0.5
Simulation Time	1000 simulation ticks

To evaluate the performance of PSO and ACO in a cloud environment, the simulations were conducted under different workload conditions, including low, medium, and high service demand. All the simulation experiment results were collected over multiple runs to ensure consistency and reliability and analyzed to compare the effectiveness of PSO and ACO in the cloud environment.

The experimental results presented in the next sections show the comparison of the performance of Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) across several key metrics, including execution time, power consumption, load balancing, SLA violations, and response time. The data was collected over (10) runs, and the results indicate consistent trends between the two algorithms.

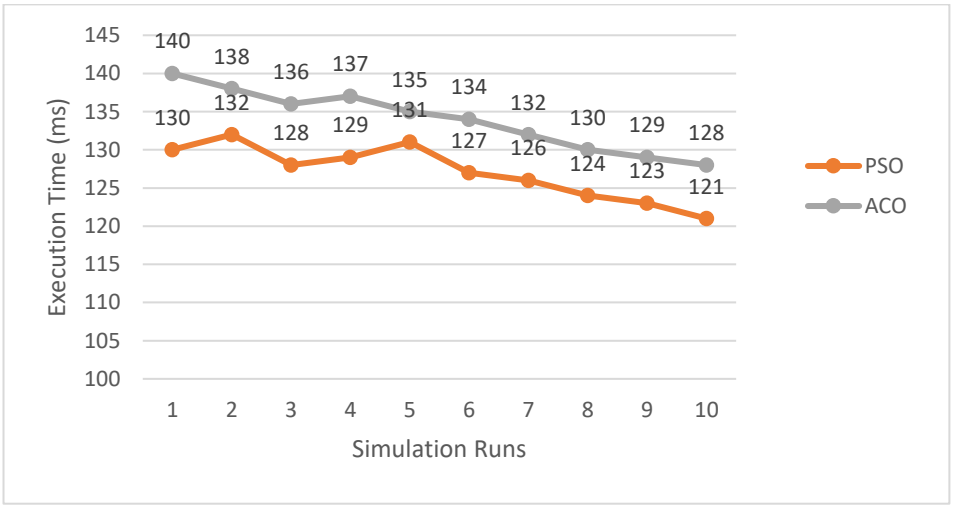

Fig. 3: Execution time comparison of PSO vs ACO

According to the results shown in Figure 3, the PSO algorithm outperformed ACO in all simulation runs, where the results confirm that all the POS tasks were completed faster than ACO every time. Figure 3 shows that the execution time using PSO ranged between 121 and 132 seconds, while ACO needed 128 to 140 seconds to finish the execution.

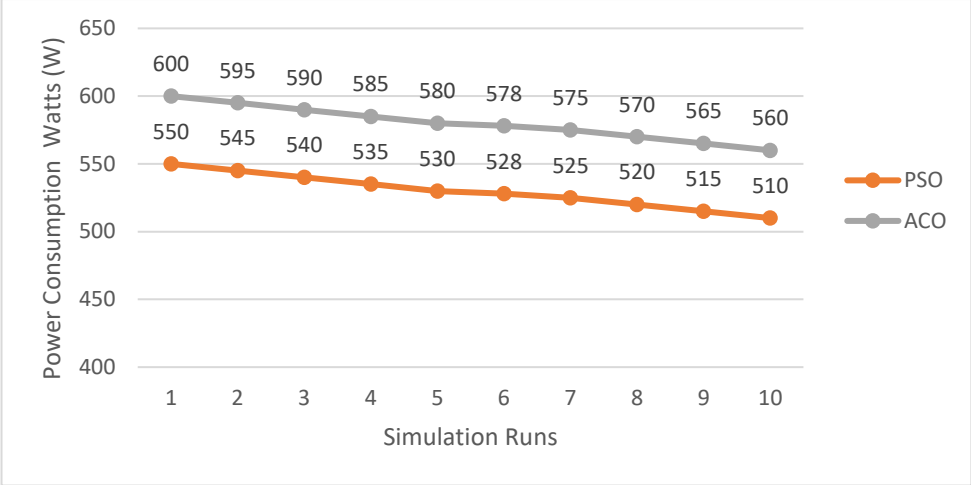

Fig. 4: Power consumption comparison of PSO vs ACO

The results illustrated in Figure 4 display that PSO demonstrates a higher power consumption in comparison to the outcomes of ACO. Wherein the energy consumption for PSO ranged from 510 to 550 Watts, while ACO's power intake ranged from 560 to 600 Watts. This suggests that PSO is more energy-efficient and electricity-green, which can be a vast advantage in environments where energy management is critical. On the other hand, ACO showed higher electricity intake than PSO, which is taken into consideration as a disadvantage in real-world practical applications.

