ISSN 2007-9737

Advancing Cloud Task Scheduling: Recent Developments and Comparative Insights

Jessica González-San-Martín[1,*], Laura Cruz-Reyes[1], Bernabé Dorronsoro[2], Héctor Fraire-Huacuja[1], Marcela Quiroz-Castellanos[3], Claudia Gómez-Santillán[1], Nelson Rangel-Valdez[4]

[1] Tecnológico Nacional de México,
Instituto Tecnológico de Ciudad Madero,
Division of Graduate Studies and Research,
Mexico

[2] University of Cadiz,
Computer Science Engineering,
Spain

[3] Universidad Veracruzana,
Artificial Intelligence Research Center,
Mexico

[4] Tecnológico Nacional de México,
Instituto Tecnológico de Ciudad Madero,
Research Fellow at Graduate Program Division,
Mexico

jessica.gs@cdmadero.tecnm.mx

Abstract. In the present landscape of cloud computing, the effective scheduling of tasks stands as a pivotal element in optimizing the operational efficiency of distributed systems. This paper conducts a thorough and comparative examination of recent trends and progress within this vital and ever-evolving domain. By meticulously reviewing crucial performance metrics and critically analyzing state-of-the-art methodologies, we present a comprehensive overview of Cloud Task Scheduling. We emphasize the shift towards multi-objective strategies, mirroring the escalating complexity and diversity witnessed in cloud environments. Employing innovative approaches and illustrative case studies, we delve into the practical implementation of prominent algorithms, including L_{ABC}, MaOEA-SIN, and MALO. The detailed analysis not only underscores their efficacy in real-world contexts but also pinpoints areas ripe for enhancement and adaptation within multi-cloud settings. Beyond offering an in-depth understanding of the latest developments in Cloud Task Scheduling, this article endeavors to stimulate collaboration and discourse within the academic and professional community. We aim to ignite future advancements, thereby contributing to the sustained growth of this strategic and dynamic field.

Keywords. Cloud task scheduling, cloud computing, strategies and techniques, multi-objective metaheuristics.

1 Introduction

Cloud computing is an information technology service delivery model that allows access to computing resources over the Internet. Instead of owning and maintaining servers and other infrastructure components locally, organizations and users can rent or use cloud services provided by specialized providers. In the current era of cloud computing, Cloud Task Scheduling emerges as an essential component to optimize resource management in distributed systems.

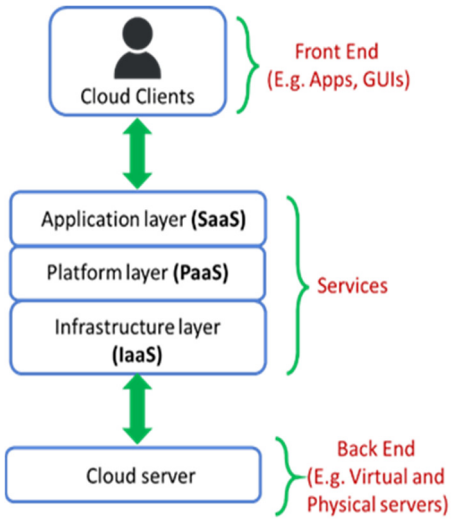

Fig. 1. Diagram by layers of cloud computing architecture [1]

Cloud computing has revolutionized the way organizations manage and access computing resources, offering flexibility and scalability. In this context, efficient task allocation becomes a crucial factor in optimizing performance and resource utilization in cloud environments.

Cloud Task Scheduling refers to the efficient planning and execution of tasks in distributed systems and cloud computing environments.

This discipline seeks to address complex challenges, such as optimal resource allocation, minimization of execution time, and maximization of the utilization of available resources. Given the increasing diversity and complexity of cloud environments, it becomes imperative to adopt advanced approaches to task scheduling.

Instead of relying on single performance metrics, the current trend is toward multi-objective strategies that consider multiple metrics simultaneously. This addresses the need to address multiple objectives and challenges inherent to cloud task scheduling, such as makespan minimization, cost optimization, and load balancing.

The adoption of multi-objective algorithms allows greater flexibility and adaptability to dynamic and heterogeneous cloud environments. This article dives into a detailed exploration of the latest trends and advancements in Cloud Task Scheduling.

Through a review of key performance metrics and critical analysis of cutting-edge methodologies, a comprehensive overview of this vital discipline is provided. The importance of multi-objective approaches is highlighted, and representative case studies will be explored.

The detailed analysis not only highlights effectiveness in real-world scenarios but also identifies areas ripe for improvements and adaptations in multi-cloud environments.

Through this work, we seek to provide an in-depth understanding of the latest developments in Cloud Task Scheduling, as well as foster collaboration within the academic and professional community and highlight future advances to contribute to the continued growth of this research field.

2 Architectural Components of Cloud Computing

The cloud computing architecture is a service model that provides on-demand access to shared computing resources over the Internet. This architecture applied to the Cloud Task Scheduling problem is generally divided into several service models and layers, each with its characteristics and functions. This allows efficient execution of distributed applications. As service models:

– Infrastructure as a Service (IaaS): Offers virtualized computing, storage, and network resources over the Internet. Users can manage and control these resources according to their needs.

- Platform as a Service (PaaS): Provides complete development and execution environments, including database services, middleware, and development tools. Users focus on application development without worrying about managing the underlying infrastructure.
- Software as a Service (SaaS): Offers complete applications through the cloud, generally accessible through a web browser. Users can use these applications without worrying about infrastructure management, updates, or maintenance.

As deployment models:

Algorithm 1 General framework of L_{ABC} algorithm

Input: System parameters

Output: Best solution

```
for i = 1 to P_size do
    Initialize the solution in a random way.
    Evaluate it and insert it into the initial population.
Record the best solution found so far.
while the stopping criterion is not satisfied do
    Employed bee phase
    for i = 1 P_size do
        Set the i^th employed bee on the i^th food source in the current population and perform the explotation task.
        Evaluate the newly generated solution and initialize the adaptive neighborhood structure.
    Onlooker bee phase
    for i = 1 P_size do
        Randomly select three solutions in the current population, select the best one as the food source for the onlooker by using the tournament selection method.
        Perform the exploitation task around the selected food source.
        Evaluate the newly generated solution and update the adaptive neighborhood structure.
        Perform the deep exploitation around the newly generated food source found by the above step.
    Scout bee phase
    If a solution in the population has not been improved during the limit trials, abandon it.
    Generate eight neighboring solutions by using the perturbation structures, and select the best neighboring solution as the scout bee to replace the current solution.
    Deep-exploitation phase
    Perform the deep-exploitation process around the best food source found so far.
    Replace the worst food source in the current population with the best one.
Output Best solution
```

- Public Cloud: Provides services over the Internet for the public. Resources are shared between multiple users and organizations.

- Private Cloud: Resources are used exclusively by one organization. It may be managed internally by the organization or by a specialized service provider. It provides greater control and customization but also involves higher costs.

- Hybrid Cloud: Combines resources from public and private clouds, allowing the portability of data and applications between them. It offers flexibility and the ability to take advantage of the benefits of both implementations.

Finally, within the context of layers, Wei [1] presents the following structure (Fig. 1) that composes the architecture of cloud computing:

– Client Layer (Client): This is the outermost layer and represents the interfaces through which users interact with cloud services. It may include graphical user interfaces, command line interfaces, mobile applications, and other means through which users access and manage cloud resources.

– Application Layer: In this layer are the applications and services that users deploy in the cloud. It may include web applications, business applications, and data analysis services, among others. These applications run on top of the infrastructure provided by the lower layers.

