

Article

SPES: A Stochastic Predictive Energy-Aware Scheduling Approach for Efficient Multi-Region Cloud Computing

Mohamed F. Yacoub ^{1,2,*} , Ahmed E. Abdel Raouf ¹, Walaa Gad ¹  and Nagwa L. Badr ^{1,3}

¹ Information Systems Department, Faculty of Computer and Information Sciences, Ain Shams University, Cairo 11566, Egypt; ahmed_ezzat@cis.asu.edu.eg (A.E.A.R.); walaagad@cis.asu.edu.eg (W.G.); nagwabadr@cis.asu.edu.eg (N.L.B.)

² Information Systems Department, Faculty of Computers and Information, Damanhour University, Damanhour 22511, Egypt

³ University of East London at EUE, New Administrative Capital 11846, Egypt

* Correspondence: mohamedyacoub@cis.dmu.edu.eg

Abstract

Cloud computing has transformed the delivery of modern applications and services by providing scalable, flexible, and cost-effective access to computing resources. One of the most critical challenges in cloud environments is the efficient distribution of dynamic workloads across heterogeneous resources, commonly addressed through load balancing and task scheduling techniques. Efficient scheduling plays a vital role in maximizing resource utilization, minimizing response time, and maintaining acceptable Quality of Service (QoS), particularly under dynamic and large-scale workloads. Despite the progress achieved by traditional heuristics such as Min-Min and metaheuristic approaches like the Improved Sparrow Search Algorithm (ISSA), challenges related to scalability, adaptability, and computational overhead remain. Metaheuristic-based approaches often involve iterative optimization processes that may limit their efficiency in real-time scheduling scenarios. In this paper, we propose a lightweight Stochastic Predictive Energy-Aware Scheduling (SPES) algorithm that integrates predictive execution estimation, multi-resource awareness, and stochastic decision-making. Unlike deterministic scheduling strategies, SPES employs a Top K candidate selection mechanism combined with probabilistic weighting and epsilon-greedy exploration to enhance adaptability and avoid suboptimal resource allocation. The proposed method considers CPU, memory, and I/O demands to achieve balanced utilization across heterogeneous hosts while implicitly addressing energy efficiency through utilization-based modeling. The proposed algorithm is implemented and evaluated using the CloudSim 5.0 simulation framework under heterogeneous multi-region cloud environments with varying workload sizes. Experimental results demonstrate that SPES consistently outperforms ISSA and achieves makespan reductions of up to 23.8% while improving scalability, resource utilization, and scheduling efficiency under dynamic cloud workloads. These results indicate that SPES provides an effective lightweight scheduling solution for large-scale and energy-aware cloud computing environments and supports green computing objectives through improved resource efficiency.

Keywords: cloud computing; CloudSim; task scheduling; resource allocation; load balancing; energy-aware computing; green computing; multi-region cloud computing



Academic Editor: Ming-Shen Jian

Received: 4 June 2026

Revised: 5 July 2026

Accepted: 6 July 2026

Published: 16 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons](https://creativecommons.org/licenses/by/4.0/)

[Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Cloud computing has emerged as one of the most influential paradigms in modern computing, enabling scalable, flexible, and on-demand access to computational resources over the Internet [1,2]. The rapid growth of cloud-based applications, big data analytics, Internet of Things (IoT), artificial intelligence, and online services, has significantly increased the demand for reliable and efficient cloud infrastructures [3,4]. Cloud computing provides organizations and individuals with cost-effective access to storage, processing power, networking, and virtualized services without requiring large investments in physical infrastructure. The underlying principles of cloud computing, including virtualization, resource sharing, service elasticity, and on-demand provisioning, have established the foundation for scalable and efficient cloud infrastructures [5]. This paradigm has fundamentally transformed how computational resources are deployed, managed, and consumed across both academic and industrial environments [6]. This evolution has established cloud computing as the fifth utility, enabling ubiquitous access to computing resources through service-oriented delivery models [7]. At the core of cloud computing are cloud data centers, which host large numbers of heterogeneous physical machines and virtual machines responsible for executing dynamic workloads and providing uninterrupted services to users worldwide [8].

Despite the significant advantages offered by cloud computing, efficient resource allocation and workload management remain major challenges [9,10]. Modern cloud environments are characterized by dynamic workload arrivals, heterogeneous resource capacities, geographically distributed infrastructures, and continuously changing utilization conditions. Dynamic resource management techniques such as auto-scaling have also been extensively investigated to adapt resource provisioning according to workload fluctuations while minimizing execution cost and meeting application deadlines [11]. These characteristics make task scheduling and load balancing highly complex optimization problems [12]. Various scheduling strategies, ranging from traditional heuristic methods to workflow-oriented scheduling algorithms, have been extensively investigated to improve resource allocation efficiency in heterogeneous cloud environments [13,14]. In large-scale multi-region cloud environments, workloads may need to be distributed across several geographically separated data centers to maintain high availability, minimize response time, and improve Quality of Service (QoS) [15]. Consequently, inefficient scheduling decisions can lead to resource underutilization, excessive task waiting time, increased makespan, which represents the total completion time of all submitted tasks, performance degradation, and unnecessary energy consumption [16].

Load balancing therefore plays a critical role in improving the efficiency and reliability of cloud systems [17]. The objective of load balancing is to distribute workloads dynamically across available computing resources in a way that prevents resource overloading while maximizing utilization and throughput [18]. Effective scheduling and load balancing mechanisms can significantly reduce execution time, minimize latency, improve fault tolerance, and enhance the overall user experience [19]. However, achieving balanced resource allocation becomes increasingly difficult in heterogeneous cloud infrastructures where hosts possess different CPU, memory, storage, and I/O capabilities [20]. In addition, the increasing scale of cloud data centers introduces further challenges related to scalability, adaptability, and energy efficiency [21].

Traditional scheduling approaches such as First Come First Serve (FCFS), Round Robin, Min-Min, and Max-Min are widely used due to their simplicity and low computational overhead [15,16]. Nevertheless, these approaches often fail to adapt effectively to dynamic workload patterns and heterogeneous resource conditions. Consequently, metaheuristic optimization techniques have attracted increasing attention as effective optimization ap-

proaches for solving complex cloud scheduling problems [22]. As cloud systems become more complex, deterministic scheduling methods may generate suboptimal resource allocation decisions and fail to maintain balanced utilization across hosts [23]. To address these limitations, researchers have explored metaheuristic and swarm intelligence techniques inspired by natural phenomena, including Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Genetic Algorithms (GA), Whale Optimization Algorithm (WOA), and Sparrow Search Algorithm (SSA) [18–20]. These techniques aim to improve scheduling quality by searching for near-optimal solutions in large and complex solution spaces. More recently, hybrid differential evolution-based optimization methods and neighborhood search techniques have further improved multi-objective cloud scheduling performance under heterogeneous environments [24,25].

Among these approaches, the Improved Sparrow Search Algorithm (ISSA) has demonstrated promising performance in cloud scheduling and resource allocation problems [21]. ISSA enhances workload distribution and improves resource utilization through adaptive swarm-based optimization strategies. However, population-based metaheuristic approaches such as ISSA generally require iterative exploration and repeated fitness evaluations, which may increase computational overhead and reduce responsiveness under real time workload conditions [23,26]. Furthermore, the stochastic behavior of such algorithms may lead to convergence instability or excessive exploration in highly dynamic multi-region cloud environments. These limitations motivate the need for lightweight scheduling strategies capable of maintaining efficient performance while reducing computational complexity. Recent studies have incorporated dynamic execution time prediction to improve scheduling quality and reduce makespan while maintaining efficient task allocation [27].

Another important challenge in modern cloud computing is energy consumption [28]. Large-scale cloud data centers consume considerable amounts of electrical power, contributing to operational costs and environmental concerns. Consequently, green computing and energy-aware scheduling have become major research directions in cloud resource management [29]. Several studies have attempted to reduce power consumption by integrating energy models, workload consolidation techniques, predictive resource allocation, and adaptive scheduling mechanisms [30,31]. Load estimation techniques and intelligent resource provisioning strategies have also been proposed to improve resource efficiency under dynamic cloud and edge computing environments [32]. In parallel, trust-aware multi-objective scheduling algorithms have been introduced to balance system performance with reliable resource allocation decisions [33]. However, many energy-aware approaches rely on complex optimization models or machine learning frameworks that may introduce additional computational overhead and reduce scalability in real-time scheduling environments.

To overcome these limitations, this paper proposes the Stochastic Predictive Energy-Aware Scheduling (SPES) algorithm, a lightweight and adaptive scheduling approach designed for heterogeneous multi-region cloud environments. The proposed algorithm integrates predictive resource estimation, multi-resource utilization analysis, stochastic decision-making, and energy-aware scheduling into a unified scheduling framework. Unlike purely deterministic or computationally expensive metaheuristic approaches, SPES employs a top k candidate host selection mechanism combined with probabilistic weighted selection and epsilon-greedy exploration to improve adaptability and avoid suboptimal allocation decisions. The scheduler evaluates CPU, memory, and I/O utilization simultaneously to achieve balanced workload distribution across heterogeneous hosts while minimizing overload conditions.

To support green computing objectives, SPES incorporates a lightweight energy-aware model using CPU utilization as an energy proxy, enabling implicit optimization of power

consumption without requiring complex power estimation models. In addition, the algorithm dynamically adapts its scheduling priorities according to current host utilization conditions, allowing efficient workload balancing under varying system loads. The proposed SPES algorithm is implemented and evaluated using the CloudSim simulation framework in a distributed multi-region cloud environment consisting of heterogeneous hosts and multiple workload sizes. Experimental results demonstrate that SPES consistently outperforms ISSA in terms of makespan, total execution time, and average execution time. SPES achieves makespan improvements of up to 23.8% while maintaining efficient resource utilization and reduced scheduling overhead under dynamic workload conditions.

The technical novelty of SPES lies in the integration of predictive execution estimation, multi-resource utilization awareness, adaptive dynamic weighting, Top K candidate host selection, and ϵ -greedy stochastic exploration within a unified lightweight scheduling approach. Unlike conventional population-based metaheuristic schedulers, SPES combines these mechanisms in a single scheduling decision without iterative optimization, enabling efficient workload distribution while maintaining low computational overhead and good scalability in heterogeneous multi-region cloud environments.