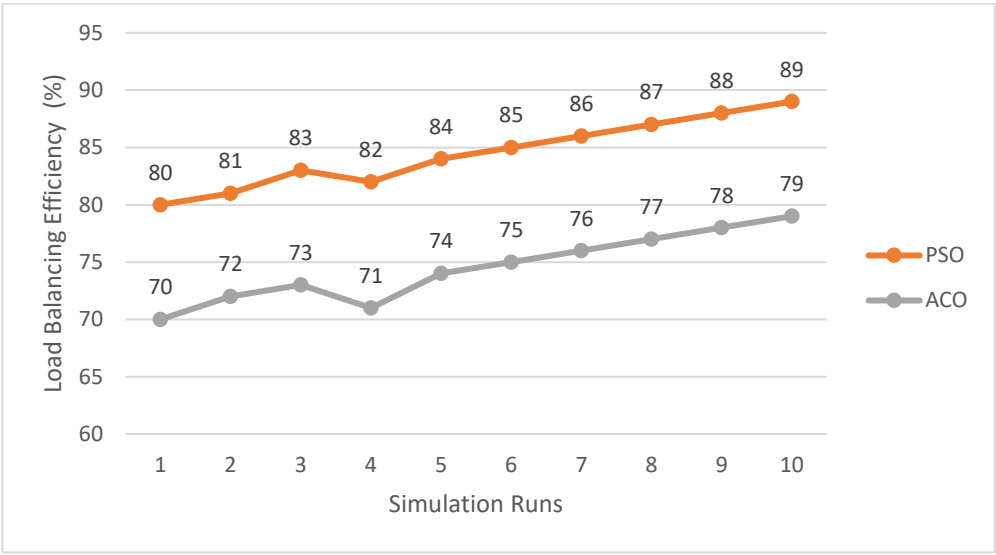

Fig. 5: Load balancing comparison of PSO vs ACO

The simulation results illustrated in Figure (5) confirm that PSO achieved higher load balancing than the ACO, this indicates that PSO is more effective at distributing workloads calmly across available resources. Also, based on the results shown in Figure 5, after multiple iterations, the optimization algorithms gradually regulate and refine their approach, eventually converging to a near-optimal solution, in which the weight is distributed across digital machines. This not only enhances efficiency but also ensures smoother task execution and improves system stability and overall performance.

According to the results displayed in Figure 6, PSO had fewer violations of service level agreements than ACO. This indicates that PSO is more reliable in fulfilling service level agreements, which means that PSO offers superior customer satisfaction, service quality, and contractual compliance. One another view, as optimization algorithms learn how to

balance workloads efficiently, all tasks are better distributed, the SLA violations are reduced, and the required resources are allocated more efficiently, ensuring SLA compliance.

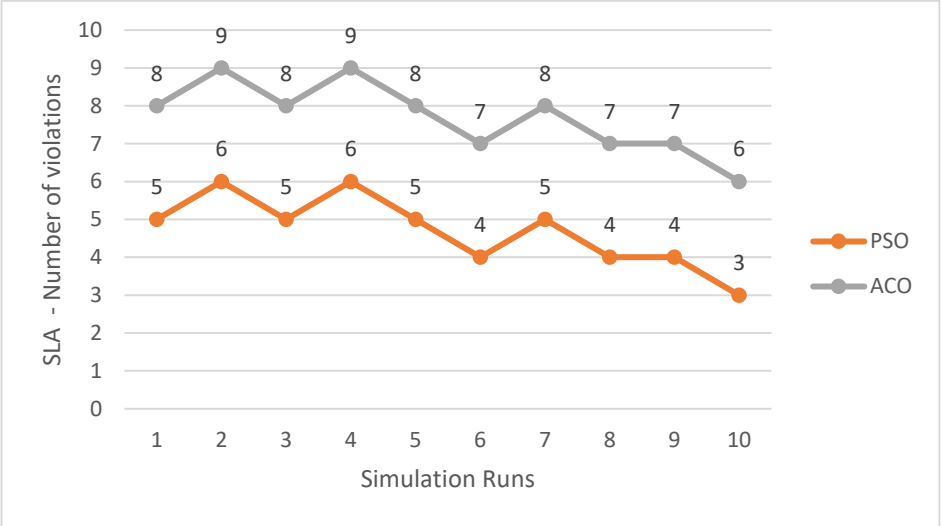

Fig. 6: SLA violations comparison of PSO vs ACO

Figure 7 illustrates that PSO also has a reduced response time compared to ACO. PSO's responses varied from 51 to 60 Milliseconds (ms), whereas ACO's ranged from 61 to 70 Milliseconds (ms). This also confirms that PSO processes tasks more effectively and generates outcomes faster.