– Platform Layer: The platform layer provides execution environments and services that facilitate the development, deployment, and management of applications. Here is the platform as a service (PaaS), which includes managed databases, application servers, development environments, and other services that allow developers to focus on application logic without worrying about the underlying infrastructure.

– Infrastructure Layer: In this layer, infrastructure as a service (IaaS) is provided that includes computing resources, storage, and networks. Users can provision and manage virtual machines, virtual disks, virtual networks, and other resources as needed.

This layer serves as the base upon which the upper layers are built.

– Cloud Servers: At the lowest level, there are the physical and virtual servers that form the cloud infrastructure.

Algorithm 2 Framework of MaOEA-SIN

Input: The population p, the reference point z
Output: Population P

$O_{min} = \min(p_i)$ *//the minimum value for the objective function*
while $(t < t_{\max})$ **do**
  $d_i = \sin d(p_i)$
  Select two individuals randomly x_1 and x_2
  If $x_1 < x_2$
    $\text{Mating Pool}[P] = \text{Mating Pool}[P] \cup x_1$
  else if $d_{x1} < d_{x2}$
    $\text{Mating Pool}[P] = \text{Mating Pool}[P] \cup x_1$
  $q = \text{crossovermutation}(\text{Mating Pool[p]})$
  $R = [p, q]\ E = ||R_i, O_{min}||$
  for $S = 1 : N$ **do** *//population with size N*
    Select two individuals with minimum angle R_i, R_j
    If $E(R_i) > E(R_j)$
      $R_i \rightarrow []$ *//Eliminating the individual*

Algorithm 3 Ant Lion optimizer algorithm

Initialize the random solutions
Calculate the fitness function
Find the best antlions and assume it as the optimal so far
while the termination criterion is not reached **do**
  for each solution (ant) **do**
    Select an antlion using Roulette wheel
    Update the perimeters e and d
    Create a random walk normalize the chosen random walk
    If $n_i = CP$ **then**
      Update the current solution by using Eq. (15)
    else if $n = CP$ **then**
      Update the current solution by using Eq. (23)
    If $n_i = CP$ **then**
      Update the current solution by using Eq. (23)
    Calculate the fitness function of all solutions using Eq. (13)
    Replace an antlion (new solution) with its corresponding ant (current) if becomes fitter.
  Update the current best solution if an antlion becomes fitter than the old best.
return Thebestsolution(elite)

These servers are managed by cloud service providers and provide the resources necessary to host applications and services. They can include globally distributed data centers to ensure availability and redundancy.

Users and developers mainly interact with the upper layers (Client, Application, and Platform), while cloud service providers manage the underlying infrastructure (Cloud Infrastructure and Servers).

This hierarchical approach facilitates the management and scalability of cloud services, allowing users to focus on application development and deployment without worrying about managing physical infrastructure.

3 Unique Challenges and Opportunities in Cloud Task Scheduling

The dynamic and distributed environment of cloud computing poses several challenges and opportunities when it comes to efficient task scheduling. Task scheduling policies and schemes have direct impacts on effective resource utilization and user task efficiency in the cloud.

Consequently, achieving optimal scheduling and allocation of user tasks remains a very important issue in the field of cloud computing [2].

Below, we will explore some of the most important aspects that professionals in the field propose to address to optimize the performance and effectiveness of task scheduling in this innovative environment. Among the challenges identified are:

- **Variability in Resources:** The shared nature of cloud resources introduces variability in virtual machine performance and resource availability. Task scheduling must be able to adapt to these fluctuations to ensure efficient execution.
- **Network Latency:** The geographic distribution of data centers and reliance on cloud services can lead to significant network latencies. Minimizing the impact of latency on task scheduling becomes a critical challenge.
- **Dynamic Elasticity:** The ability to scale resources on demand is essential in the cloud.

Table 1. Most commonly used performance metrics in cloud task scheduling

Performance metric	Definition
Makespan	The total time from start to completion of all scheduled tasks.
Throughput	The number of tasks completed per unit of time.
Latency	The time a task takes from request to completion.
Resource utilization	The proportion of resources (CPU, memory, etc.) used during task execution.
Resource allocation	The system's ability to allocate resources in an equitable and optimized manner.
Cost	The total expenditure associated with the execution of tasks, considering factors such as the cost of infrastructure and energy.
Load balancing	The equitable distribution of the workload among available resources.
Energy efficiency	The system's ability to perform tasks with the lowest possible energy consumption.

However, effectively implementing dynamic elasticity without compromising performance presents specific challenges.

- **Coordination and Communication:** Effective coordination between distributed tasks and efficient management of communication between components are crucial aspects to avoid bottlenecks and ensure smooth execution of tasks. In the area of opportunities, the following standout:
- **Resource Optimization:** Flexibility in cloud resource allocation provides opportunities to optimize resource utilization, reducing costs and improving energy efficiency.
- **Orchestration Services:** The increasing availability of orchestration services, such as Kubernetes, offers opportunities to simplify the management and coordination of distributed tasks.
- **Predictive Analysis:** Predictive analytics based on historical data can be used to anticipate load patterns and improve decision-making in task scheduling, thereby optimizing performance.
- **Intelligent Automation:** Applying intelligent automation techniques, such as machine learning, can improve adaptive capacity and real-time decision-making to address dynamic cloud challenges.

Exploring these challenges and opportunities will provide a more complete view of the critical aspects to consider when designing effective task scheduling strategies in the cloud computing environment.

4 Innovative Strategies and Techniques

Research in cloud task scheduling has experienced notable advances in the last decade, highlighting innovative strategies and advanced techniques. From 2019 to the present, numerous studies have explored and refined approaches to optimize task allocation in cloud computing environments, creatively addressing changing challenges. In this section, we present a summary of the most recent works in the literature, covering the period from 2019 to the present, and highlighting the strategies used by each.

With a total of 26 studies selected, an important shift towards multi-objective strategies instead of a single objective is highlighted. This change reflects the complexity of cloud environments, where optimizing a single objective may not be enough. Considering multiple performance metrics becomes essential for more adaptable and efficient solutions.

Next, we present works that adopt this multi-objective approach, highlighting the importance of considering multiple performance criteria in cloud task scheduling.

We seek to offer a comprehensive view of the latest innovations, emphasizing the effectiveness of multi-objective approaches in this context.

Pang, 2019 [3]

- Algorithm: EDA-CG
- Year: 2019
- Strategy/Technique: Estimation of distribution algorithm (EDA) and genetic algorithm (GA).
- No. Objectives: 2
- Objectives: Makespan and load balancing.

Langhnoja and Joshiyara, 2019 [4]

- Algorithm: Multi-objective based Integrated Task scheduling.
- Year: 2019
- Strategy/Technique: A ranking method to find the best possible solution.
- No. Objectives: 3
- Objectives: Makespan, cost, and load balancing.

Abdullahi et al., 2019 [5]

- Algorithm: CMSOS
- Year: 2019
- Strategy/Technique: Chaotic optimization strategy and chaotic local search strategy are applied to Pareto Fronts.
- No. Objectives: 2
- Objectives: Makespan and cost.

Abdullah et al., 2019 [6]

- Algorithm: MOPSO and MOPSO_SI
- Year: 2019
- Strategy/Technique: Multi-Objectives PSO (MOPSO) and MOPSO with Importance Strategy (IS).
- No. Objectives: 3
- Objectives: Makespan, cost, and load balancing.

Li and Han, 2020 [7]

- Algorithm: L_{ABC}
- Year: 2020
- Strategy/Technique: Hybrid discrete artificial bee colony (ABC) algorithm and permutation-based encoding method.
- No. Objectives: 3
- Objectives: Makespan, device workload, and total workloads.

Cai et al., 2020 [8]

- Algorithm: MaOEA-SIN
- Year: 2020
- Strategy/Technique: Many-objective intelligent algorithm with sine function.
- No. Objectives: 6
- Objectives: Makespan, cost, throughput, energy, resource utilization, and balancing load.