The remainder of this paper is organized as follows. Section 2 reviews related work and discusses existing heuristic and metaheuristic scheduling approaches. Section 3 presents the proposed SPES methodology, including its design principles and algorithmic formulation. Section 4 evaluates the performance of SPES through simulation experiments and comparative analysis with ISSA. Finally, Section 5 concludes the paper and outlines future research directions.

2. Related Work

Recent research in cloud computing has extensively explored resource allocation and load balancing techniques to address challenges related to scalability, energy efficiency, and workload distribution. As summarized in Table 1, a variety of approaches have been proposed, ranging from heuristic and metaheuristic algorithms to hybrid and learning-based methods. These studies are typically evaluated using simulation environments such as CloudSim 5.0 or Python-based frameworks, enabling controlled analysis of scheduling performance under diverse workload conditions.

Table 1. Comparative study of related work.

Ref.	Approach	Focus	Used Techniques	Simulation Environment	Dataset	Contribution
[34]	Multi-objective Survivors Optimization (MO-survivors' optimization)	Load balancing; Multi-objective optimization; Resource management	Hybrid bio-inspired metaheuristic (Rattus social behavior + Lapin hide-and-seek + Hawks group hunt)	Python simulation	Google Cloud Jobs Dataset (GoCJ)	Reduced makespan by 223.03 s while improving resource utilization by 5.36% and throughput by 1.77%.
[35]	DA-DBL Federated Learning with RO-COA (Dilated & Attention Bi-LSTM with Random Opposition Coati Optimization)	Resource prediction & Load balancing	Federated Learning; Dilated & Attention 1D-CNN (DA-DBL); Bi-LSTM; RO-COA optimization; multi-objective constraints (makespan, QoS, resource utilization, power)	Python simulation	N/A	Achieved 75–80% reduction in the cost function compared with TSO, FOA, and CSA metaheuristics, with improved prediction accuracy and power efficiency.
[36]	Intercrossed Chimp & Bald Eagle Algorithm (IC&BA)	Load balancing, energy, execution, migration cost	Hybrid of Bald Eagle Search (BES) and Chimp Optimization (CA)	Python simulation	Google Cluster Workload Traces 2019	Improved load balancing performance while reducing energy consumption in heterogeneous cloud environments.
[23]	Region-Aware Dynamic Scheduling Algorithm (RASA)	Workload distribution; latency reduction	Adaptive Chaotic Sparrow Search Optimization (ACSSA), Coalitional Game	CloudSim Toolkit	Run-time generated tasks	Enhanced task scheduling precision and reduced latency through adaptive workload distribution.

Table 1. Cont.

Ref.	Approach	Focus	Used Techniques	Simulation Environment	Dataset	Contribution
[37]	Intelligent Weighted PSO with Feedback Controller	Task scheduling; VM migration	PSO + Double Deep Q model + Conditional GAN feedback	Python (Windows 10, i3, 8 GB RAM, Spyder IDE)	N/A	Provided fault-tolerant and energy-efficient load balancing with optimized workload management.
[38]	EcoSched Framework	Resource allocation; eco-efficiency	Predictive analytics-based scheduling	CloudSim + VMware + AWS; Python; Intel i7, 16 GB RAM	OpenVC Dataset	Improved scalability, energy efficiency, and cost effectiveness for large-scale cloud scheduling.
[39]	EMPBT-LB Algorithm	Dynamic load balancing	Non-AI heuristic method	4 Data Centers, 6 Regions, Linux + Xen	Real-time workload data	Enhanced response time and overall system performance under dynamic workloads.
[40]	LBAO (Load Balance-Aware Offloading)	Cloud–Edge–End (CEE) collaboration	Genetic Algorithm with heuristic decoding	Custom GA-based simulator	GitHub (fanyueqi/LBAO-algorithm)	Achieved efficient cloud–edge–end task offloading with improved dynamic load balancing capability.
[41]	PMHEFT Review (Comparative Study)	Benchmarking of load balancing algorithms	Comparison: FIFO, PHMEFT, DRALBA, GWO-PSO, Markov	Review (no simulation)	N/A	Identified PMHEFT as the best-performing approach for improving throughput and minimizing response time.
[42]	ReT-ELBa Algorithm	Load balancing efficiency	Modified Throttled Algorithm with Priority	Cloud Analyst Tool	4 Cases ([50, 50, 100, and 150] Virtual Machines, [1 (25), 2(25 Each), 2(50 Each), and 3(50 Each)] Data Centers).	Optimized virtual machine assignment and reduced response time in cloud data centers.
[43]	Hybrid Lyrebird Falcon Optimization (HLFO)	Energy & memory optimization	FOA + LOA + Reinforcement Learning	CloudSim 4.0 + Python 3.11.1	Real world data	Improved energy efficiency through adaptive and intelligent scheduling mechanisms.
[44]	Modified Min-Min Heuristic	Workflow scheduling; precedence constraints	Modified Min-Min heuristic	Simulation on benchmark workflows	– Random (100 to 500 tasks) – Montage workflows (25 to 100 tasks)	Enhanced resource utilization and reduced overall task execution time.

Metaheuristic and hybrid optimization approaches have gained significant attention due to their ability to handle complex scheduling problems. Algorithms such as Multi-objective Survivors Optimization (MO-survivors) [34], IC&BA [36], and RASA [23] incorporate bio-inspired and game-theoretic concepts to improve load balancing, reduce makespan, and enhance resource utilization. Similarly, the federated learning-based ROCOA framework [35] integrates attention mechanisms with Coati Optimization to improve adaptive load balancing and resource allocation in dynamic cloud environments. Likewise, the MO-Survivors approach [34] performs multi-objective optimization to simultaneously improve load balancing, resource utilization, and scheduling efficiency. Although both approaches achieve promising performance, they rely on relatively sophisticated optimization or AI-based learning mechanisms, which may increase computational complexity in large-scale cloud environments. Other approaches, including PSO-based feedback models [37], EcoSched [38], and EMPBT-LB [39], aim to improve fault tolerance, energy efficiency, and response time through hybrid and adaptive strategies.

Despite their effectiveness, these approaches often rely on complex optimization processes, parameter tuning, or iterative convergence mechanisms, which may limit their applicability in real-time and large-scale cloud environments. Additionally, cloud–edge and hybrid computing paradigms [40] introduce further complexity due to the need for distributed and region-aware resource management.

Traditional heuristic scheduling algorithms remain attractive because of their simplicity and low computational overhead. However, their limited adaptability often leads to suboptimal scheduling decisions in heterogeneous cloud environments [15–17]. To over-

come these limitations, recent studies have increasingly adopted metaheuristic and swarm intelligence techniques, particularly ISSA [21], to improve scheduling quality and resource utilization. Although these approaches achieve promising optimization performance, their population-based iterative search mechanisms may introduce additional computational overhead and reduce responsiveness under dynamic cloud workloads [23,26].

Game-theoretic approaches have also been explored to improve scheduling efficiency with reduced computational complexity. For instance, the Regional Awareness Dynamic Scheduling Algorithm (RASA) [23] integrates coalitional game theory with adaptive optimization techniques to improve throughput and latency. However, these methods often require complex modeling and parameter tuning, which can limit their flexibility in rapidly changing environments.

More recently, machine learning and artificial intelligence techniques have been introduced to enhance scheduling decisions. Predictive analytics [45] and deep reinforcement learning (DRL) [46] enable adaptive and data-driven resource allocation strategies, while other approaches focus on energy-efficient scheduling using collective intelligence models such as the Krill Herd algorithm [47]. In addition, Lyu et al., [48] proposed a quantization-aware collaborative inference framework for large embodied AI models to improve inference efficiency under delay and energy constraints, whereas Reddy et al., [49] presented a secure and energy-efficient cloud computing framework that combines encryption, intelligent data center selection, and energy-aware resource management. Although these approaches improve prediction accuracy, energy efficiency, and system intelligence, they generally address complementary cloud computing challenges rather than lightweight task scheduling and may introduce additional computational complexity or require advanced AI models.

Based on the reviewed literature, several research gaps remain. Classical heuristics lack scalability and adaptability for dynamic and large-scale cloud environments. Metaheuristic and hybrid approaches improve optimization quality but often suffer from high computational cost and slow convergence. Swarm-based algorithms, including ISSA, provide improved load balancing but may struggle with responsiveness under dynamic workloads. Furthermore, while energy-aware and predictive scheduling techniques have been explored, integrating these features into a lightweight and efficient real-time scheduling framework remains a significant challenge.

To address these limitations, this paper proposes the Stochastic Predictive Energy-Aware Scheduling (SPES) algorithm, which combines lightweight heuristic design with stochastic decision-making and predictive resource estimation. Unlike existing approaches, SPES integrates multi-resource awareness (CPU, memory, and I/O), dynamic load balancing, and implicit energy optimization, enabling efficient and scalable scheduling suitable for modern cloud environments.

3. Proposed Methodology

The proposed Stochastic Predictive Energy-Aware Scheduling (SPES) algorithm is designed to achieve efficient, balanced, and energy-aware task scheduling in heterogeneous distributed cloud environments. The approach aims to bridge the gap between predictive modeling and real-time scheduling by integrating execution time estimation, multi-resource load balancing, and energy awareness into a unified probabilistic decision-making framework. Unlike traditional deterministic heuristics or complex metaheuristic approaches, SPES adopts a computationally efficient stochastic strategy that dynamically adapts to varying workload conditions while maintaining minimal algorithmic overhead. The scheduling problem addressed by SPES can be formulated as a constrained multi-objective host selection problem. For each incoming cloudlet, the scheduler aims to identify

the most appropriate host by simultaneously minimizing predicted execution time, resource imbalance, and estimated energy consumption while satisfying the available CPU, memory, and I/O resource constraints. Instead of solving a computationally expensive mathematical optimization model, SPES employs a lightweight heuristic scoring mechanism combined with stochastic candidate selection to efficiently approximate the optimal scheduling decision under dynamic heterogeneous cloud environments.

3.1. Multi-Layer Scheduling Architecture

As illustrated in Figure 1, the SPES model operates within a multi-layer scheduling architecture composed of four main layers, Task Layer, Resource Layer, Scheduling Layer, and Execution Layer.