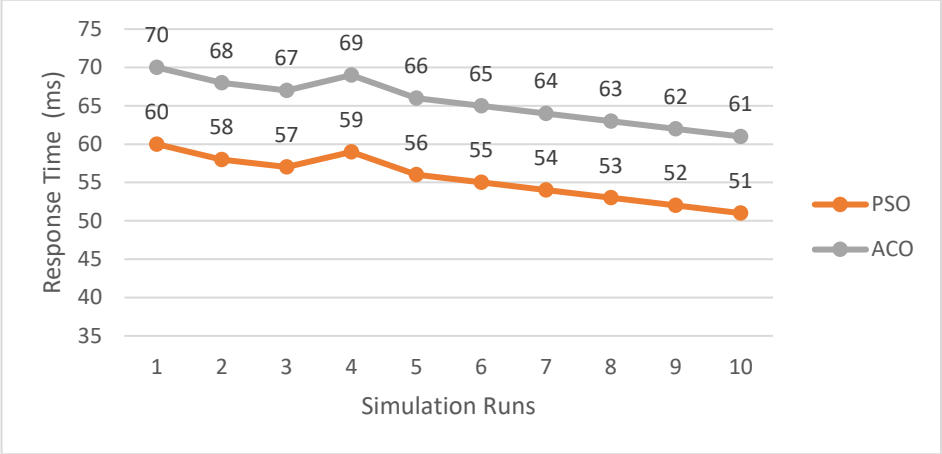

Fig. 7: Response time comparison of PSO vs ACO

Figure 8 illustrates the results of the PSO and ACO Virtual Machine (VM) utilization of over 10 simulation runs.

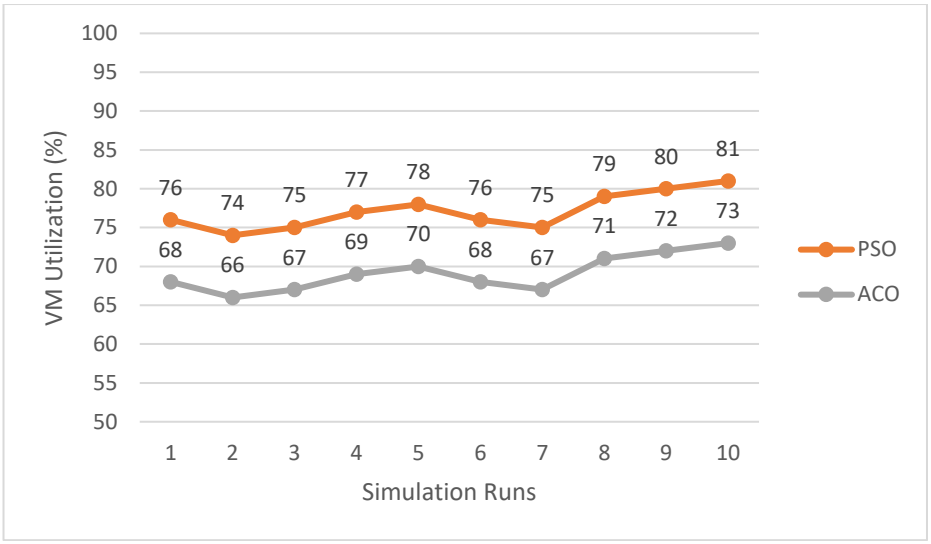

Fig. 8: VM utilization comparison of PSO vs ACO

The trend in Figure (8) shows that PSO continuously achieves better VM utilization as compared to ACO, this confirms that PSO is more powerful in optimizing resource allocation and challenge distribution throughout VMs, leading to better VM utilization. The relatively stable yet increasing pattern in PSO's performance highlights its adaptability in managing computational resources efficiently, while ACO exhibits lower and more fluctuating utilization values.