Singh et al., 2020 [9]

- Algorithm: CPO-MTS
- Year: 2020
- Strategy/Technique: Crow Search optimization Algorithm (CSA) and the Penguin Search Optimization Algorithm (PeSOA).
- No. Objectives: 4
- Objectives: Load balancing, resource utilization, makespan, and Quality of Service.

Abualigah and Diabat, 2021 [10]

- Algorithm: MALO
- Year: 2021
- Strategy/Technique: Hybrid antlion optimization algorithm with elite-based differential evolution.
- No. Objectives: 3
- Objectives: Makespan, response time (CPU), and resource utilization.

Guo, 2021 [11]

- Algorithm: Fuzzy self-defense algorithm
- Year: 2021
- Strategy/Technique: Fuzzy self-defense algorithm.
- No. Objectives: 3
- Objectives: Makespan, load balancing, and cost.

Emara et al., 2021 [12]

- Algorithm: G-MOTSA
- Year: 2021
- Strategy/Technique: Modified genetic algorithm (GA).
- No. Objectives: 6
- Objectives: Makespan, throughput, scheduling length, resource utilization, energy, and imbalance degree.

Kruekaew and Kimpan, 2022 [13]

- Algorithm: MOABCQ
- Year: 2022
- Strategy/Technique: Hybrid artificial bee colony algorithm with reinforcement learning.
- No. Objectives: 3
- Objectives: Makespan, cost, and resource utilization.

Mahmoud et al., 2022 [14]

- Algorithm: TS-DT
- Year: 2022
- Strategy/Technique: Multi-objective task scheduling algorithm is proposed based on the decision tree.
- No. Objectives: 3
- Objectives: Makespan, resource utilization, and load balancing.

Mangalampalli et al., 2022 [15]

- Algorithm: CSO
- Year: 2022
- Strategy/Technique: Cat Swarm Optimization algorithm.
- No. Objectives: 4
- Objectives: Makespan, migration time, energy, and cost.

Mangalampalli et al., 2023 [16]

- Algorithm: MOTSGWO
- Year: 2023
- Strategy/Technique: Grey wolf optimization algorithm.
- No. Objectives: 3
- Objectives: Makespan, migration time, and energy.

Cui et al., 2023 [17]

- Algorithm: MO-MFO
- Year: 2023
- Strategy/Technique: Evolutionary multi-factorial optimization algorithm.
- No. Objectives: 3
- Objectives: Makespan, cost, and load balancing.

Chandrashekar et al., 2023 [18]

- Algorithm: HWACO
- Year: 2023
- Strategy/Technique: Hybrid Weighted Ant Colony Optimization algorithm.
- No. Objectives: 2
- Objectives: Makespan and cost.

Agarwal et al., 2023 [19]

- Algorithm: HGA-ECS
- Year: 2023
- Strategy/Technique: Integration of Genetic Algorithm (GA) and Energy Conscious Scheduling (ECS) model.
- No. Objectives: 3
- Objectives: Makespan, energy consumption, and optimization of task scheduling over processors.

Mangalampalli et al., 2023 [20]

- Algorithm: MOTSWAO
- Year: 2023
- Strategy/Technique: Whale Optimization Algorithm.
- No. Objectives: 2
- Objectives: Makespan and energy consumption.

Malti et al., 2023 [21]

- Algorithm: Hybrid Multi-objective Optimization Algorithm.
- Year: 2023

- Strategy/Technique: Combination of flower pollination behavior and grey wolf optimizer strategy for task scheduling optimization.
- No. Objectives: 4
- Objectives: Makespan, resource utilization, degree of imbalance, and maximization of throughput in heterogeneous IaaS cloud environments.

Pirozmand et al., 2023 [22]

- Algorithm: IPSO
- Year: 2023
- Strategy/Technique: Multi-adaptive learning strategy to shorten the execution time of the original PSO algorithm.
- No. Objectives: 3
- Objectives: Makespan, load balancing and execution time.

Khan, 2024 [23]

- Algorithm: HLFO
- Year: 2024
- Strategy/Technique: Convolutional and Recurrent Neural Networks in a deep learning model for load calculation, Reinforcement Learning with a Hybrid Lyrebird Falcon Optimization (HLFO) algorithm.
- No. Objectives: 4
- Objectives: Makespan, energy consumption, resource utilization and Quality of Service (QoS).

Sabat et al., 2024 [24]

- Algorithm: Adaptive PSO-ACO
- Year: 2024
- Strategy/Technique: Adaptive particle swarm optimization (PSO) and ant colony optimization (ACO).
- No. Objectives: 3
- Objectives: Cost, makespan and execution time.

Gupta and Singh, 2024 [25]

- Algorithm: WOA-Scheduler
- Year: 2024
- Strategy/Technique: Whale Optimization Algorithm.
- No. Objectives: 3
- Objectives: Cost, makespan and load balancing.

Ciptaningtyas et al., 2024 [26]

- Algorithm: Improved Squirrel Search Algorithm (SSA)
- Year: 2024
- Strategy/Technique: Integration with Opposition Based Learning (OBL) method to address premature convergence.
- No. Objectives: 3
- Objectives: Makespan, throughput, and resource utilization.

Nithiavathy et al., 2024 [27]

- Algorithm: AGDESMA
- Year: 2024
- Strategy/Technique: Slime Mould Algorithm (SMA) and Adaptive Guided Differential Evolution (AGDE).
- No. Objectives: 2
- Objectives: Makespan and cost.

Behera and Sobhanayak, 2024 [28]

- Algorithm: Hybrid GA-GWO
- Year: 2024
- Strategy/Technique: Grey Wolf Optimization Algorithm (GWO) and the Genetic Algorithm (GA).
- No. Objectives: 3
- Objectives: Makespan, cost and energy consumption.

5 Critical Examination of State-of-the-Art Methods

The previous section has provided an overview of the most recent works in cloud task scheduling, highlighting innovative strategies and advanced techniques used by various researchers. Now, we will delve into a critical analysis focused on the three most representative works in this collection.

Table 2. Results obtained in [7] for the different objectives that L_{ABC}addresses

Problem	L_{ABC}					
	f_1	f_2	f_3	Fitness Value	Average Makespan	Time (s)
1	23+	19	119.65	41.13+	23+	1.25
2	297+	193.6	1642.7	535.12+	297+	0.53

Table 3. Results comparison presented in [7] of L_{ABC} algorithm against other approaches in the literature in terms of makespan

Problem	L_{ABC}		AIS		SFLA		EDA	
	f_1	Average Makespan	f_1	Average	f_1	Average.	f_1	Average
1	23+	23+	27	27	24	24	23	23.4
2	297+	297+	-	-	297	307.3	297	297.4

Table 4. Parameter settings for cloud simulation [8]

Cloud	Mips	Cost	Bandwidth	Transmission
Cloud1	300-450	0.03	1024-2048	0.01
Cloud2	500-1000	0.06	2048-3072	0.02
Cloud3	1500-2000	0.09	3072-4096	0.03

Each of these studies has contributed to the evolution of methods and approaches in task allocation optimization. We will break down in detail the techniques used, and the algorithms implemented by these selected works.

By critically examining these notable studies, we seek to provide an in-depth understanding of the key contributions that have driven the current state of the art in cloud task scheduling. This analysis will not only illustrate the strengths and limitations of each approach but will also establish a solid framework for understanding the broader landscape of research in this dynamic and constantly evolving field.