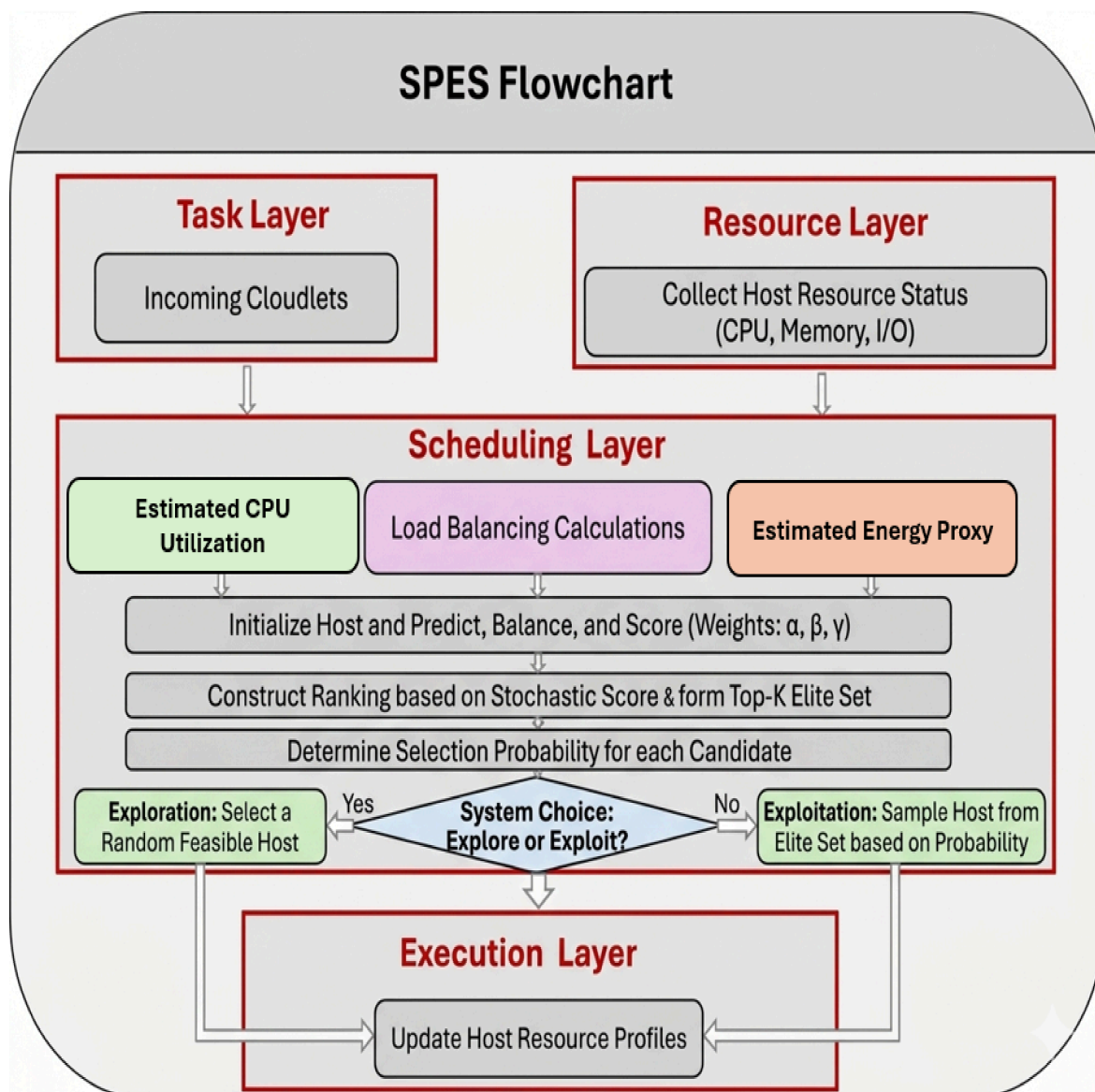


Figure 1. The Proposed SPES Algorithm.

The Task Layer is responsible for defining and profiling incoming cloudlets based on their computational requirements, including CPU, memory, and I/O demands. This profiling enables context-aware scheduling decisions. The Resource Layer represents the heterogeneous infrastructure of cloud data centers distributed across multiple regions. It continuously monitors and reports real-time utilization metrics of hosts, forming the

basis for scheduling decisions. The Scheduling Layer implements the core SPES logic. It evaluates candidate host-task mappings using a multi-objective scoring function derived from predictive and real-time system parameters. The Execution Layer handles task execution and updates host utilization dynamically, forming a feedback loop that ensures continuous adaptation to system changes.

3.2. Mathematical Modeling

To provide a clear understanding of the mathematical framework used in the proposed SPES algorithm, Table 2 summarizes the notations, parameters, and variables deployed throughout Equations (1)–(10). The model evaluates scheduling decisions based on multi-resource utilization metrics (CPU_u, MEM_u, and IO_u), statistical properties of host loads (μ and σ^2), and adaptive scoring coefficients (α , β , and γ). Additionally, the parameters governing the stochastic candidate selection, weight normalization, and exploration–exploitation trade-off (δ , ε , and k) are detailed to ensure the reproducibility of the scheduling strategy.

Table 2. Mathematical notations and parameters of the proposed SPES algorithm.

Symbol	Description
h	A specific candidate host in the cloud infrastructure.
CPU_u	Estimated total CPU utilization of host h after task assignment.
$CPU_{current}(h)$	Current CPU capacity currently utilized by active workloads on host h .
CPU_{task}	CPU capacity required by the incoming task (cloudlet).
$CPU_{capacity}(h)$	Maximum CPU capacity supported by host h .
MEM_u	Estimated total Memory utilization of host h after task assignment.
$MEM_{current}(h)$	Memory capacity currently utilized on host h .
MEM_{task}	Memory capacity required by the incoming task.
$MEM_{capacity}(h)$	Maximum Memory capacity supported by host h .
IO_u	Estimated total Input/Output (I/O) utilization of host h after task assignment.
$IO_{current}(h)$	I/O currently utilized on host h .
IO_{task}	I/O required by the incoming task.
$IO_{capacity}(h)$	Maximum I/O capacity supported by host h .
μ	Mean resource utilization across CPU, Memory, and I/O for a given host.
σ^2	Imbalance variance representing the deviation in utilization among different resource types.
S/Sh_i	Unified scheduling score computed for a candidate host/host h_i .
E	Estimated Energy Proxy (represented by the host's estimated CPU utilization).
α	Adaptive weighting coefficient assigned to the CPU utilization component.
β	Adaptive weighting coefficient assigned to the resource imbalance variance (σ^2).
γ	Adaptive weighting coefficient assigned to the Energy Proxy component (E).
η	An extra constant or bias term included in the scheduling score function.

Table 2. Cont.

Symbol	Description
C	Candidate Set containing the filtered top-k best performing hosts.
k	The number of top-performing hosts selected to form the Candidate Set.
w _i	Inverse weight calculated for host h _i based on its absolute scheduling score.
δ	Stabilization constant added to the denominator to prevent division by zero.
P(h _i)	Normalized selection probability assigned to host h _i within the candidate set.
ε	Exploration parameter used in the epsilon-greedy strategy to control randomness.
Hfeasible	The complete pool of all feasible candidate hosts capable of executing the task.
hselected	The final host chosen by the scheduler for task allocation.
PE	Predicted Execution Estimation

The first component in the scheduling layer is the Estimated CPU Utilization, which represents the expected processor load on a given host after assigning a cloudlet. As illustrated in Equation (1), this value is computed by dividing the total expected CPU usage, current utilization plus cloudlet demand, by the host's CPU capacity. A lower value of the estimated CPU utilization indicates a less loaded host, enabling the scheduler to select resources that can handle tasks more efficiently and minimize potential processing delays.

The second component is the Load Balancing metric, which evaluates how evenly the host's resources, CPU, memory, and I/O, will be utilized after task assignment. After computing the utilization of each resource using Equations (1)–(3), the mean utilization is calculated as defined in Equation (4). However, the mean alone does not fully capture resource balance. Therefore, the imbalance variance is computed using Equation (5). A lower variance value indicates a more balanced distribution of resource utilization, which helps reduce bottlenecks and enhances overall system stability and reliability.

To measure resources' usage, let CPU_u, MEM_u, and IO_u represent the utilization of CPU, memory, and I/O resources respectively after assigning a task. The internal computation of the SPES algorithm begins by calculating the Estimated CPU Utilization (CPU_u) as the primary performance indicator as shown in Equation (1).

Equations (1)–(3) compute utilization values for the three core resources based on host usage and cloudlet requirements.

$$\text{CPU}_u = [\text{CPU}_{\text{current}}(h) + \text{CPU}_{\text{task}}] / \text{CPU}_{\text{capacity}}(h) \quad (1)$$

$$\text{MEM}_u = [\text{MEM}_{\text{current}}(h) + \text{MEM}_{\text{task}}] / \text{MEM}_{\text{capacity}}(h) \quad (2)$$

$$\text{IO}_u = [\text{IO}_{\text{current}}(h) + \text{IO}_{\text{task}}] / \text{IO}_{\text{capacity}}(h) \quad (3)$$

The third component is the Estimated Energy Proxy, which provides an indicator of power consumption based on CPU utilization, that serves as a practical approximation of runtime power demand, enabling the system to balance execution performance with energy efficiency.

The internal computation of the SPES algorithm then proceeds by calculating the mean utilization (μ) across these three core resources, as shown in Equation (4), which reflects the average resource usage after virtual task placement.

$$\mu = [\text{CPU}_u + \text{MEM}_u + \text{IO}_u]/3 \quad (4)$$

This measure provides a general indicator of how intensively a host's resources will be utilized. However, mean utilization alone does not reveal imbalance among various resources' aspects. To address this challenge, the imbalance variance (σ^2), as shown in Equation (5), is computed to measure unequal distribution among CPU, memory, and I/O. A smaller value of variance indicates that resources are being used more balanced, that leads to improved performance stability and resources management. The imbalance variance is computed independently for each candidate host during task evaluation, allowing the scheduler to compare the expected resource balance across all feasible hosts before final task allocation.

$$\sigma^2 = [(\text{CPU}_u - \mu)^2 + (\text{MEM}_u - \mu)^2 + (\text{IO}_u - \mu)^2]/3 \quad (5)$$

3.3. Adaptive Multi-Objective Function

Finally, the three components, Estimated CPU Utilization, Load Balancing, and Estimated Energy Proxy, are combined into a unified scheduling score using adaptive weights, as shown in Equation (6). The weighting coefficients (α , β , γ) are adaptively adjusted using utilization-aware threshold rules based on the average resource utilization of each evaluated host. The scheduler first computes the average utilization of CPU, memory, and I/O resources before task allocation. When the average utilization exceeds a predefined threshold of 0.8, higher priority is assigned to the utilization component by increasing α to 0.6, while the imbalance and energy-related coefficients are reduced ($\beta = 0.25$, $\gamma = 0.15$) to minimize processing pressure and avoid host overloading under heavy workloads. Otherwise, under moderate utilization conditions, more balanced coefficients are applied ($\alpha = 0.5$, $\beta = 0.3$, $\gamma = 0.2$) to emphasize balanced resource distribution and energy efficiency. The coefficient values (α, β, γ) and the utilization threshold were experimentally tuned through iterative simulation trials to achieve a balanced trade-off among scheduling efficiency, workload distribution, and energy-aware behavior. Several parameter combinations were evaluated during the preliminary testing phase, and the selected values demonstrated stable and consistent performance across different workload scenarios. This adaptive weighting mechanism enables the SPES scheduler to dynamically respond to varying workload conditions while maintaining efficient and stable task allocation across heterogeneous cloud environments.