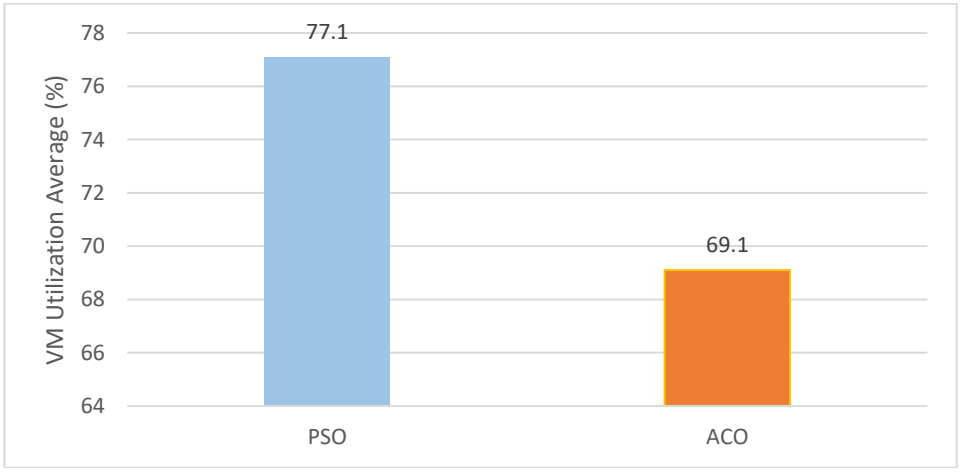

Fig. 9: VM utilization AVG comparison of PSO vs ACO

Figure 9 shows the VM Utilization AVG Comparison of PSO vs ACO, the outcome evidence was strongly in support of PSO which achieved an average utilization of 77.1%, PSO surpassed ACO, which only managed to get 69.1%. This correlates with my previous observations on the figure (6) confirm that PSO improves the efficiency of resource usage while it reduces idle times and improves the performance. Where the higher average utilization proves that PSO effectively balances workloads across available VMs, leading to improved system performance.

Overall, all the simulation experimental results indicate that PSO surpassed ACO in execution duration, energy usage, load distribution, SLA breaches on the contract, and response time. These results were observed in every iteration of the simulation, which indicates the reliability of PSO in different workload scenarios.

6 Conclusion

In this section you should present a conclusion of your work together with future work.

This study proposed a new cloud service composition method based on the PSO algorithm, to improve resource allocation and energy efficiency in cloud computing. As cloud-based e-Gov. services continue to expand and provide more services, effective task scheduling and load balancing become critical for delivering dependable, reliable, cost efficient, and high quality digital public services. other approaches such as ACO have issues related to difficulty with dynamic cloud environments because of their low adaptability, flexibility and convergence speed. According to the results of the VMs average utilization suggests that PSO overcomes the challenges of workload balancing among VMs which results in improved system performance. It also proves the effectiveness of PSO in cloud service composition which is ideal for resource distribution in shifting cloud settings.

As previously mentioned, the evaluation through experiments using CloudSim 5.0 simulator was able to confirm how the PSO approach outperformed ACO in key performance indicators. The identified results were higher VM utilization rates which allowed for more appropriate resource allocation, alongside lower energy consumption, which helped sustain eco-friendly initiatives within the clouds. Moreover, the enhanced scheduling mechanism improved the computational load imbalance problems for virtual machines so that resources were efficiently distributed. These improvements are especially relevant to e-government platforms, where they focus on maintaining uninterrupted service distribution, reducing infrastructure expenditure and improving the accountability of the system to increase citizen satisfaction and confidence in digital government systems, where conclusions expose many benefits for digital government's services, where improved execution and response times enhance the efficiency of citizen transactions, Increased SLA compliance strengthens trust in critical platforms, while energy efficiency supports cost reduction and sustainable IT operations in the public sector, thus improve the e-payment transactions, healthcare scheduling, tax filing and licensing..

The results of this research point out the increased relevance of intelligent optimization algorithms in solving the problems associated with the heightened level of sophistication in cloud computing for e-government purposes. Adaptable and energy aware scheduling will be necessary as the public sector cloud environment grows to include multi-cloud and edge computing. This work showed the impact of PSO algorithm on the optimization of the cloud service composition; however, better results could be obtained from the application of hybrid optimization methods or machine learned predictive resource management. This research aims at creating more efficient, scalable and sustainable cloud infrastructure for e-government, and, consequently, improving digital transformation and public service delivery.

Despite the promising results, there are some limitations of this study. First, the evaluation was performed using CloudSim 5.0, which lacks comprehensive coverage of actual cloud environments ,where the simulation cannot fully capture the real-world heterogeneity, unpredictable workloads, and scaling challenges encountered the e-Gov. cloud platforms. Accordingly, we think that future work should be expanded by deploying a proposed model on hybrid or commercial cloud testbeds, such as AWS or Azure to validate the results under realistic operating conditions.

