*Proceedings of the International Conference on Industrial Engineering and Operations Management*

Publisher: IEOM Society International, USA DOI: 10.46254/AF6.20250096
Published: April 8, 2025

# Optimizing Task Scheduling in Cloud Computing: A Comparative Analysis of Min-Min, Max-Min, AIGA, and ELOA

**Kıvılcım Naz Böke**
Department of Industrial Engineering
Çankaya University,Ankara
Türkiye
c2012056@student.cankaya.edu.tr

**Syed Shah Sultan Mohiuddin Qadri**
Department of Industrial Engineering
Çankaya University,Ankara
Türkiye
syedshahsultan@cankaya.edu.tr

**Ahmet Kabarcık**
Department of Industrial Engineering
Çankaya University,Ankara
Türkiye
a.kabarcik@cankaya.edu.tr

## Abstract

Cloud computing is one of the latest commercial ideas that has become popular and provides users with limitless access to virtual resources. Task scheduling in cloud computing refers to assigning VMs to cloud tasks. Due to service demand costs associated with the usage of software and hardware, such as bandwidth, storage, and processing, effective task scheduling and balancing the usage of VMs are essential for increasing data center efficiency and reducing energy consumption. Therefore, using an effective task scheduling algorithm that minimizes makespan and maximizes resource utilization is crucial. In many studies, different algorithms—such as Min-Min, Max-Min, and Adaptive Incremental Genetic Algorithm (AIGA)—have been developed to solve this problem, and these algorithms are used on many platforms. The goal of this study is to compare commonly used heuris tic-based algorithms, which include Min-Min and Max-Min, bio-inspired algorithm AIGA, and offer an exact solution guarantee through the Exhaustive Learning Optimization Algorithm (ELOA). With this aim, the instance types of Amazon EC2 have been used to implement virtual machines with various computing capacities on CloudSim. The results show that Min-Min and Max-Min algorithms provide computationally efficient feasible solutions, and AIGA also provides feasible solutions with adaptive and efficient processes. Finally, ELOA provides an optimal solution that minimizes makespan. This makes Min-Min, Max-Min, and AIGA suitable for large-scale task scheduling in cloud computing, while ELOA is a great choice for small-scale problems.

## Keywords
Cloud Computing, Task Scheduling, Virtual Machines, Optimization, Exhaustive Learning Optimization Algorithm.

## Introduction

Nowadays, information technologies are becoming more important. In many industrial fields, technological innovations are increasingly used as they increase efficiency and service delivery (Ullah et al. 2024; Zhang et al. 2023). Consequently, internet usage in society has increased considerably, leading to changes in consumer behavior and expectations (Naz et al. 2023). This change necessitates more efficient use of computing resources to effectively meet the needs of a large and ever-growing user base (Gulbaz et al. 2021). Examples include robust data processing capabilities, enhanced storage solutions, and advanced networking infrastructure to guarantee absolute and reliable access to information and services (Sahu et al. 2012; Gajbhiye & Shrivastva 2014). Thus, the increasing needs of consumers require new technological solutions and the continuous development of information technology.

As one of the biggest technological solutions of recent years, Cloud Computing systems are used in many organizations thanks to their features such as scalability, dynamic and high-capacity computing (Radu 2017). Therefore, it becomes one of the biggest paradigms in the IT sector, providing applications such as business intelligence and data archiving to users (Rasheed 2014). According to the National Institute of Standards and Technology's (NIST), Cloud Computing is a concept that makes it possible to access a shared network environment anywhere at any time (Mell & Grance et al. 2011). Cloud computing offers three main service models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) (Mell & Grance et al. 2011; Duan et al. 2018; Mahmood et al. 2017). SaaS is a service model where a service provider hosts applications and consumers can access the applications from various client devices through a thin client interface, such as a web browser or a program interface. In a PaaS model, users are provided with libraries, programming languages, services, and tools to develop applications. IaaS systems provide users with computing resources that can include an arbitrary number of operating systems and applications on demand. These services are delivered with five essential characteristics: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service (Mell & Grance et al. 2011). With the on-demand self-service characteristic, consumers can access computing capabilities unilaterally when needed. Broad network access enables users to get benefits from the features of the cloud environment through standard networking methods.

Resource pooling provides a valuable opportunity for network providers to dynamically aggregate computing resources while efficiently serving multiple users at the same time. Rapid elasticity ensures the flexible scaling of resources, adapting quickly to demand fluctuations. Additionally, measured service enables automated resource monitoring and control, optimizing efficiency and performance. These fundamental characteristics make cloud computing a highly adaptable and scalable solution for diverse industrial and research applications.Resources are instantaneously made available to users based on their needs thanks to cloud computing, creating a productive and dynamic ecosystem that promotes communication between the main stakeholders: users and cloud service providers (Duan et al. 2018). While consumers primarily focus on service costs, service providers aim to solve more users' computing and storage tasks while reducing expenses and energy consumption (Liu & Wang 2020). Therefore, developing solutions to reduce costs for both consumers and service providers forms the basis of a sustainable and productive ecosystem.