The host with the best final score is selected, ensuring that the SPES scheduler achieves fast, balanced, and energy-aware task allocation across heterogeneous cloud environments. Besides performance and balance, energy efficiency is a key aspect considered. To maintain a lightweight evaluation process, CPU utilization is employed as an energy proxy (E), representing relative power consumption across hosts without relying on complex energy models. The overall scheduling score (S), as illustrated in Equation (6), combines estimated CPU utilization, imbalance variance, and energy consumption into a single metric. Dynamic weighting factors, α , β , and γ , are adaptively tuned based on system utilization to ensure proper prioritization with various load scenarios. The adaptive threshold and weighting coefficients used in SPES were empirically determined through multiple preliminary simulation experiments conducted under different workload conditions. Several parameter combinations were evaluated to achieve a balanced trade-off between scheduling performance, resource balancing, and energy efficiency. The utilization threshold value of 0.8 was selected because resource contention and performance degradation become

more noticeable when host utilization exceeds approximately 80% of its available capacity. Therefore, under high utilization conditions, the scheduling process assigns higher priority to utilization-aware allocation to reduce potential overload and improve execution stability.

$$S = (\alpha \times \text{CPUu} + \beta \times \sigma^2 + \gamma \times E) + \eta \quad (6)$$

Equations (1)–(6) are based on standard resource utilization analysis and multi-objective optimization concepts commonly adopted in cloud scheduling frameworks [21,23,50]. The proposed SPES model integrates CPU, memory, and I/O utilization evaluation with statistical imbalance analysis and energy-aware scheduling into a unified lightweight stochastic scheduling algorithm. In SPES, the weighting coefficients are dynamically adjusted according to the system's operational state, giving higher priority to lower resource utilization during peak load conditions while emphasizing balance and energy efficiency under moderate utilization levels.

3.4. Stochastic Candidate Selection and Epsilon-Greedy Exploration

Equation (7) defines the selection of the Candidate Set (C). Instead of selecting a single optimal host, the SPES scheduler identifies the top k hosts that exhibit the lowest scores, representing best performance.

$$C = \{h_1, h_2, \dots, h_k\} \mid S_{h_1} \leq S_{h_2} \leq \dots \leq S_{h_n} \quad (7)$$

This filtering process ensures that the subsequent stochastic selection is confined to high-performing resources, maintaining a balance between rigorous optimization and the flexibility needed for dynamic load distribution.

Before constructing the Top K candidate set, SPES first evaluates all feasible hosts according to resource availability constraints. A host is considered feasible only if its available CPU, memory, and I/O capacities satisfy the corresponding resource requirements of the incoming cloudlet. Hosts violating any of these constraints are excluded from further evaluation, ensuring that the subsequent optimization process operates only on feasible candidate hosts while preventing infeasible task assignments.

$$W_i = 1/(|S_{h_i}| + \delta), P(h_i) = W_i / [\sum_{j=1}^k W_j] \quad (8)$$

Equation (8) is used to implement the stochastic nature of the algorithm, each host within the candidate set C is assigned a selection probability $P(h_i)$. This is achieved by first calculating an inverse weight w_i based on the host's score, where δ is a stabilization constant to prevent division by zero. The probabilities are then normalized across the top k candidates. This mechanism ensures that hosts with better performance metrics have a higher likelihood of selection, while still allowing other qualified candidates to be utilized, thus preventing resource over saturation.

$$h_{\text{selected}} = \begin{cases} \text{Sample from } P(h_i) \text{ with probability } 1 - \epsilon \\ \text{Random } h \in H_{\text{feasible}} \text{ with probability } \epsilon \end{cases} \quad (9)$$

Equation (9) governs the final host selection process using an epsilon-greedy exploration strategy. With a probability of $(1 - \epsilon)$, the scheduler exploits the predictive scheduling mechanism by selecting a host from the Top K candidate hosts according to the weighted probability distribution $P(h_i)$, where hosts with lower scheduling scores are assigned higher selection probabilities. Conversely, with a small exploration probability ϵ , the scheduler randomly selects a host from the feasible candidate host pool regardless of its ranking score. In the proposed SPES implementation, the exploration parameter is

experimentally set to $\varepsilon = 0.09$, meaning that 91% of scheduling decisions rely on predictive exploitation, while 9% are dedicated to stochastic exploration. This exploration and exploitation balance helps the scheduler avoid repetitive deterministic selections, reduces the risk of local optima convergence, and improves workload distribution stability across heterogeneous multi-region cloud environments.

Once the scheduling decision is completed, the Execution Layer carries out the processing of assigned tasks while continuously monitoring host utilization and task progress. Upon cloudlet completion, updated utilization data is sent back to the Scheduling Layer, establishing a closed feedback loop that enhances real time adaptivity and maintains global system awareness. This loop ensures that subsequent decisions reflect the most recent infrastructure state, contributing to reduced energy waste by preventing persistent underutilization and enabling dynamic workload redistribution.

Through the seamless integration of its four interactive layers, the proposed SPES algorithm establishes a scheduling ecosystem that supports predictive modeling, adaptive decision making, and efficient execution. This workflow significantly improves makespan, resource utilization, and energy efficiency, demonstrating high scalability and stability in heterogeneous multi-region cloud environments.

3.5. Predictive Execution Estimation

Unlike machine learning-based prediction models that require historical training data, the predictive component of the proposed SPES algorithm performs lightweight execution estimation using the current resource status of candidate virtual machines. Before assigning a cloudlet, SPES estimates its expected execution behavior by jointly considering the current CPU utilization, memory utilization, I/O utilization, and the processing capability of each candidate VM.

Specifically, the scheduler predicts the relative execution suitability of each candidate resource using the latest utilization measurements available during scheduling. These measurements provide an estimation of the expected workload that the VM is likely to experience immediately after task allocation. Consequently, the scheduling decision is performed proactively rather than reactively, allowing the algorithm to avoid assigning tasks to resources expected to become overloaded.

Since the prediction relies only on instantaneous system information instead of historical datasets or computationally expensive learning models, the proposed approach preserves the lightweight nature of SPES while maintaining fast scheduling decisions suitable for dynamic cloud environments. The overall Predicted Execution (PE_i) score for a candidate host i is formulated as follows:

$$PE_i = \alpha \times CPU_u + \beta \times MEM_u + \gamma \times IO_u \quad ; \quad \alpha + \beta + \gamma = 1 \quad (10)$$

A lower predicted execution score ($\$PE_i\$$) indicates a more suitable and efficient resource for task allocation.

3.6. Energy-Aware Scheduling

The proposed SPES algorithm promotes energy-efficient scheduling by incorporating CPU utilization as an indirect indicator of energy consumption during task allocation. Although actual power consumption in cloud data centers is influenced by multiple hardware components, including processors, memory, storage, and networking devices, previous cloud computing studies have demonstrated that CPU utilization is strongly correlated with server power consumption and therefore represents a practical and widely adopted energy proxy [51,52].

Accordingly, SPES does not attempt to estimate the exact electrical power consumed by each host. Instead, it minimizes excessive CPU utilization by distributing workloads more evenly across available virtual machines, thereby reducing resource hotspots and indirectly improving overall energy efficiency. This lightweight strategy avoids the computational overhead associated with detailed power models while remaining consistent with the abstraction level provided by the CloudSim simulation framework.

Nevertheless, the proposed approach relies on CPU utilization as an approximation rather than a complete energy model.

3.7. Operational Workflow

The scheduling process of SPES as shown in Figure 2, which incorporates a stochastic decision-making strategy to improve adaptability and avoid local optima. The scheduling workflow evaluates all candidate hosts, maintains a subset of top-performing candidates (Top K), and selects the final host using probabilistic weighting based on computed scores. The Top K candidate mechanism was introduced to reduce the search space while preserving solution diversity. Instead of evaluating all available hosts during the stochastic selection stage, only the highest-ranked candidate hosts are considered. The selected value of K was empirically determined through simulation experiments to provide a balance between scheduling performance and computational efficiency. Smaller K values reduced exploration capability, whereas larger values increased computational overhead without noticeable improvement in performance. Additionally, an epsilon exploration mechanism introduces controlled randomness, allowing the scheduler to occasionally explore alternative host assignments and enhance global performance. The ϵ parameter controls the exploration–exploitation trade-off during stochastic host selection. A small exploration probability enables SPES to occasionally examine alternative candidate hosts, reducing the likelihood of repeatedly selecting the same resource under dynamic workload conditions. The adopted ϵ value was experimentally tuned through preliminary simulations to maintain scheduling stability while preserving sufficient exploration capability.

The overall scheduling process begins by collecting real-time utilization metrics from all available hosts. For each incoming cloudlet, SPES filters feasible hosts based on resource constraints, computes predictive and utilization metrics, evaluates the scheduling score, and selects the target host using the stochastic selection mechanism. After assignment, host utilization values are updated, and the process repeats iteratively until all tasks are scheduled.

The detailed operational workflow of the proposed SPES algorithm is illustrated in Figure 3, which summarizes the main scheduling stages, including feasible host filtering, predictive resource evaluation, adaptive scoring, and stochastic task allocation. The detailed mathematical computations associated with utilization estimation, imbalance variance, adaptive weighting, and stochastic selection are presented through Equations (1)–(10). Through the integration of predictive estimation, multi-resource balancing, and stochastic exploration, the SPES algorithm achieves an effective trade-off between performance, scalability, and energy efficiency. Compared to population-based metaheuristic approaches such as ISSA, SPES operates as a lightweight scheduling strategy with lower computational overhead, enabling efficient and scalable task scheduling in heterogeneous and multi-region cloud environments.