5.1 Hybrid Multi-Objective Artificial Bee Colony Algorithm

Li and Han [7] proposed an algorithm called L_{ABC}, a hybrid and improved version of the artificial bee colony (ABC) algorithm. In this approach, the initial problem is modeled as a hybrid flow shop scheduling (HFS) problem, addressing both single and multiple objectives. In the context of multi-objective HFS problems, three objectives are simultaneously considered: minimizing the makespan, the maximum workload on the device, and the total workloads on all devices. The scope of the algorithm extends to two distinct types of HFS: those with identical parallel machines and those involving unrelated machines. The proposed approach incorporates three categories of artificial bees, namely employed, observer, and scout bees, similar to the classical ABC scheme.

Each solution is represented by a string of integers. To adapt to the particularities of the problem, various perturbation structures are explored, and designed to improve the search capabilities of the algorithm.

The inclusion of an improved version of the adaptive perturbation structure in the proposed algorithm stands out, which seeks to effectively balance the exploitation and exploration capacity during the optimization process. A simple but highly effective selection strategy, along with an updated approach, is implemented to enhance the exploitation process.

To further intensify mining capabilities, a deep mining operator is introduced. In addition, an improved version of the scout bee is introduced that uses various local search methods to find the best food source or abandoned solution. This approach significantly contributes to improving the convergence ability of the proposed algorithm.

Table 5. Numerical analysis of different algorithms vs MaOEA-SIN with six objectives [8]

Algorithm	Total time (Min)	Cost (Min)	Throughput (Max)	L. Balancing (Min)	RU (Max)	Energy (Min)
	Average					
NSGA-III	5.2260×10^{5}	9.8234×10^{5}	3.9296×10^{-4}	2.6874×10^{7}	6.7570×10^{-1}	3.5439×10^{4}
VaEA	3.8351×10^{5}	7.7333×10^{5}	3.8251×10^{-4}	2.2864×10^{7}	7.9762×10^{-1}	2.7588×10^{4}
GrEA	4.7793×10^{5}	8.6360×10^{5}	3.7990×10^{-4}	2.6619×10^{7}	7.3459×10^{-1}	3.3846×10^{4}
Two_Arch2	4.7032×10^{5}	9.8155×10^{5}	3.8555×10^{-4}	2.5797×10^{7}	7.3034×10^{-1}	3.3580×10^{4}
KnEA	4.5334×10^{5}	9.0127×10^{5}	**4.0737×10^{-4}**	2.5005×10^{7}	7.3168×10^{-1}	3.2529×10^{4}
MaOEA-SIN	**2.9254×10^{5}**	**5.4047×10^{5}**	4.0393×10^{-4}	**1.9016×10^{7}**	**9.2716×10^{-1}**	**2.1341×10^{4}**

The effectiveness of the algorithm is tested using widely recognized benchmark instance sets, and performance verification of the proposed algorithm is performed.

5.2 Many-Objective Intelligent Algorithm with Sine Function

Cai et al. [8] developed a multi-objective distributed programming model that covers six objectives: total time, cost, cloud performance, energy consumption, resource utilization, and load balancing.

Furthermore, they introduced an intelligent multi-objective algorithm with a sine function to implement this model, called MaOEA-SIN. This algorithm considers the variation trend of the diversity strategy in the population, modeling it in an analogous way to the sine function.

The experimental results show outstanding programming efficiency, which contributes to improving security. This work presents a new perspective to address the challenging problem of data processing in the Internet of Things.

5.3 Multi-Objective Optimization Method using Hybrid Antlion Optimizer Algorithm

Abualigah and Diabat [10] introduced an innovative algorithm, called MALO, that combines antlion optimization with elite-based differential evolution to solve multi-objective task scheduling problems in cloud computing environments. In this method, the multi-objective nature of the problem arises from the need to minimize the makespan and maximize the resource utilization simultaneously.

The antlion optimization algorithm was improved by incorporating elite-based differential evolution as a local search technique. This approach improves the exploitability of the algorithm and prevents the possibility of getting trapped in local optima. The obtained results revealed that MALO outperformed other well-known optimization algorithms.

Notably, MALO showed faster convergence compared to other approaches when applied to larger search spaces, positioning it as a suitable option to address large-scale programming problems. In addition, a statistical analysis was carried out using t-tests, evidencing a significant improvement in the results obtained by MALO. The comprehensive evaluation of leading methods in cloud task scheduling reveals a diversity of innovative approaches and advanced strategies.

The three works examined have proven to be pioneers in the development of efficient and effective solutions to the challenges inherent in this dynamic field. Together, these works have not only contributed significantly to the current state of the art in cloud task scheduling but also provided valuable insights and foundations for future research in this dynamic and challenging field.

6 Performance Metrics and Benchmarks

In the dynamic and challenging realm of Cloud Task Scheduling, accurate evaluation of algorithm performance becomes a crucial component for efficient solution development and deployment. To carry out this evaluation, an essential set of tools is used: performance metrics and benchmarks.

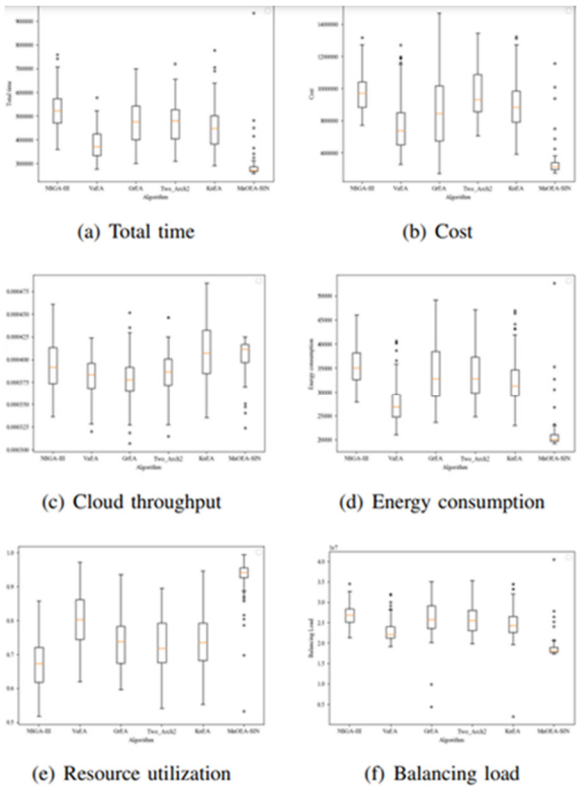

Fig. 2. Comparison of performance presented in [8] of different algorithms on six objectives

Table 6. CloudSim test settings [10]

Element	Parameter	Values
Datacenter	No. of datacenter	2
Cloudlet	No. of cloudlets	100-1000
	Length	1000-2000
Virtual machine	RAM	512 MB
	MIPS	100-1000
	Size	10000
	Bandwidth	1000
	Policy type	Time Shared
	No. of CPUs	1
Host	No. of Hosts	2
	RAM	2048 MB
	Storage	1 million
	Bandwidth	10000

These tools provide the foundation upon which researchers and developers can measure, compare, and continually improve the performance of cloud task scheduling algorithms.

6.1 Performance Metrics

Performance metrics play an essential role in evaluating and improving algorithms. These measures quantify the effectiveness and efficiency of an algorithm by providing objective information about its performance on various tasks.

Using performance metrics, developers and data scientists can evaluate effectiveness, compare algorithms, optimize parameters, diagnose problems, and perform sensitivity analysis, among other actions [29].

In the context of cloud computing, various performance metrics have been used to evaluate the efficiency and effectiveness of the algorithms used, offering detailed insight into system performance, and assisting developers in making informed decisions. Below, Table 1 presents some of the most used performance metrics in Cloud Task Scheduling.