Second, the optimization work was done on specific set of predefined workloads which makes it impossible to adapt to real-life unpredictable workloads in practical cloud

deployments. Moreover, even if PSO was shown to be better than ACO in a good number of performance coefficients, more work needs to be done on its effectiveness across the variations of service demands and different cloud architectures. Also, the implications of e-government services, such as security limitations, network latency, and data privacy, crucial in evaluation, were not studied in this research. Lastly, although PSO has demonstrated superior performance against ACO, further parameter tuning and hybridization with other metaheuristic techniques may improve its efficiency.

Future research might include multiple directions to extend this study, first is the inclusion of hybrid optimization approaches, integrating PSO and machine learning-based predictive models for improving dynamic service composition. Moreover, extend the proposed model to study multi-cloud and edge computing to make it applicable to actual cloud infrastructure environments. Furthermore, other cloud security policies, regulatory compliance, and privacy-related mechanisms affecting e-government service composition may also be addressed in future studies. Moreover, incorporating accurate prediction models for real-time workloads can enhance resource allocation and scalability, aiding the effective operation of cloud-based public services during fluctuating demand. And finally, Deeper insights might be gained by real-time deployment and testing on commercial cloud platforms like AWS or Azure. Also, Future research should extend the proposed approach to work with multi-cloud and edge environments, and to support various services and applications including IoT and smart cities public services. This expansion must handle challenges of interoperability, data sovereignty, and cross-domain resource allocation to ensure effective and scalable adaptation.

Finally, even though the proposed technical advancements and adopting several algorithms enhance the performance of the e-government cloud. However, deployment may prove ineffective until legal, moral and organizational challenges are addressed, as they are neglected by the trusts, adoption and reduced risks.

ACKNOWLEDGEMENTS

This research is part of the work conducted by the E-Government Services and UX research group at The University of Jordan. It is a collaborative effort by several group members, whose contributions have been instrumental in the successful completion of this study. Also, we confirm that this research was conducted without any financial support from funding agencies, grants, or institutional sponsorships

References

[1] Panayiotou, N. A., & Stavrou, V. P. (2021). Government to business e-services – A systematic literature review. Government Information Quarterly, 38(2), 101576.

[2] Almarabeh, T., Majdalawi, Y. Kh., & Mohammad, H. (2016). Cloud Computing of E-Government. CN, 08(01), 1–8.

[3] Joshi, P., Islam, S., & Islam, S. (2017). A Framework for Cloud Based E-Government from the Perspective of Developing Countries. Future Internet, 9(4), 80.

[4] Abraham, A., Hörandner, F., Zefferer, T., & Zwattendorfer, B. (2020). E-government in the public cloud: requirements and opportunities. EG, 16(3), 260.

[5] Duso, T., & Schiersch, A. (2025). Let's switch to the cloud: cloud usage and its effect on labor productivity. Information Economics and Policy, 101130.

[6] Xiong, L., Wang, J., Yu, L., Xiong, N., & Wu, H. (2025). An Efficient Privacy-Preserving Access Control Scheme for Cloud Computing Services. IEEE Transactions on Consumer Electronics, 1–1.

[7] e-GCloud: Next Wave of E-Government. (2025). FPA, 17(1).

[8] Idahosa, M. E., Eireyi-Edewede, S., & Eruanga, C. E. (2024). Application of Cloud Computing Technology for Enhances E-Government Services in Edo State. In Lytras, M. D., Alkhaldi, A. N., & Ordóñez De Pablos, P. (Eds.), Advances in Electronic Government, Digital Divide, and Regional Development (pp. 221–278). IGI Global.

[9] Tarawneh, H., Alhadid, I., Khwaldeh, S., & Afaneh, S. (2022). An Intelligent Cloud Service Composition Optimization Using Spider Monkey and Multistage Forward Search Algorithms. Symmetry, 14(1), 82.

[10] Wang, N., Luo, F., & Ren, S. (2025). A service composition and optimal selection method considering candidate service capabilities in cloud manufacturing. International Journal of Computer Integrated Manufacturing, 1–18.

[11] Xiao, L., Shan, H., Zhu, J., Mao, R., & Pan, S. (2025). Lightweight cloud service composition and orchestration for edge intelligence. Intelligent Decision Technologies, 1–18.

[12] Arora, A., Karnik, N., Saluja, A., H, V., Pandey, P., & Kaur, A. (2025). Developing an effectual model for improving multi-cloud service composition system. Int J Syst Assur Eng Manag.

[13] Ma, H., Chen, Y., Zhu, H., Zhang, H., & Tang, W. (2018). Optimization of Cloud Service Composition for Data-intensive Applications via E-CARGO. In 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design (CSCWD) (pp. 785–789). IEEE.