3.8. Computational Complexity Analysis

The computational efficiency of the proposed SPES scheduler mainly results from its lightweight deterministic evaluation strategy. Unlike population-based metaheuristic algorithms, SPES performs a direct evaluation of candidate hosts for each arriving task without

iterative optimization, population evolution, or repeated fitness updates. Consequently, the scheduling overhead remains relatively low even as the workload size increases.

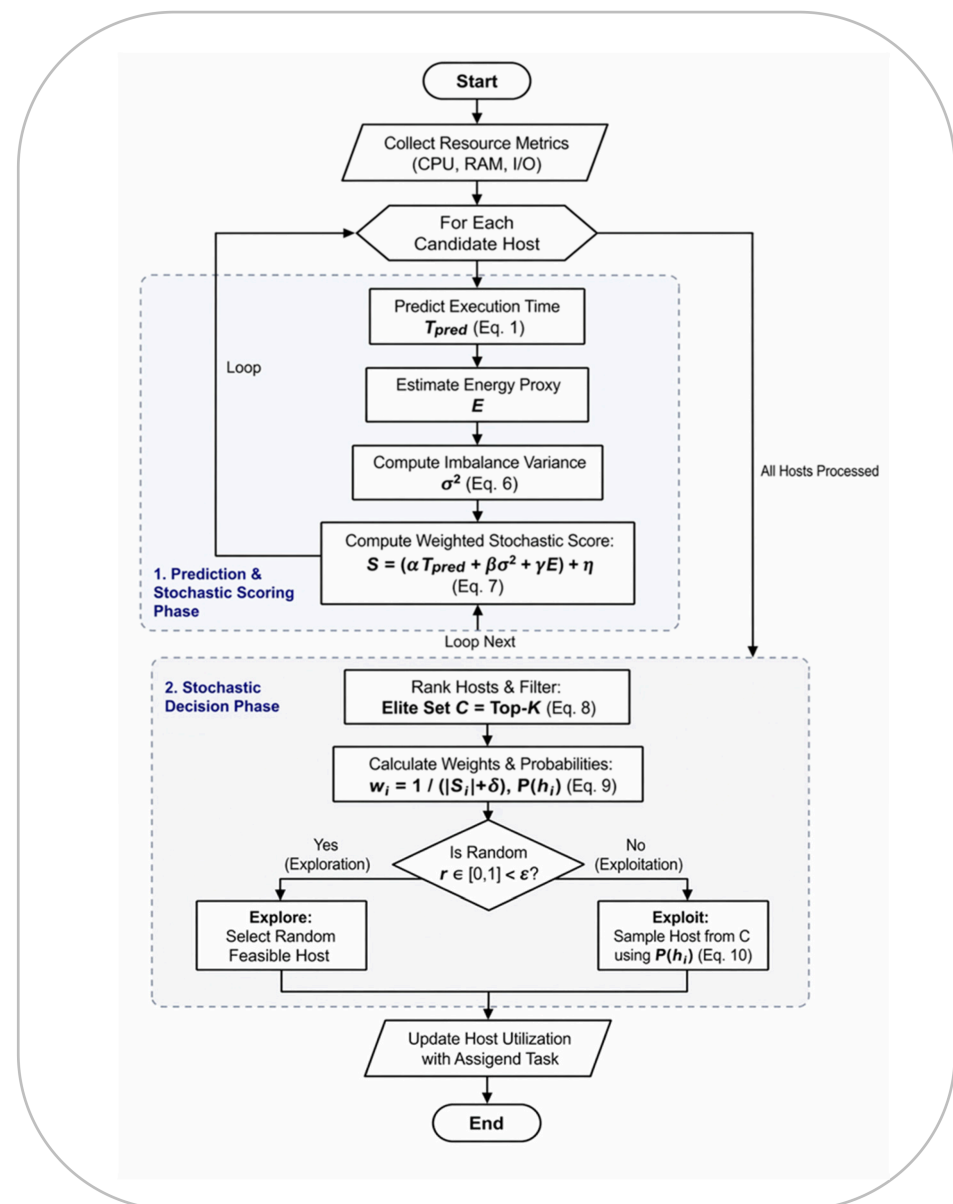


Figure 2. Flowchart of the Cloudlet Execution in the Proposed SPES Algorithm.

Assuming that N tasks are scheduled over H available hosts, SPES evaluates each feasible host once for every task. For each host, only a fixed number of arithmetic operations are required, including execution time estimation, resource utilization calculation, imbalance variance computation, score normalization, adaptive weighting, and stochastic perturbation. Since these operations execute in constant time, SPES has a time complexity of $O(N \times H)$. In contrast, ISSA is a population-based metaheuristic algorithm whose computational complexity depends on the population size (P), the number of optimization iterations (I), and the number of evaluated hosts. Therefore, ISSA has a time complexity of $O(N \times P \times I \times H)$ because multiple candidate solutions are repeatedly updated and evaluated throughout the optimization process.

```

1  Input: Set of hosts H, Set of tasks T
2  Output: Task-to-Host mapping M

3  Initialize  $M \leftarrow \emptyset$ 

4  for each task  $t_i \in T$  do

5      candidateHosts  $\leftarrow$  Filter feasible hosts

6      if candidateHosts =  $\emptyset$  then
7          continue

8      for each host  $h_j \in$  candidateHosts do

9          Compute estimated CPU utilization
10         Compute resource imbalance
11         Compute energy proxy
12         Compute adaptive scheduling score

13     Select Top-K candidate hosts

14     Apply weighted stochastic selection

15     Apply epsilon-greedy exploration

16     Assign task to selected host

17     Update host utilization

18 return M

```

Figure 3. Workflow of the Proposed SPES Algorithm.

Therefore, although both algorithms evaluate candidate hosts, SPES eliminates the iterative optimization process required by ISSA, resulting in substantially lower computational overhead while maintaining competitive scheduling quality. This theoretical analysis is consistent with the experimental results presented in Section 4, where SPES achieves lower execution times and maintains efficient scalability as the workload size increases.

4. Results and Discussion

4.1. Experimental Setup

The experiments were conducted using the CloudSim 5.0 simulation toolkit to evaluate the performance of the proposed Stochastic Predictive Energy-Aware Scheduling (SPES) algorithm. To simulate a multi-region cloud environment, two data centers were deployed, representing Region A and Region B. Each data center consists of four heterogeneous custom hosts with varying CPU, memory, and I/O capacities, reflecting realistic cloud infrastructure conditions.

The simulated environment includes a total of 30 virtual machines (VMs), where each VM is configured with 1 vCPU, 512 MB RAM, 1000 MIPS, and 1 Gbps bandwidth. This configuration ensures a balanced yet constrained environment suitable for evaluating scheduling performance under heterogeneous resource conditions. To assess the scalability and efficiency of the proposed approach, experiments were conducted using different workload sizes consisting of 30, 70, 100, and 200 independent cloudlets. These workloads are designed to reflect increasing system demand and evaluate the behavior of the scheduling algorithm under varying load conditions.

Table 3 summarizes the hardware configuration of the heterogeneous hosts used in the CloudSim simulation. The simulated infrastructure consists of two data centers

with identical host configurations, providing a total of eight heterogeneous hosts. This configuration enables the evaluation of SPES under diverse resource capacities while ensuring the reproducibility of the experimental environment.

Table 3. Heterogeneous configuration of the simulated hosts.

Host	Processing Elements (PEs)	CPU Capacity (MIPS)	RAM (MB)	I/O Capacity (Bandwidth)
Host 1	6	400,000	6000	1,000,000
Host 2	4	200,000	6000	1,000,000
Host 3	2	100,000	9000	800,000
Host 4	3	300,000	6000	1,500,000
Host 5	6	400,000	6000	1,000,000
Host 6	4	200,000	6000	1,000,000
Host 7	2	100,000	9000	800,000
Host 8	3	300,000	6000	1,500,000

The simulated environment includes 30 virtual machines distributed across the eight heterogeneous hosts using an initial round-robin placement strategy to provide a balanced execution environment. During scheduling, SPES first selects the most appropriate host according to the proposed predictive and resource-aware decision mechanism, after which the cloudlet is assigned to one of the virtual machines associated with the selected host. This design enables dynamic task allocation while maintaining a consistent VM configuration throughout the simulation.

The performance evaluation is based on three key metrics, all measured in seconds. The first metric is the makespan, defined as the maximum completion time among all cloudlets, which reflects the overall system responsiveness. The second metric is the total execution time, representing the cumulative CPU processing time across all cloudlets, indicating the computational efficiency of the system. The third metric is the average execution time, defined as the mean execution time per cloudlet, which provides insight into scheduling fairness and workload distribution.

The performance of the proposed SPES algorithm is compared against the Improved Sparrow Search Algorithm (ISSA) [23], which serves as a compared scheduling approach. ISSA was selected as the primary baseline because it represents a recent and intelligent metaheuristic scheduling approach specifically designed for cloud resource allocation and load balancing. In contrast to traditional scheduling algorithms such as FCFS and Round Robin, ISSA incorporates adaptive optimization capabilities suitable for heterogeneous and dynamic cloud environments. Therefore, comparing SPES against ISSA provides a stronger and more meaningful evaluation of scheduling efficiency and scalability under dynamic workload conditions. This comparison evaluates the effectiveness of SPES in improving scheduling efficiency, reducing execution time, and maintaining balanced resource utilization under different workload conditions. Overall, these metrics provide a comprehensive assessment of the efficiency, scalability, and adaptability of the proposed algorithm in heterogeneous multi-region cloud environments.

To improve the reliability of the reported results, each simulation scenario was executed multiple times using different randomly generated workload characteristics while maintaining the same experimental configuration. The reported performance metrics correspond to the average values obtained from these independent simulation runs. Since the cloudlets were generated with different CPU, memory, and I/O utilization profiles in each run, slight variations in the scheduling metrics naturally occurred.

4.2. Experiments Results

As presented in Table 4, the experimental evaluation was conducted across four workload scenarios consisting of 30, 70, 100, and 200 cloudlets to assess the performance of the proposed Stochastic Predictive Energy-Aware Scheduling (SPES) algorithm in comparison with the Improved Sparrow Search Algorithm (ISSA). All experiments were performed under identical simulation settings to ensure a fair comparison. Lower metric values indicate better performance.