6.2 Benchmarks

Test instances or benchmarks are collections of data created for the specific purpose of evaluating and testing algorithms. The use of instances in the evaluation of algorithms in Cloud Task Scheduling provides a structured and objective framework to analyze and improve the performance of solutions in a dynamic and distributed environment.

These instances provide an accurate representation of real-world challenges and scenarios, allowing developers to make informed decisions and refine their approaches.

There are different sets of instances widely used for the evaluation of Cloud Task Scheduling algorithms. Some notable examples include:

– Google Cluster-Trace Dataset (GoCJ) [30]: GoCJ provides real traces of jobs and tasks executed on Google clusters. It contains valuable insights into the variability and dynamics of work in large-scale cloud environments.

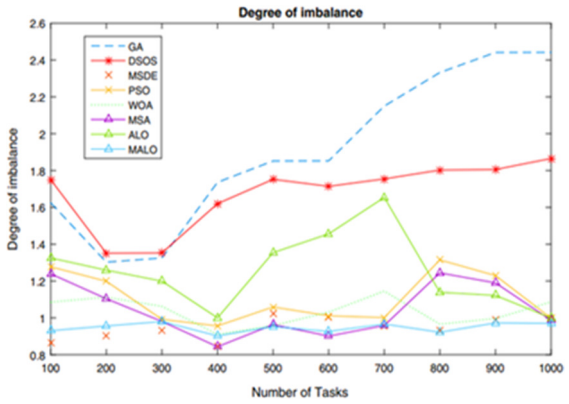

Fig. 3. Degree of imbalance of scheduling algorithms presented in [10]

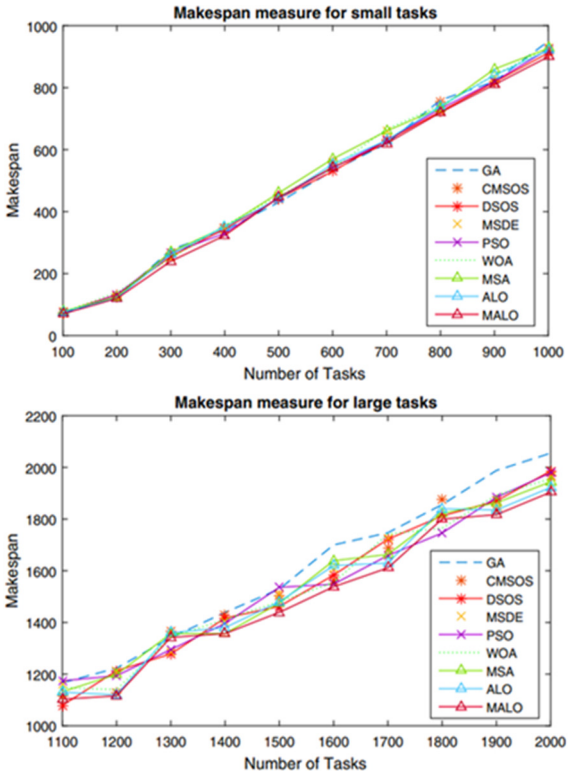

Fig. 4. The average makespan values for executing small and large tasks [10]

– NASA Ames iPSC/860 [31]: This set of instances is based on execution traces of scientific applications on the NASA Ames Research Center iPSC/860 supercomputer. Provides realistic data on scientific workloads in high-performance environments.
– HPC2N-2002 [32]: Derived from execution traces at the High-Performance Computing Center North (HPC2N) in Sweden in 2002. Contains information about the execution of jobs on a high-performance cluster.
– CEC 2005 Benchmark Functions [33]: Although most associated with benchmarking functions for optimization algorithms, the CEC 2005 instance set is also used in some cases to evaluate Cloud Task Scheduling algorithms.

These instance sets are used by the research community to evaluate and compare Cloud Task Scheduling algorithms in various contexts. Each data set presents specific characteristics that allow different aspects of performance in cloud computing environments to be simulated and analyzed.

On the other hand, there is CloudSim [34], which is a simulation framework that provides sets of simulated instances for the evaluation of cloud task scheduling algorithms. It allows you to create simulated cloud environments for performance evaluations. Programmers can generate tasks, virtual machines, and hosts randomly and with different characteristics.

CloudSim is widely used by researchers to evaluate their algorithms in different generated environments.

7 Study Cases and Future Directions

In this section, we present a detailed analysis of case studies that highlight the practical applications and results obtained by the three selected approaches: L_{ABC} [7], MaOEA-SIN [8], and MALO [10]. We will mention the instances used in their experiments, as well as the conditions under which they were carried out and the results obtained.

Through this case study analysis, we seek to provide a deeper understanding of the performance of these algorithms in real-world situations, considering different data sets and application scenarios. Furthermore, we will outline possible future directions that arise from the lessons learned and the results obtained, thus helping to guide subsequent research in task scheduling in cloud computing environments.

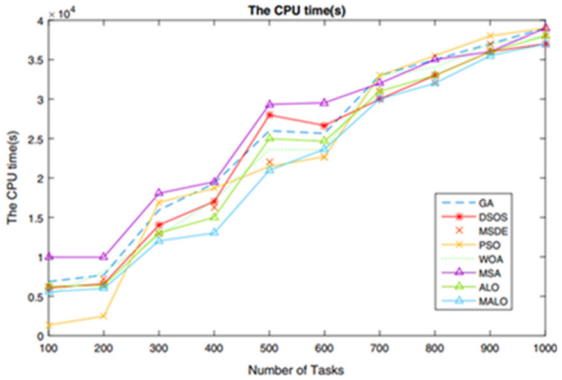

Fig. 5. The CPU time(s) of the task scheduling algorithms for the synthetic datasets [10]

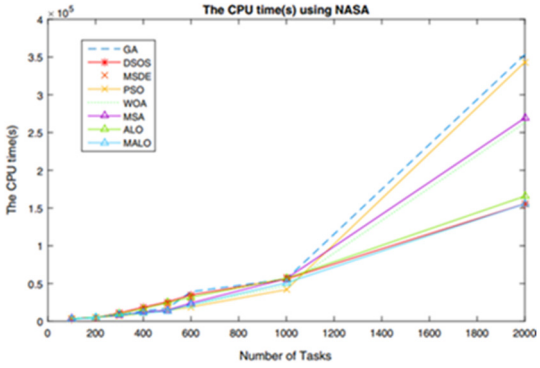

Fig. 7. The CPU time(s) of the task scheduling algorithms for solving the NASA Ames datasets [10]

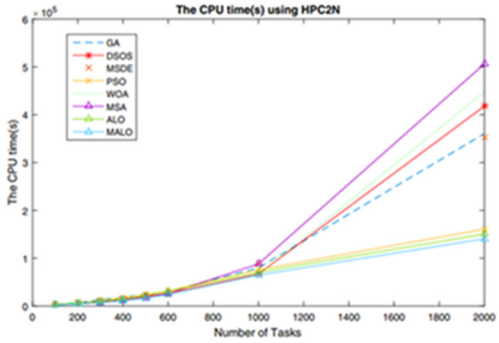

Fig. 6. The CPU time(s) of the task scheduling algorithms for solving the HPC2N Seth datasets [10]

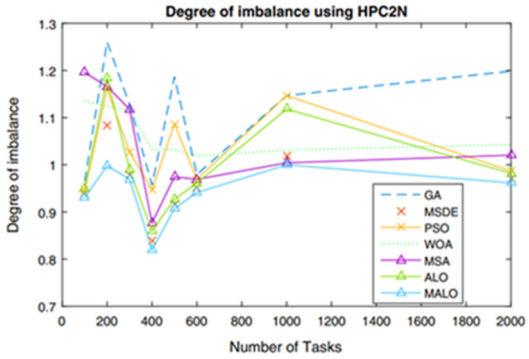

Fig. 8. The degree of imbalance of the tasks scheduling optimization algorithms using the HPC2N Seth datasets [10]

7.1 Study Cases

Li and Han Li and Han compared their proposed L_{ABC} [7] against three different algorithms from the literature: EDA (Wang et al. [35]), AIS (Liu et al. [36]), and SFLA (Xu et al. [37]). For the experimentation, they used a computer with a 3.3 GHz Intel Core i5 processor and 4 GB memory.