[14] Wajid, U., Marin, C. A., & Karageorgos, A. (2013). Optimizing Energy Efficiency in the Cloud Using Service Composition and Runtime Adaptation Techniques. In 2013 IEEE International Conference on Systems, Man, and Cybernetics (pp. 115–120). IEEE.

[15] Nezafat Tabalvandani, M. A., Hosseini Shirvani, M., & Motameni, H. (2024). Reliability-aware web service composition with cost minimization perspective: a multi-objective particle swarm optimization model in multi-cloud scenarios. Soft Comput, 28(6), 5173–5196.

[16] Khwaldeh, S., Abu-taieh, E., Alhadid, I., Alkhawaldeh, R. S., & Masa'deh, R. (2019). DyOrch: Dynamic Orchestrator for Improving Web Services Composition. In Proceedings of the 33rd International Business Information Management Association Conference, IBIMA 2019 (pp. 6030–6047).

[17] Alhadid, I., et al. (2021). Optimizing Service Composition (SC) Using Smart Multistage Forward Search (SMFS). Intelligent Automation & Soft Computing, 28(2), 321–336.

[18] AlHadid, I., & Abu-Taieh, E. (2018). Web Services Composition Using Dynamic Classification and Simulated Annealing. MAS, 12(11), 395.

[19] Masdari, M., Nozad Bonab, M., & Ozdemir, S. (2021). QoS-driven metaheuristic service composition schemes: a comprehensive overview. Artificial Intelligence Review, 54(5), 3749–3816.

[20] Nazif, H., Nassr, M., Al-Khafaji, H. M. R., Jafari Navimipour, N., & Unal, M. (2023). A cloud service composition method using a fuzzy-based particle swarm optimization algorithm. Multimedia Tools and Applications, 83(19), 56275–56302.

[21] Sreeramulu, M. D., Mohammed, A. S., Kalla, D., Boddapati, N., & Natarajan, Y. (2024). AI-driven Dynamic Workload Balancing for Real-time Applications on Cloud Infrastructure. In 2024 7th International Conference on Contemporary Computing and Informatics (IC3I) (pp. 1660–1665). IEEE.

[22] Abualigah, L., et al. (2024). Particle swarm optimization algorithm: review and applications. In Metaheuristic Optimization Algorithms (pp. 1–14). Elsevier.

[23] Jain, M., Saihjpal, V., Singh, N., & Singh, S. B. (2022). An Overview of Variants and Advancements of PSO Algorithm. Applied Sciences, 12(17), 8392.

[24] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In Proceedings of ICNN'95 - International Conference on Neural Networks (pp. 1942–1948). IEEE.

[25] Liang, H. T., & Kang, F. H. (2016). Adaptive mutation particle swarm algorithm with dynamic nonlinear changed inertia weight. Optik, 127(19), 8036–8042.

[26] Dordaie, N., & Navimipour, N. J. (2018). A hybrid particle swarm optimization and hill climbing algorithm for task scheduling in the cloud environments. ICT Express, 4(4), 199–202.

[27] Liu, Q., Wei, W., Yuan, H., Zhan, Z.-H., & Li, Y. (2016). Topology selection for particle swarm optimization. Information Sciences, 363, 154–173.

[28] Guntsch, M., & Middendorf, M. (2002). A Population Based Approach for ACO. In Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., & Raidl, G. R. (Eds.), Applications of Evolutionary Computing (Lecture Notes in Computer Science, Vol. 2279, pp. 72–81). Springer.

[29] Stutzle, T., & Dorigo, M. (1999). ACO algorithms for the traveling salesman problem. Evolutionary algorithms in engineering and computer science, 4, 163–183.

[30] Yin, C., Li, S., & Li, X. (2024). An optimization method of cloud manufacturing service composition based on matching-collaboration degree. International Journal of Advanced Manufacturing Technology, 131(1), 343–353.

[31] Bei, L., Wenlin, L., Xin, S., & Xibin, X. (2024). An improved ACO based service composition algorithm in multi-cloud networks. Journal of Cloud Computing, 13(1), 17.

[32] Prity, F. S., Uddin, K. M. A., & Nath, N. (2024). Exploring swarm intelligence optimization techniques for task scheduling in cloud computing: algorithms, performance analysis, and future prospects. Iran Journal of Computer Science, 7(2), 337–358.

[33] Gupta, S., & Tripathi, S. (2023). A comprehensive survey on cloud computing scheduling techniques. Multimedia Tools and Applications, 83(18), 53581–53634.