Table 4. Performance comparison between the proposed SPES algorithm and the compared ISSA under different workload sizes. Metrics include makespan, total execution time, and average execution time in seconds.

Cloudlets	Algorithm	Makespan	Total Execution Time	Average Execution Time
30	SPES	2.2	32.0	1.067
30	ISSA	2.2	36.0	1.200
70	SPES	3.199	171.967	2.457
70	ISSA	4.199	177.963	2.542
100	SPES	4.199	343.854	3.439
100	ISSA	5.199	345.904	3.459
200	SPES	7.197	1341.367	6.707
200	ISSA	8.197	1343.361	6.717
500	SPES	17.195	8338.121	16.676
500	ISSA	18.192	8345.835	16.692

For the workload of 30 cloudlets, both SPES and ISSA achieved identical makespan values of 2.2 s. However, SPES demonstrated improved efficiency in terms of total execution time (32.0 s vs. 36.0 s) and average execution time (1.067 s vs. 1.200 s), indicating better resource utilization even under light workload conditions.

As the workload increased to 70 cloudlets, SPES showed clear performance improvements over ISSA. Specifically, the makespan was reduced from 4.199 s to 3.199 s, representing an improvement of approximately 23.8%. Additionally, total execution time decreased from 177.963 s to 171.967 s (3.4%), while average execution time improved slightly from 2.542 s to 2.457 s. These results highlight the ability of SPES to maintain balanced resource allocation under moderate workload conditions.

For 100 cloudlets, SPES continued to outperform ISSA by reducing the makespan from 5.199 s to 4.199 s, achieving an improvement of approximately 19.2%. While the reduction in total execution time was relatively modest (345.904 s to 343.854 s), the results indicate consistent performance gains and improved scheduling efficiency.

Under the largest workload of 200 cloudlets, SPES maintained its performance advantage by reducing the makespan from 8.197 s to 7.197 s (12.2%). The total execution time also showed slight improvement (1343.361 s to 1341.367 s), along with marginal enhancement in average execution time. These results demonstrate that SPES scales effectively and maintains stable performance even under high-workload conditions.

To further verify the consistency of the proposed scheduler, an additional scalability experiment was conducted using 500 cloudlets while maintaining the same simulation configuration. The obtained results continued to demonstrate the superiority of SPES over ISSA, while exhibiting naturally varying performance metrics under the larger workload.

Overall, the experimental results confirm that SPES consistently outperforms ISSA in terms of makespan reduction and execution efficiency. The improvements become more evident as workload size increases, highlighting the effectiveness of the proposed stochastic,

multi-resource-aware scheduling strategy in handling dynamic and heterogeneous cloud environments.

The comparative analysis between the proposed Stochastic Predictive Energy-Aware Scheduling (SPES) algorithm and the Improved Sparrow Search Algorithm (ISSA) is illustrated through Figures 4–6, which present makespan, average execution time, and total execution time, respectively.

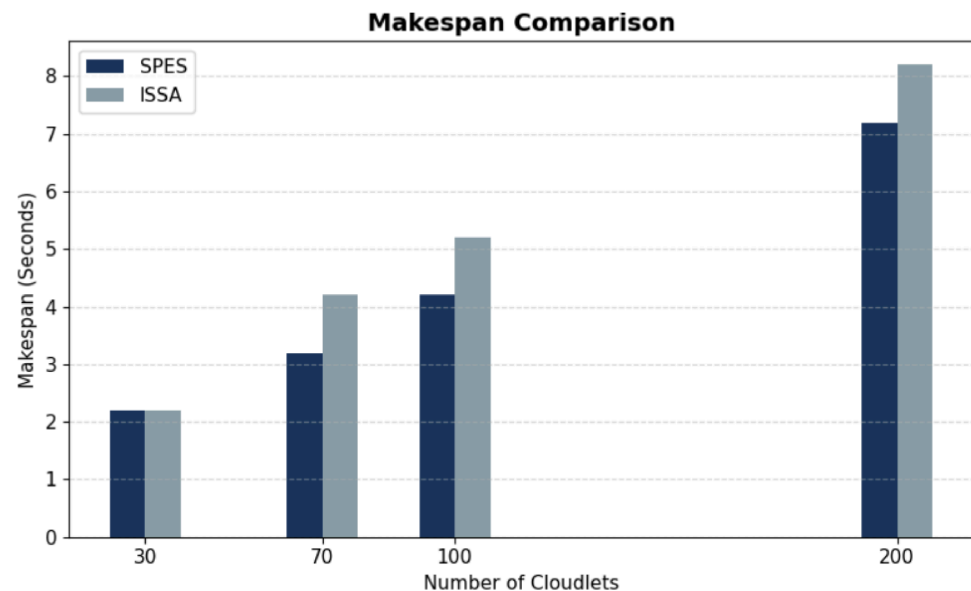


Figure 4. A Comparison of Makespan for ISSA and SPES Algorithms with Varying Numbers of Cloudlets.

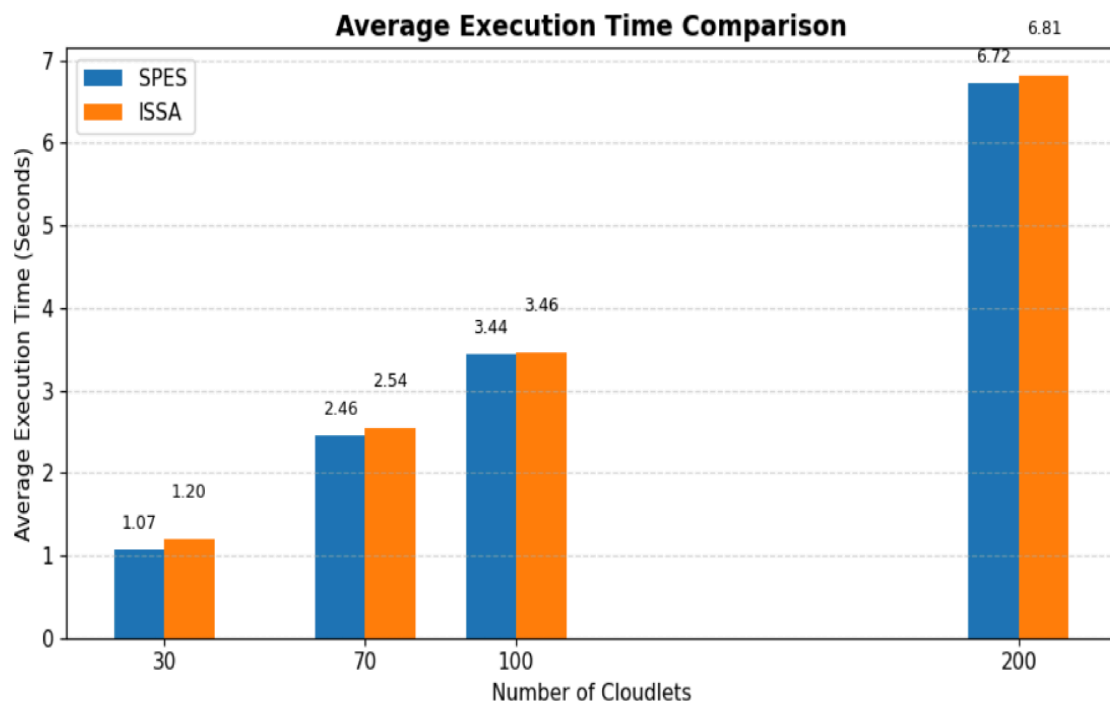


Figure 5. Average Execution Time of ISSA and SPES Algorithms with Varying Numbers of Cloudlets.

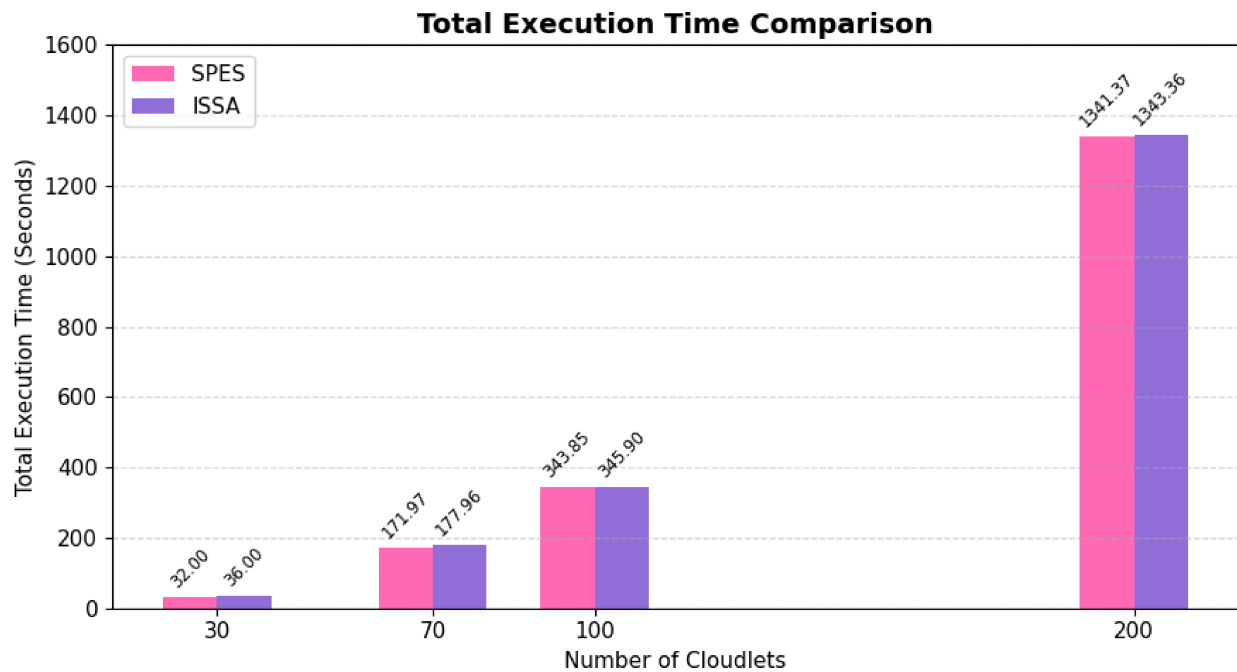


Figure 6. Total Execution Time of ISSA and SPES Algorithms with Varying Numbers of Cloudlets.