To test the performance of the algorithms in a multi-objective environment, they selected as instances two unrelated real machine HFS problems from [35] to make the problem more like reality in a cloud system.

The results obtained by L_{ABC} are presented in Table 2 and 3 revealing the following highlights:

1 When analyzing the comparison of results for each instance, superior performance by the algorithm is evident.

2 In terms of the average value of makespan, the computational results generated by the algorithm match the optimal values for each instance on average, thus underlining the robustness of L_{ABC}.

3 Considering the calculation times used in the test instances, the L_{ABC} algorithm also exhibits superior performance.

The L_{ABC} algorithm according to Li and Han [7] stands out for its competitive performance against various efficient algorithms. This success is based on the introduction of eight meticulously designed disturbance structures to improve exploitation

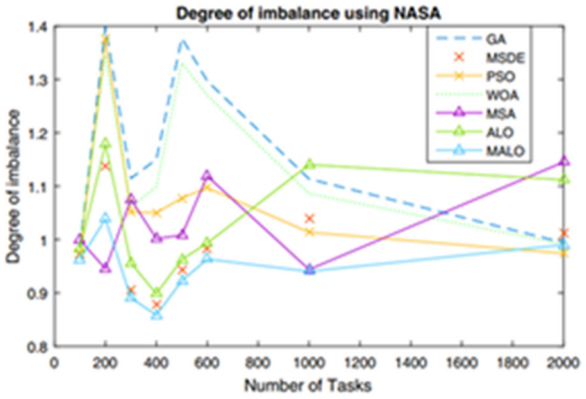

Fig. 9. The degree of imbalance of the tasks scheduling optimization algorithms using the NASA Ames datasets [10]

capacity, the use of an improved deep exploitation observer bee mechanism to intensify local search, the implementation of an adaptive perturbation that balances exploitation and exploration, and a specific approach for scout bees that boosts the convergence capacity of the algorithm.

Cai et al. [8] conducted their experiments through the simulation of three clouds with different characteristics, the main difference is the Mips execution speed and the cost. The execution cost of the first cloud is cheaper, but the execution speed is slower and takes longer, which is suitable for smaller tasks. The second cloud has a relatively medium execution speed and execution cost.

The third cloud has a faster execution speed, but the cost also becomes high, suitable for performing larger tasks. Table 4 presents the detailed parameter settings of the simulated clouds. The experimentation was carried out by generating 300 tasks with an initial length of 500 million instructions (MI), the file size is 200 KB and the output file size is 100 KB. Then each task gradually increases with a trend of 500 MI, 10 KB, and 10 KB respectively.

The comparison of the MaOEA-SIN algorithm was carried out against five algorithms from the literature: NSGA-III, VaEA, GrEA, Two_Arch2, and KnEA. The study evaluated the performance of the MaOEA-SIN algorithm in a multi-cloud model by comparing six objective values.

The best, worst, and average solutions were selected based on the performance of these values. Table 5 presents six objectives: Total time, cost, cloud throughput, load balancing, resource utilization (RU) and Energy consumption, for six different approaches.

The average results presented in Table 5, indicate that MaOEA-SIN outperforms GrEA in cost, although it shows inferior performance in the cloud throughput objective. MaOEA-SIN stands out for its excellent average performance, demonstrating strong convergence.

Figure 2 shows the distribution of the data in the form of a box plot. Looking at the upper quartile, the median, and the lower quartile, the MaOEA-SIN algorithm has better convergence and distribution for six objectives. Convergence is reflected in the fact that the midline values are all optimal values in each objective.

The distribution is reflected in the fact that the MaOEA-SIN algorithm has more dispersed points. In summary, the performance of the MaOEA-SIN algorithm is superior to that of other algorithms. To validate the effectiveness of the MALO algorithm, Abualigah and Diabat [10] present two series of experiments using synthetic datasets and real trace datasets.

For the first set of instances, the Cloudsim environment was used, which is a set of tools to imitate cloud computing scenarios [34], because the investigation of new procedures or approaches in the real cloud computing ecosystem is usually limited by solid foundations, such as protection, security, speed and the high cost of money if experiments are carried out.m Therefore, it is difficult to conduct such research in repeatable, reliable, and scalable ecosystems (environments) using real world cloud environments [10].

For experimentation, they built two data centers within CloudSim, each with two hosts. Each host has 20 GB of RAM (one host is a dual-core machine and the other is a quad-core machine) and one TB of memory storage.

Each host has a collective processing power of one million MIPS. Several virtual machines were designed with different distributions generated such as 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, and 2000 instances. The CloudSim configuration is presented in Table 6. The MALO algorithm is compared against seven approaches from the literature: (Genetic Algorithm (GA) [38], Discrete Symbiotic Organism Search (DSOS) Algorithm [39], Hybrid Moth Search Algorithm

(MSDE) [40], Particle Swarm Optimization (PSO) Algorithm [41], Whale Optimization Algorithm (WOA) [42], Moth Search Algorithm (MSA) [43], and Antlion Optimizer (ALO) Algorithm [44]).

Figure 3 presents the results obtained by the algorithms in terms of the degree of imbalance (DI), this metric shows us how equitably the tasks are distributed in the different resources. A lower degree of imbalance translates to a better use of resources and here we can see that MALO obtains the best DI compared to the other algorithms.

On the other hand, Figure 4 shows the average makespan values obtained by the algorithms for small and large tasks. The MALO algorithm reduced the value of makespan in all task cases. It is concluded that the value of the makespan increases slowly as the size of the tasks increases.

The average value of makespan when using the modified optimization algorithms is better than traditional optimization algorithms [10]. Meanwhile, the average time interval of the MALO algorithm is smaller than that of other comparative methods. Figure 5 displays the response times (CPU) achieved by different task scheduling algorithms (GA, DSOS, MSDE, PSO, WOA, MSA, ALO, and the proposed MALO).

It is highlighted that MALO achieved minimum response times to solve problems of various sizes compared to the other methods, indicating a significant improvement in the efficiency of the algorithm. Specifically, for a task size of 600, the PSO algorithm recorded the lowest response time compared to other methods. For evaluation results of real trace datasets Abualigah and Diabat used the NASA Ames dataset [32] and the HPC2N Seth dataset [33].

Figures 6 and 7 present the response times (CPU) of various task scheduling algorithms (GA, DSOS, MSDE, PSO, WOA, MSA, ALO, and MALO) when performing tasks with real trace data sets. In Figure 6, MALO manages to almost reach the minimum response time for solving tasks of all sizes compared to other methods, especially using the HPC2N Seth datasets.

Similarly, in Figure 7, MALO stands out in approaching the minimum response time for tasks of all sizes, especially with the NASA Ames datasets. The difference in algorithm response times is evident across all task sizes, particularly with the HPC2N Seth datasets, standing out over other methods. Although the difference in response times of the MALO algorithm across all task sizes is not as clear compared to DSOS, an overall improvement is observed that contributes to the reduction in the time needed to find optimal solutions.

The degree of imbalance results between the MALO algorithm and other benchmark algorithms are presented in Figures 8 and 9 for the HPC2N Seth and NASA Ames datasets. MALO achieved a higher load balance (lower degree of imbalance) compared to the other methods. In almost all cases of data sets (100-2000), MALO exhibited the lowest degree of imbalance, highlighting its superior performance compared to comparative optimization algorithms.