[34] Aslanpour, M. S., Gill, S. S., & Toosi, A. N. (2020). Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research. Internet of Things, 12, 100273.

[35] Kumar, M., Sharma, S. C., Goel, A., & Singh, S. P. (2019). A comprehensive survey for scheduling techniques in cloud computing. Journal of Network and Computer Applications, 143, 1–33.

[36] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., & Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience, 41(1), 23–50.

[37] Choudhary, V., & Vithayathil, J. (2013). The Impact of Cloud Computing: Should the IT Department Be Organized as a Cost Center or a Profit Center? Journal of Management Information Systems, 30(2), 67–100.

[38] Assudani, P., & Abimannan, S. (2018). Cost Efficient Resource Scheduling in Cloud Computing: a Survey. International Journal of Engineering & Technology, 7(4), 38–43.

[39] Katal, A., Dahiya, S., & Choudhury, T. (2023). Energy efficiency in cloud computing data centers: a survey on software technologies. Cluster Computing, 26(3), 1845–1875.

[40] Mao, M., & Humphrey, M. (2012). A Performance Study on the VM Startup Time in the Cloud. In 2012 IEEE Fifth International Conference on Cloud Computing (pp. 423–430). IEEE.

[41] Yakubu, I. Z., Musa, Z. A., Muhammed, L., Ja'afaru, B., Shittu, F., & Matinja, Z. I. (2020). Service Level Agreement Violation Preventive Task Scheduling for Quality of Service Delivery in Cloud Computing Environment. Procedia Computer Science, 178, 375–385.

[42] Ahmad, M., & Abawajy, J. H. (2014). Service Level Agreements for the Digital Library. Procedia - Social and Behavioral Sciences, 147, 237–243.

[43] Qazi, F., Kwak, D., Khan, F. G., Ali, F., & Khan, S. U. (2024). Service Level Agreement in cloud computing: Taxonomy, prospects, and challenges. Internet of Things, 25, 101126.

[44] Ghandour, O., El Kafhali, S., & Hanini, M. (2024). Adaptive workload management in cloud computing for service level agreements compliance and resource optimization. Computers and Electrical Engineering, 120, 109712.

[45] Ala'anzy, M., & Othman, M. (2019). Load Balancing and Server Consolidation in Cloud Computing Environments: A Meta-Study. IEEE Access, 7, 141868–141887.

[46] Devi, N., et al. (2024). A systematic literature review for load balancing and task scheduling techniques in cloud computing. Artificial Intelligence Review, 57(10), 276.

[47] Shah, S. K., & Nautiyal, A. (2022). Task Scheduling and Load Balancing for Minimization of Response Time in IoT Assisted Cloud Environments. In 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon) (pp. 1–7). IEEE.

[48] Shafiq, D. A., Jhanjhi, N. Z., & Abdullah, A. (2022). Load balancing techniques in cloud computing environment: A review. Journal of King Saud University - Computer and Information Sciences, 34(7), 3910–3933.

[49] Milani, A. S., & Navimipour, N. J. (2016). Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends. Journal of Network and Computer Applications, 71, 86–98.

[50] Kadarla, K., Sharma, S. C., Bhardwaj, T., & Chaudhary, A. (2017). A Simulation Study of Response Times in Cloud Environment for IoT-Based Healthcare Workloads. In 2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS) (pp. 678–683). IEEE.

[51] Jayaswal, C. J., & Bindulal, P. (2023). Edge computing: Applications, Challenges and Opportunities. *JoCTA*.

[52] Zhou, Z., Li, F., Zhu, H., Xie, H., Abawajy, J. H., & Chowdhury, M. U. (2020). An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. *Neural Computing & Applications, 32*(6), 1531–1541.

[53] Antonopoulos, N., & Gillam, L. (Eds.). (2017). *Cloud Computing: Principles, Systems and Applications*. Cham: Springer International Publishing.

[54] Sharma, N., Sonal, & Garg, P. (2022). Ant colony based optimization model for QoS-based task scheduling in cloud computing environment. *Measurement: Sensors, 24*, 100531.

[55] Jula, A., Sundararajan, E., & Othman, Z. (2014). Cloud computing service composition: A systematic literature review. *Expert Systems with Applications, 41*(8), 3809–3824.

[56] Kadhim, Q. K., Yusof, R., Mahdi, H. S., Al-shami, S. S. A., & Selamat, S. R. (2018). A Review Study on Cloud Computing Issues. *J. Phys.: Conf. Ser., 1018*, 012006.