As shown in Figure 4, SPES consistently achieves lower makespan values compared to ISSA as the workload increases. For instance, SPES reduces the makespan by 23.8% for 70 cloudlets, 19.2% for 100 cloudlets, and 12.2% for 200 cloudlets. This improvement highlights the scalability of SPES and its ability to maintain efficient scheduling under increasing system load. The smoother makespan trend observed for SPES is primarily attributed to its stochastic Top K selection strategy combined with predictive execution time estimation and variance-based load balancing, which collectively reduce resource contention and prevent host overloading.

Figure 5 presents the comparison of average execution time per cloudlet. The results indicate that SPES achieves slightly lower average execution times compared to ISSA across all workload scenarios. For example, the average execution time is reduced from 2.542 s to 2.457 s for 70 cloudlets and from 3.459 s to 3.439 s for 100 cloudlets. Although the improvements are relatively modest, they demonstrate that SPES provides more balanced task distribution and reduces execution variability across virtual machines. This behavior is mainly driven by the multi-objective scoring model, which jointly considers execution time, resource imbalance, and energy-aware factors.

Figure 6 illustrates the total execution time comparison, providing a system-level perspective on scheduling efficiency. SPES achieves slightly lower total execution time values than ISSA, such as reducing the total execution time from 177.963 s to 171.967 s for 70 cloudlets and from 1343.361 s to 1341.367 s for 200 cloudlets. These results indicate that SPES improves overall computational efficiency while maintaining stable performance as workload increases. The improvement can be attributed to better workload distribution and reduced resource imbalance rather than drastic changes in execution behavior.

In addition, SPES incorporates energy awareness through the use of CPU utilization as a lightweight proxy for power consumption. While this does not represent a detailed energy model, it enables the scheduler to indirectly reduce unnecessary resource usage and improve overall system efficiency, aligning with the principles of green computing.

Overall, the results presented in Figures 4–6 demonstrate that SPES outperforms ISSA primarily in terms of makespan reduction, with consistent but moderate improvements in execution time metrics. The findings confirm that SPES provides a scalable, reliable,

and efficient scheduling approach capable of adapting to dynamic and heterogeneous multi-region cloud environments.

4.3. Discussion

The experimental results indicate that the proposed Stochastic Predictive Energy-Aware Scheduling (SPES) algorithm provides noticeable improvements over the Improved Sparrow Search Algorithm (ISSA), particularly in terms of makespan reduction. The observed improvements become more evident as the workload size increases, where SPES achieves reductions of up to approximately 23.8% and 19.2% for moderate and high workloads, respectively. This behavior confirms the scalability of the proposed approach under dynamic and heterogeneous cloud conditions. The performance gains of SPES can be attributed to its stochastic multi-objective scheduling strategy, which integrates predictive execution time estimation, variance-based load balancing, and adaptive scoring. Unlike deterministic or purely metaheuristic approaches, SPES combines Top K candidate selection with stochastic exploration, enabling more flexible and balanced task allocation decisions. This reduces the likelihood of overloading specific hosts and improves overall system stability. In addition, the use of a variance-based imbalance metric allows SPES to maintain more proportional utilization of CPU, memory, and I/O resources. This contributes to reducing resource bottlenecks and enhances workload distribution across heterogeneous hosts, which is reflected in the slight improvements observed in average and total execution time metrics.

Furthermore, SPES incorporates energy awareness through a lightweight CPU utilization proxy. Although this approach does not rely on detailed energy modeling, it enables the scheduler to indirectly avoid inefficient resource usage and reduce unnecessary load concentration, contributing to improved operational efficiency in line with green computing principles. Overall, the results demonstrate that SPES offers a balanced trade-off between performance improvement, computational simplicity, and adaptability. The algorithm is particularly effective in reducing makespan while maintaining stable execution behavior, making it a suitable candidate for scalable and dynamic multi-region cloud environments.

The effectiveness of the proposed SPES algorithm becomes more noticeable in heterogeneous and dynamic multi-region cloud environments where workload distribution and resource contention continuously change over time. The predictive and stochastic scheduling behavior of SPES enables adaptive task allocation under medium- and high-workload conditions, particularly when balancing CPU, memory, and I/O utilization is critical for maintaining stable system performance. In contrast, under light or nearly static workloads with minimal resource contention, the performance difference between SPES and simpler scheduling strategies may become less significant due to the reduced scheduling complexity in such environments. Furthermore, although SPES provides lightweight adaptive scheduling with lower computational overhead than population-based metaheuristic approaches, its predictive mechanism is primarily utilization-based and does not rely on advanced workload forecasting or machine learning models. Therefore, highly bursty or extremely unpredictable workload patterns may require more sophisticated predictive mechanisms in future extensions of the proposed framework.

Although ISSA was selected as the only baseline algorithm in this study, this choice was intentional because ISSA represents the most closely related state-of-the-art scheduling approach and serves as the primary methodological reference for the proposed SPES scheduler. Moreover, the original ISSA study extensively compared its performance with several established scheduling algorithms, including QMPSO, ACOFTF, and ADA, demonstrating its superiority across multiple performance metrics. Therefore, achieving consistent im-

provements over ISSA provides meaningful evidence of the effectiveness of SPES while maintaining a focused comparison between two closely related scheduling strategies.

5. Conclusions and Future Work

This paper presented the Stochastic Predictive Energy-Aware Scheduling (SPES) algorithm, a lightweight and adaptive scheduling approach designed for efficient task allocation in heterogeneous multi-region cloud environments. The proposed method combines predictive execution time estimation, multidimensional resource imbalance evaluation, and energy awareness within a unified multi-objective scoring framework, enabling adaptive and efficient scheduling decisions under dynamic workloads. The experimental results demonstrate that SPES achieves significant improvements in makespan compared to the Improved Sparrow Search Algorithm (ISSA), particularly under moderate- and high-workload conditions. While improvements in total and average execution time are relatively modest, the results indicate more balanced workload distribution and stable execution behavior across heterogeneous resources. These findings confirm that SPES provides effective scalability and improved scheduling efficiency in dynamic cloud environments.

In addition, SPES incorporates energy awareness through a lightweight CPU utilization proxy, enabling more efficient resource usage without introducing complex energy modeling overhead. This design supports practical applicability while aligning with green computing principles. Overall, SPES offers a balanced trade-off between performance improvement, computational simplicity, and adaptability, making it a suitable solution for large-scale and multi-region cloud scheduling scenarios.

Future work will focus on enhancing the predictive capabilities of SPES by integrating machine learning-based workload forecasting techniques. Although CPU utilization provides an effective proxy for energy-aware scheduling in CloudSim-based environments, future work will integrate detailed power consumption models that consider multiple hardware components to further enhance energy optimization. Additionally, adaptive parameter tuning strategies will be explored to further improve responsiveness under highly dynamic conditions. Further extensions will investigate the integration of fault tolerance mechanisms and large-scale evaluations across geographically distributed cloud environments to validate the robustness and real-world applicability of the proposed approach.

Author Contributions: Conceptualization, M.F.Y.; methodology, M.F.Y., A.E.A.R., W.G. and N.L.B.; software, M.F.Y.; validation, A.E.A.R., W.G. and N.L.B.; formal analysis, M.F.Y.; investigation, M.F.Y.; data curation, M.F.Y.; writing—original draft preparation, M.F.Y.; writing—review and editing, A.E.A.R., W.G. and N.L.B.; supervision, A.E.A.R., W.G. and N.L.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data analyzed in this study were dynamically generated during the simulation process using the CloudSim 5.0 framework. The simulation setups and generated datasets are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Arunagiri, R.; Kandasamy, V. A survey of scheduling algorithm in cloud computing environment. *Int. J. Control Theory Appl.* **2016**, *9*, 137–145.