This is reflected in a better balance between virtual machines in all problem instances. bMALO converged faster than the other approaches for larger search spaces, making it suitable for large scheduling problems.

7.2 Future Directions

Based on insights gained from case studies of three prominent algorithms in Cloud Task Scheduling [7,8,10], we identify promising directions for future research and development.

1 Improvements in Hybrid Algorithms:

- Inspired by the success of L_{ABC} in introducing innovative perturbation structures, there is potential to explore new hybrid algorithms that combine different optimization strategies for improved performance.
- Investigating advanced exploitation mechanisms and adaptive perturbation strategies, as demonstrated in L_{ABC}, can be crucial to address the constantly evolving challenges in Cloud Task Scheduling.

2 Multi-Cloud Environments:

- Given the dynamic nature of cloud environments, future studies could focus on multi-cloud scenarios with variable execution speeds and costs. This aligns with the approach of Cai et al. by simulating three clouds with different characteristics.

- Exploring adaptive algorithms capable of dynamically adjusting to various cloud configurations can improve the adaptability of scheduling algorithms.

3 Real World Cloud Experiments:

- Bridging the gap between simulations and real-world cloud environments remains a challenge. Future research could explore methodologies for conducting repeatable, reliable, and scalable experiments in real-world cloud ecosystems, considering factors such as security, cost, and efficiency.
- Addressing the limitations associated with real-world experiments would significantly contribute to the practical applicability of the proposed scheduling algorithms.

4 Expansion of Performance Metrics:

- Extending the set of performance metrics beyond traditional objectives, such as exploring energy efficiency, security, and adaptability, can offer a more comprehensive evaluation of scheduling algorithms.
- Investigating the impact of scheduling decisions on the overall sustainability and security of cloud systems would be a valuable avenue for future research.

5 Optimization for Large-Scale Problems:

- The success of MALO in handling larger search spaces suggests the need for algorithms that can scale efficiently for large programming problems.
- Future studies could explore optimization techniques specifically designed to handle the complexity and scale associated with task scheduling in expansive cloud environments.

6 Dynamic Workloads and Task Characteristics:

- Adapting algorithms to accommodate dynamic workloads and diverse task characteristics is crucial. Future research could focus on developing scheduling approaches capable of dynamically adjusting to varying task requirements and environmental conditions.

These future directions aim to guide researchers and practitioners in advancing the field of Cloud Task Scheduling, addressing emerging challenges, and ensuring the continued evolution of efficient and adaptive scheduling algorithms.

8 Conclusions

This comprehensive study explores Cloud Task Scheduling, analyzing recent developments and offering valuable comparative insights. The importance of performance metrics and benchmarks in the evaluation of algorithms is highlighted, underlining their critical role in continuous improvement.

We provide a comprehensive overview of the latest work in the literature, highlighting the richness of multi-objective approaches that seek to simultaneously improve multiple performance metrics. This shift toward more complex and comprehensive strategies reflects the growing awareness of the multifaceted and challenging nature of cloud task scheduling.

Detailed analysis of case studies, including L_{ABC}, MaOEA-SIN and MALO, provide significant insights into their strengths and areas for improvement. The applicability and robustness of these approaches are highlighted in multi-cloud environments and with real data sets.

Promising future directions are identified, from improvements in hybrid algorithms to adaptation to real cloud environments and exploration of additional metrics.

This article aims to contribute to the continued growth of the field by providing an in-depth overview of recent developments. We seek to foster collaboration and dialogue in the academic and professional community, paving the way for future achievements in cloud task scheduling.

References

1. **Wei, X. (2020).** Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing. Journal of Ambient Intelligence and Humanized Computing, pp. 1–12. DOI: 10.1007/s12652-020-02614-7.
2. **Madni, S. H. H., Abd Latiff, M. S., Coulibaly, Y. (2016)**. Resource scheduling for

infrastructure as a service (IaaS) in cloud computing: Challenges and opportunities. Journal of Network and Computer Applications, Vol. 68, pp. 173–200. DOI: 10.10 16/j.jnca.2016.04.016.

3. **Pang, S., Li, W., He, H., Shan, Z., Wang, X. (2019).** An EDA-GA hybrid algorithm for multi-objective task scheduling in cloud computing. IEEE Access, Vol. 7, pp. 146379–146389. DOI: 10.1109/access.2019.2946216.
4. **Langhnoja, H. K., Joshiyara, P. H. A. (2019).** Multi-objective based integrated task scheduling in cloud computing. 3rd International conference on Electronics, Communication and Aerospace Technology. DOI: 10.1109/iceca.2019.8821912.
5. **Abdullahi, M., Ngadi, M. A., Dishing, S. I., Abdulhamid, S. M., Ahmad, B. I. (2019).** An efficient symbiotic organisms search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment. Journal of Network and Computer Applications, Vol. 133, pp. 60–74. DOI: 10.1016/j.jnca.2019.02.005.
6. **Abdullah, M., Al-Muta'a, E. A., Al-Sanabani, M. (2019)**. Integrated MOPSO algorithms for task scheduling in cloud computing. Journal of Intelligent & Fuzzy Systems, Vol. 36, No. 2, pp. 1823–1836. DOI: 10.3233/jifs-181005.
7. **Li, J., Han, Y. (2019).** A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system. Cluster Computing, Vol. 23, No. 4, pp. 2483–2499. DOI: 10.1007/s10586-019-03022- z.
8. **Cai, X., Geng, S., Wu, D., Cai, J., Chen, J. (2021).** A multicloud-model-based many-objective intelligent algorithm for efficient task scheduling in internet of things. IEEE Internet of Things Journal, Vol. 8, No. 12, pp. 9645–9653. DOI: 10.1109/jiot.2020.3040019.
9. **Singh, H., Tyagi, S., Kumar, P. (2020).** Crow–penguin optimizer for multiobjective task scheduling strategy in cloud computing. International Journal of Communication Systems, Vol. 33, No. 14, pp. e4467. DOI: 10.1 002/dac.4467.
10. **Abualigah, L., Diabat, A. (2020).** A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. Cluster Computing, Vol. 24, No. 1, pp. 205–223. DOI: 10.1007/ s10586-020-03075-5.
11. **Guo, X. (2021).** Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm. Alexandria Engineering Journal, Vol. 60, No. 6, pp. 5603–5609. DOI: 10.1016/j.aej.20 21.04.051.
12. **Emara, F. A., Gad-Elrab, A., Sobhi, A., Raslan, K. R. (2021).** Genetic-based multi-objective task scheduling algorithm in cloud computing environment. International Journal of Intelligent Engineering and Systems, Vol. 14, No. 5, pp. 571–582. DOI: 10.22266/ijies20 21.1031.50.
13. **Kruekaew, B., Kimpan, W. (2022).** Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning. IEEE Access, Vol. 10, pp. 17803–17818. DOI: 10.1109/access.2022. 3149955.
14. **Mahmoud, H., Thabet, M., Khafagy, M. H., Omara, F. A. (2022).** Multiobjective task scheduling in cloud environment using decision tree algorithm. IEEE Access, Vol. 10, pp. 36140–36151. DOI: 10.1109/access.2022. 3163273.
15. **Mangalampalli, S., Swain, S. K., Mangalampalli, V. K. (2021).** Multi objective task scheduling in cloud computing using cat swarm optimization algorithm. Arabian Journal for Science and Engineering, Vol. 47, No. 2, pp. 1821–1830. DOI: 10.1007/s13369-021-06076-7.
16. **Mangalampalli, S., Karri, G. R., Kumar, M. (2022).** Multi objective task scheduling algorithm in cloud computing using grey wolf optimization. Cluster Computing, Vol. 26, No. 6, pp. 3803–3822. DOI: 10.1007/s10586-022-03786-x.
17. **Cui, Z., Zhao, T., Wu, L., Qin, A. K., Li, J. (2023).** Multi-objective cloud task scheduling optimization based on evolutionary multi-factor algorithm. IEEE Transactions on Cloud

Computing, Vol. 11, No. 4, pp. 3685–3699. DOI: 10.1109/tcc.2023.3315014.