[57] Saberikia, M., Farbeh, H., & Fazeli, M. (2025). Improving energy efficiency and fault tolerance of mission-critical cloud task scheduling: A mixed-integer linear programming approach. *Sustainable Computing: Informatics and Systems, 45*, 101068.

[58] Pallathadka, H., et al. (2022). An investigation of various applications and related challenges in cloud computing. *Materials Today: Proceedings, 51*, 2245–2248.

[59] Razdar, M., Adibi, M. A., & Haleh, H. (2025). An Optimization of multi-level multi-objective cloud production systems with meta-heuristic algorithms. *Decision Analytics Journal, 14*, 100540.

[60] Li, H., Liu, L., Duan, X., Li, H., Zheng, P., & Tang, L. (2024). Energy-efficient offloading based on hybrid bio-inspired algorithm for edge–cloud integrated computation. *Sustainable Computing: Informatics and Systems, 42*, 100972.

[61] Khaleel, M. I. (2025). The application of hybrid spider monkey optimization and fuzzy self-defense algorithms for multi-objective scientific workflow scheduling in cloud computing. *Internet of Things, 30*, 101517.

[62] Shao, K., Fu, H., & Wang, B. (2023). An Efficient Combination of Genetic Algorithm and Particle Swarm Optimization for Scheduling Data-Intensive Tasks in Heterogeneous Cloud Computing. *Electronics, 12*(16), 3450.

[63] Cheikh, S., & Walker, J. J. (2021). Solving Task Scheduling Problem in the Cloud Using a Hybrid Particle Swarm Optimization Approach. *International Journal of Applied Metaheuristic Computing, 13*(1), 1–25.

[64] Kurdi, H., Ezzat, F., Altoaimy, L., Ahmed, S. H., & Youcef-Toumi, K. (2018). MultiCuckoo: Multi-Cloud Service Composition Using a Cuckoo-Inspired Algorithm for the Internet of Things Applications. *IEEE Access, 6*, 56737–56749.

[65] Wei, X. (2020). Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing. *Journal of Ambient Intelligence and Humanized Computing*.

[66] Chahal, P. K., Kumar, K., & Soodan, B. S. (2024). Grey wolf algorithm for cost optimization of cloud computing repairable system with N-policy, discouragement and two-level Bernoulli feedback. *Mathematics and Computers in Simulation, 225*, 545–569.

[67] Behera, I., & Sobhanayak, S. (2024). Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach. *Journal of Parallel and Distributed Computing, 183*, 104766.

[68] Yu, N., Zhang, A.-N., Chu, S.-C., Pan, J.-S., Yan, B., & Watada, J. (2025). Innovative Approaches to Task Scheduling in Cloud Computing Environments Using an Advanced Willow Catkin Optimization Algorithm. *CMC, 82*(2), 2495–2520.

[69] Pradhan, A., Bisoy, S. K., & Das, A. (2022). A survey on PSO based meta-heuristic scheduling mechanism in cloud computing environment. *Journal of King Saud University - Computer and Information Sciences, 34*(8), 4888–4901.

[70] Dubey, K., & Sharma, S. C. (2021). A novel multi-objective CR-PSO task scheduling algorithm with deadline constraint in cloud computing. *Sustainable Computing: Informatics and Systems, 32*, 100605.

[71] Fidanova, S. (2021). Ant Colony Optimization. In *Ant Colony Optimization and Applications* (Studies in Computational Intelligence, Vol. 947, pp. 3–8). Springer.

[72] Tawfeek, M. A., El-Sisi, A., Keshk, A. E., & Torkey, F. A. (2013). Cloud task scheduling based on ant colony optimization. In *2013 8th International Conference on Computer Engineering & Systems (ICCES)* (pp. 64–69). IEEE.

[73] Chandrashekar, C., Krishnadoss, P., Kedalu Poornachary, V., Ananthakrishnan, B., & Rangasamy, K. (2023). HWACOA Scheduler: Hybrid Weighted Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing. *Applied Sciences, 13*(6), 3433.

[74] Liu, H. (2022). Research on cloud computing adaptive task scheduling based on ant colony algorithm. *Optik, 258*, 168677.

[75] CloudSim 5.0. (2025). Cloud Computing Projects | Source Code | Research Topics. Available: https://cloudcomputingprojects.net/how-can-i-learn-cloudsim/download-cloudsim-4-0/

[76] Ab. Aziz, N. A., Ibrahim, Z., Mubin, M., Nawawi, S. W., & Mohamad, M. S. (2018). Improving particle swarm optimization via adaptive switching asynchronous – synchronous update. *Applied Soft Computing, 72*, 298–311.