2. Singh, S.; Jeong, Y.S.; Park, J.H. A survey on cloud computing security: Issues, threats, and solutions. *J. Netw. Comput. Appl.* **2016**, *75*, 200–222. [[CrossRef](#)]
3. Naik, A.; Sooda, K. Efficient Job Scheduling in Cloud Environments using Reinforcement Learning Actor-Critic Models. *Eng. Technol. Appl. Sci. Res.* **2024**, *14*, 16559–16564. [[CrossRef](#)]
4. Zhan, Z.-H.; Liu, X.F.; Gong, Y.J.; Zhang, J.; Chung, H.S.H.; Li, Y. Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches. *ACM Comput. Surv.* **2015**, *47*, 63. [[CrossRef](#)]
5. Ahmady, E. Cloud Computing Principles. *Int. J. Adv. Res. Comput. Commun. Eng.* **2024**, *13*, 18–21.
6. Singh, S.; Chana, I. QRSF: QoS-aware resource scheduling framework in cloud computing. *J. Supercomput.* **2015**, *71*, 2419–2453.
7. Buyya, R.; Yeo, C.S.; Venugopal, S.; Broberg, J.; Brandic, I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.* **2009**, *25*, 599–616. [[CrossRef](#)]
8. Abdel-Basset, M.; Mohamed, R.; Abd Elkhaliq, W.; Sharawi, M.; Sallam, K.M. Task scheduling approach in cloud computing environment using hybrid differential evolution. *Mathematics* **2022**, *10*, 4049. [[CrossRef](#)]
9. Amini Motlagh, A.; Movaghar, A.; Rahmani, A.M. Task scheduling mechanisms in cloud computing: A systematic review. *Int. J. Commun. Syst.* **2020**, *33*, e4302.
10. Sanjalawe, Y.; Al-E'mari, S.; Fraihat, S.; Makhadmeh, S. AI-driven job scheduling in cloud computing: A comprehensive review. *Artif. Intell. Rev.* **2025**, *58*, 197. [[CrossRef](#)]
11. Mao, M.; Humphrey, M. Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In Proceedings of the SC Conference, Seattle, WA, USA, 12–18 November 2011; pp. 1–11.
12. Kumar, M.; Sharma, S.C.; Goel, A.; Singh, S.P. A comprehensive survey for scheduling techniques in cloud computing. *J. Netw. Comput. Appl.* **2019**, *143*, 1–33. [[CrossRef](#)]
13. AlMansour, N.; Allah, N.M. A Survey of Scheduling Algorithms in Cloud Computing. In Proceedings of the 2019 International Conference on Computer and Information Sciences (ICCIS), Sakaka, Saudi Arabia, 3–4 April 2019; pp. 1–6.
14. NoorianTalouki, R.; Hosseini Shirvan, M.; Motameni, H. A heuristic-based task scheduling algorithm for scientific workflows in heterogeneous cloud computing platforms. *J. King Saud. Univ.-Comput. Inf. Sci.* **2022**, *34*, 4902–4913. [[CrossRef](#)]
15. Cui, M.; Wang, Y. An Effective QoS-Aware Hybrid Optimization Approach for Workflow Scheduling in Cloud Computing. *Sensors* **2025**, *25*, 4705. [[CrossRef](#)] [[PubMed](#)]
16. Sati, S.O.; Zhang, J.; Tu, S.; Waqas, M. Performance Evaluation of Scheduling Algorithms Using CloudSim. Master's Thesis, School of Engineering, University of Greenwich, London, UK, 2024.
17. Saxena, D.; Chauhan, R.K.; Saxena, S. Evaluating Makespan of Min-Min and Max-Min Scheduling Techniques in Cloud Computing: Concepts and Comparison. *Int. J. Trend Res. Dev.* **2017**, *3*, 12–19. [[CrossRef](#)]
18. Jomah, S.; Aji, S. Meta-Heuristic Scheduling: A Review on Swarm Intelligence and Hybrid Meta-Heuristics Algorithms for Cloud Computing. *SN Oper. Res. Forum* **2024**, *5*, 94. [[CrossRef](#)]
19. Zhou, L.; Li, B.; Yu, Q.; Dai, G.; Zhou, G. Multi-Strategy Improvement of Sparrow Search Algorithm for Cloud Manufacturing Service Composition. *Wuhan Univ. J. Nat. Sci.* **2024**, *29*, 323–337. [[CrossRef](#)]
20. Wu, M.; Yang, D.; Yang, Z.; Guo, Y. Sparrow Search Algorithm for Solving Flexible Jobshop Scheduling Problem. In *Proceedings of the Advances in Swarm Intelligence, 12th International Conference, ICSI 2021, Qingdao, China, 17–21 July 2021*; Springer: Cham, Switzerland, 2021; pp. 140–154.
21. Li, Y.; Han, S.; Li, Z.; Yang, F.; Sun, Z. An Improved Sparrow Search Algorithm for Flexible Job-Shop Scheduling Problem with Setup and Transportation Time. *Int. J. Adv. Comput. Sci. Appl.* **2025**, *16*, 408. [[CrossRef](#)]
22. Qawqzeh, Y.; Alharbi, M.T.; Jaradat, A.; Sattar, K.N.A. A review of swarm intelligence algorithms deployment for scheduling and optimization in cloud computing environments. *PeerJ Comput. Sci.* **2021**, *7*, e696. [[CrossRef](#)] [[PubMed](#)]
23. Khaleel, M.I. Region-aware dynamic job scheduling and resource efficiency for load balancing based on adaptive chaotic sparrow search optimization and coalitional game in cloud computing environments. *J. Netw. Comput. Appl.* **2024**, *221*, 103788. [[CrossRef](#)]
24. Gunduz, A.; Senan, S.; Gurkas-Aydin, Z. A hybrid DECSO algorithm for efficient multi objective task scheduling in cloud computing environments. *Clust. Comput.* **2025**, *28*, 848–867. [[CrossRef](#)]
25. Kumar, D.S.; Kumar, P.; Nalini, S. Job scheduling using variable neighborhood search method. *AIP Conf. Proc.* **2024**, *3075*, 020101. [[CrossRef](#)]
26. Zarif, S.A.A.M.; Suma, L.K.; Rahman, M.S.; Islam, M.M.; Pham, P.H.; Nguyen, P.D.T. GLOPS: A Hybrid Approach for Enhanced Scheduling in Cloud Computing Environments via Machine Learning-Based Process Prediction. *IEEE Access* **2025**, *13*, 134385–134402. [[CrossRef](#)]
27. Banerjee, P.; Roy, S.; Sinha, A.; Hassan, M.M.; Burje, S.; Agrawal, A.; Bairagi, A.K.; Alshathri, S.; El-Shafai, W. MTD-DHJS: Makespan-Optimized Task Scheduling Algorithm for Cloud Computing with Dynamic Computational Time Prediction. *IEEE Access* **2023**, *11*, 118465–118479. [[CrossRef](#)]
28. Anbarkhan, S.H. Optimizing Cloud Resource Allocation with Machine Learning: Strategies for Efficient Computing. *Ingénierie Des Systèmes D'information* **2025**, *30*, 1–9. [[CrossRef](#)]

29. Tsakalidou, V.N.; Mitsou, P.; Papakostas, G.A. Machine Learning for Cloud Resources Management-An Overview. *Lect. Notes Data Eng. Commun. Technol.* **2022**, *141*, 903–915. [\[CrossRef\]](#)
30. Balaji, M.; Kumar, C.A.; Subrahmanya, G.; Rao, V.R.K. Predictive Cloud resource management framework for enterprise workloads. *J. King Saud. Univ.-Comput. Inf. Sci.* **2018**, *30*, 404–415. [\[CrossRef\]](#)
31. Rubak, A.; Taheri, J. Machine Learning for Predictive Resource Scaling of Microservices on Kubernetes Platforms. In Proceedings of the 16th IEEE/ACM International Conference on Utility and Cloud Computing, Taormina, Italy, 4–7 December 2024; pp. 1–8.
32. Guo, J.; Li, C.; Chen, Y.; Luo, Y. On-demand resource provision based on load estimation and service expenditure in edge cloud environment. *J. Netw. Comput. Appl.* **2020**, *151*, 102506. [\[CrossRef\]](#)
33. Mangalampalli, S.; Karri, G.R.; Kose, U. Multi objective trust aware task scheduling algorithm in cloud computing using whale optimization. *J. King Saud. Univ.-Comput. Inf. Sci.* **2023**, *35*, 791–809. [\[CrossRef\]](#)
34. Nayaka, R.K.; Rao, G.S. Load balancing in cloud computing with multi-objective survivors optimization. *Comput. Electr. Eng.* **2025**, *126*, 110502. [\[CrossRef\]](#)
35. Kathole, A.B.; Singh, V.K.; Goyal, A.; Kant, S.; Savyanavar, A.S.; Ubale, S.A.; Jain, P.; Islam, M.T. Novel load balancing mechanism for cloud networks using dilated and attention-based federated learning with Coati Optimization. *Sci. Rep.* **2025**, *15*, 15268. [\[CrossRef\]](#) [\[PubMed\]](#)
36. Geetha, P.; Vivekanandan, S.J.; Yogitha, R.; Jeyalakshmi, M.S. Optimal load balancing in cloud: Introduction to hybrid optimization algorithm. *Expert Syst. Appl.* **2024**, *237*, 121450. [\[CrossRef\]](#)
37. Ghafir, S.; Alam, M.A.; Siddiqui, F.; Naaz, S. Load balancing in cloud computing via intelligent PSO-based feedback controller. *Sustain. Comput. Inform. Syst.* **2024**, *41*, 100948. [\[CrossRef\]](#)
38. Amutha, S.; Kamarajapandian, P.; Nirmaladevi, J.; Saravanan, S.; Vijayalakshmi, S.; Athimoolam, S. Optimizing cloud resource allocation and load balancing through eco-efficient task scheduling. *Int. J. Intell. Syst. Appl. Eng.* **2024**, *12*, 137–143.
39. Laha, J.; Pattnaik, S.; Chaudhury, K.S. Dynamic load balancing in cloud computing: A review and a novel approach. *EAI Endorsed Trans. Internet Things* **2024**, *10*, 1–7.
40. Fan, Y. Load balance-aware dynamic cloud-edge-end collaborative offloading strategy. *PLoS ONE* **2024**, *19*, e0296897. [\[CrossRef\]](#) [\[PubMed\]](#)
41. Sree, T.N.; Khan, P.F.A.; Kalyan, R.V. Load balancing strategies for cloud computing: A comprehensive review. In *Proceedings of the 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*, Bengaluru, India, 4–6 January 2024; Springer: Singapore, 2024; pp. 1023–1032.
42. Rashid, Y.; Nakpih, C.I. ReT-ELBa: A novel algorithm for efficient load balancing in cloud computing. *Asian J. Res. Comput. Sci.* **2024**, *17*, 56–64. [\[CrossRef\]](#)
43. Khan, A.R. Dynamic load balancing in cloud computing: Optimized RL-based clustering with multi-objective optimized task scheduling. *Processes* **2024**, *12*, 519. [\[CrossRef\]](#)
44. Choudhary, A.; Rajak, R. A novel strategy for deterministic workflow scheduling with load balancing using modified min-min heuristic in cloud computing environment. *Clust. Comput.* **2024**, *27*, 6985–7006. [\[CrossRef\]](#)
45. Zhou, X.; Yang, J.; Li, Y.; Li, S.; Su, Z. Deep reinforcement learning-based resource scheduling for energy optimization and load balancing in SDN-driven edge computing. *Comput. Commun.* **2024**, *226–227*, 107925. [\[CrossRef\]](#)
46. Houssein, E.H.; Gad, A.G.; Wazery, Y.M.; Suganthan, P.N. Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends. *Swarm Evol. Comput.* **2021**, *62*, 100841. [\[CrossRef\]](#)
47. Soltanshahi, M.; Asemi, R.; Shafiei, N. Energy-aware virtual machines allocation by krill herd algorithm in cloud data centers. *Heliyon* **2019**, *5*, e02066. [\[CrossRef\]](#) [\[PubMed\]](#)
48. Lyu, Z.; Xiao, M.; Skoglund, M.; Debbah, M.; Poor, H.V. Quantization-Aware Collaborative Inference for Large Embodied AI Models. *arXiv* **2026**, arXiv:2602.13052.
49. Reddy, M.I.; Rao, P.V.; Kumar, T.S. Encryption with access policy and cloud data selection for secure and energy-efficient cloud computing. *Multimed. Tools Appl.* **2024**, *83*, 15649–15675.
50. Yu, D.; Xu, Z.; Mei, M. Multi-objective Task Scheduling Optimization Based on Improved Bat Algorithm in Cloud Computing Environment. *(IJACSA) Int. J. Adv. Comput. Sci. Appl.* **2023**, *14*, 550–556. [\[CrossRef\]](#)
51. Beloglazov, A.; Buyya, R. Energy Efficient Resource Management in Virtualized Cloud Data Centers. In Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, Melbourne, Australia, 17–20 May 2010; pp. 826–831.
52. Beloglazov, A.; Buyya, R.; Lee, Y.C.; Zomaya, A. A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems. *Adv. Comput.* **2011**, *82*, 47–111. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.