18. **Chandrashekar, C., Krishnadoss, P., Poornachary, V. K., Ananthakrishnan, B., Rangasamy, K. (2023).** HWACOA scheduler: Hybrid weighted ant colony optimization algorithm for task scheduling in cloud computing. Applied Sciences, Vol. 13, No. 6, pp. 3433. DOI: 10.3390/app13063433.
19. **Agarwal, G., Gupta, S., Ahuja, R., Rai, A. K. (2023).** Multiprocessor task scheduling using multi-objective hybrid genetic algorithm in fog–cloud computing. Knowledge-Based Systems, Vol. 272, pp. 110563. DOI: 10.1016/j.knosys.2023.110563.
20. **Mangalampalli, S., Karri, G. R., Kose, U. (2023).** Multi objective trust aware task scheduling algorithm in cloud computing using whale optimization. Journal of King Saud University - Computer and Information Sciences, Vol. 35, No. 2, pp. 791–809. DOI: 10.1016/j.jksuci.2023.01.016.
21. **Malti, A. N., Hakem, M., Benmammar, B. (2023).** A new hybrid multi-objective optimization algorithm for task scheduling in cloud systems. Cluster Computing. DOI: 10.1007/s10586-023-04099-3.
22. **Pirozmand, P., Jalalinejad, H., Hosseinabadi, A. A. R., Mirkamali, S., Li, Y. (2023).** An improved particle swarm optimization algorithm for task scheduling in cloud computing. Journal of Ambient Intelligence and Humanized Computing, Vol. 14, No. 4, pp. 4313–4327. DOI: 10.1007/s12652-023-04541-9.
23. **Khan, A. R. (2024).** Dynamic load balancing in cloud computing: optimized RL-based clustering with multi-objective optimized task scheduling. Processes, Vol. 12, No. 3, pp. 519. DOI: 10.3390/pr12030519.
24. **Sabat, N. R., Sahoo, R. R., Pradhan, M. R., Acharya, B. (2024).** Hybrid technique for optimal task scheduling in cloud computing environments. TELKOMNIKA (Telecommunication Computing Electronics and Control), Vol. 22, No. 2, pp. 380. DOI: 10.12928/telkomnika.v22i2.25641.
25. **Gupta, S., Singh, R. S. (2024).** User-defined weight based multi objective task scheduling in cloud using whale optimization algorithm. Simulation Modelling Practice and Theory, Vol. 133, pp. 102915. DOI: 10.1016/j.simpat.2024.102915.
26. **Ciptaningtyas, H. T., Shiddiqi, A. M., Purwitasari, D., Rosyadi, F. D., Fauzan, M. N. (2024).** Multi-objective task scheduling algorithm in cloud computing using improved squirrel search algorithm. International Journal of Intelligent Engineering & Systems, Vol. 17, No. 1.
27. **Nithiavathy, R., Janakiraman, S., Priya, M. D. (2023).** Adaptive guided differential evolution-based slime mould algorithm-based efficient multi-objective task scheduling for cloud computing environments. Transactions on Emerging Telecommunications Technologies, Vol. 35, No. 1. DOI: 10.1002/ett.4902.
28. **Behera, I., Sobhanayak, S. (2024).** Task scheduling optimization in heterogeneous cloud computing environments: a hybrid GA-GWO approach. Journal of Parallel and Distributed Computing, Vol. 183, pp. 104766. DOI: 10.1016/j.jpdc.2023.104766.
29. **González-San-Martín, J., Cruz-Reyes, L., Gómez-Santillán, C., Fraire, H., Rangel-Valdez, N., Dorronsoro, B., Quiroz-Castellanos, M. (2023).** Comparative study of heuristics for the one-dimensional bin packing problem. Studies in Computational Intelligence, pp. 293–305. DOI: 10.1007/978-3-031-28999-6_19.
30. **Hussain, A., Aleem, M. (2018).** GoCJ: google cloud jobs dataset for distributed and cloud computing infrastructures. Data, Vol. 3, No. 4, pp. 38. DOI: 10.3390/data3040038.
31. **Feitelson, D. G., Nitzberg, B. (1995).** Job characteristics of a production parallel scientific workload on the NASA Ames iPSC/860. Lecture Notes in Computer Science, pp. 337–360. DOI: 10.1007/3-540-60153-8_38.
32. **HPC2N. (n.d.).** High performance computing center north (HPC2N) is a national center for scientific and parallel computing. https://www.hpc2n.umu.se/.

33. **Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A., Tiwari, S. (2005).** Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. KanGAL report, pp. 1–50.
34. **Calheiros, R. N., Ranjan, R., De-Rose, C. A., Buyya, R. (2009)**. CloudSim: A novel framework for modeling and simulation of cloud computing infrastructures and services. ArXiv. DOI: 10.48550/ARXIV.0903.2525.
35. **Wang, S., Wang, L., Liu, M., Xu, Y. (2013).** An enhanced estimation of distribution algorithm for solving hybrid flow-shop scheduling problem with identical parallel machines. The International Journal of Advanced Manufacturing Technology, Vol. 68, No. 9-12, pp. 2043–2056. DOI: 10.1007/s00170-013- 4819-y.
36. **Liu, F., Zhang, X., Zou, F., Zeng, L. (2009).** Immune clonal selection algorithm for hybrid flow-shop scheduling problem. Chinese Control and Decision Conference. DOI: 10.1109/ccdc.2009.5194868.
37. **Xu, Y., Wang, L., Zhou, G., Wang, S. (2011).** An effective shuffled frog leaping algorithm for solving hybrid flow-shop scheduling problem. Lecture Notes in Computer Science, pp. 560–567. DOI: 10.1007/978-3-642-24728-6_76.
38. **Zhang, L., Li, K., Li, C., Li, K. (2017).** Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems. Information Sciences, Vol. 379, pp. 241–256. DOI: 10.1016/j.ins. 2016.08.003.
39. **Abdullahi, M., Ngadi, M. A., Abdulhamid, S. M. (2016).** Symbiotic organism search optimization based task scheduling in cloud computing environment. Future Generation Computer Systems, Vol. 56, pp. 640–650. DOI: 10.1016/j.future.2015.08.006.
40. **Elaziz, M. A., Xiong, S., Jayasena, K., Li, L. (2019).** Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. Knowledge-Based Systems, Vol. 169, pp. 39–52. DOI: 10.1016/j. knosys.2019.01.023.
41. **Zuo, X., Zhang, G., Tan, W. (2014).** Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud. IEEE Transactions on Automation Science and Engineering, Vol. 11, No. 2, pp. 564–573. DOI: 10.1109/tase.2013.2272758.
42. **Mirjalili, S., Lewis, A. (2016).** The whale optimization algorithm. Advances in Engineering Software, Vol. 95, pp. 51–67. DOI: 10.1016/j.advengsoft.2016.01.008.
43. **Wang, G. (2016).** Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. Memetic Computing, Vol. 10, No. 2, pp. 151–164. DOI: 10.1007/ s12293-016-0212-3.
44. **Mirjalili, S. (2015).** The ant lion optimizer. Advances in Engineering Software, Vol. 83, pp. 80–98. DOI:10.1016/j.advengsoft.2015. 01.010.

Article received on 31/01/2024; accepted on 24/04/2024.
**Corresponding author is Jessica González-San-Martín.*