Among the services, this study focuses on IaaS, which provides customers with VMs that allow them to operate without physical devices. However, the increasing number of customers makes it challenging to allocate resources based on individual needs, although Cloud Computing provides service for complex tasks. This challenge highlights the importance of task scheduling and resource utilization in VMs. Task scheduling algorithms are the most efficient way to address these issues.

### 1.1 Objectives

This study aims to compare features of two popular heuristic algorithms, including Min-Min and Max-Min, the bio-inspired algorithm Adaptive Incremental Genetic Algorithm (AIGA) and Exhaustive Learning Optimization Algorithm (ELOA). With the help of these algorithms, it is aimed to minimize makespan in task scheduling, which is essential to reducing cost and energy consumption. In addition, the performance of these algorithms has also been scrutinized through a series of extensive sets of experiments in order to compare the precision and optimality of the solutions provided by these algorithms.

## 2. Literature Review

Management and the distribution of computational resources have been enhanced significantly by cloud computing. This has been done by offering both scalable and efficient solutions tailored to a wide range of computational requirements. These innovative solutions provide organizations with a valuable opportunity to seamlessly adapt to evolving demands, allowing them to optimize resource efficiency and effectiveness for peak performance. One of the most significant challenges in this domain is task scheduling, a problem that is inherently complex and classified as NP-hard, making optimal workload distribution a difficult task (Ebadifard & Babamir 2018). Effective task scheduling is crucial, as it directly impacts operational costs and overall cloud performance. This literature review explores different algorithms designed to tackle this challenge, with a focus on widely used heuristic algorithms, a bio-inspired approach, and an exhaustive search optimization algorithm.

Heuristic algorithms, such as Min-Min (Laszewski 2003) and Max-Min (Mao et al. 2014), are widely used due to their simplicity and efficiency in various scenarios. The Min-Min algorithm begins by identifying all unassigned tasks and then allocates each one to the machine with the shortest completion time. It prioritizes tasks with the smallest overall completion time in each iteration, gradually reducing makespan (Braun et al. 2001). In contrast, the Max-Min algorithm improves resource utilization by focusing on the longest tasks first. It determines the minimum completion time for each task across available machines and assigns the task with the highest of these minimum times. This approach is particularly beneficial in environments that seek to optimize the performance of slower machines by assigning them the most computationally demanding tasks early in the scheduling process. While Min-Min and Max-Min are effective in reducing makespan, their heuristic nature often prevents them from finding the most optimal solution, especially in more complex and diverse task environments (Duan et al. 2018). These algorithms assume uniformity in task execution and machine performance, which does not always hold in dynamic cloud environments. Furthermore, although they are fast and easy to implement, they can lead to inefficient task distribution, particularly under high workloads or when tasks have varying computational demands.

To overcome these limitations, researchers have explored alternative algorithms inspired by natural processes. One notable example is the AIGA algorithm proposed by Duan et al. (2018). AIGA is a bio-inspired approach that mimics the adaptability of biological systems to optimize task scheduling. By dynamically adjusting its parameters based on system conditions, AIGA enhances search efficiency and adaptability, making it highly suitable for cloud environments where scheduling needs frequently shift and require flexibility.This study aims to compare various task scheduling algorithms—Min-Min, Max-Min, AIGA, and ELOA—to evaluate their applicability and efficiency in cloud computing. By analyzing their performance under different conditions, this research seeks to highlight their strengths and limitations, offering insights into their practical usability and guiding future advancements in cloud task scheduling. Through this comparative analysis, the study aims to draw definitive conclusions on the scalability, efficiency, and practicality of each algorithm in handling the dynamic and often unpredictable workloads of cloud computing environments.

## 3. Proposed Methodology

With the aim of minimizing makespan, scheduling jobs on parallel machines is well-known as a complex issue and is typically classified as NP-Hard. For these types of challenges, heuristic algorithms are often used to find solutions that may not necessarily give the optimal solution but provide near-optimal solutions (Kokash N., 2005; Hosseini Shirvani 2020). These algorithms are particularly effective for tackling large-scale and computationally intensive problems. However, in small-scale problems, ELOA can be utilized if the objective is to find the exact optimal solution. In this study, Heuristic algorithms Min-Min, Max-Min, bio-inspired algorithm AIGA, and ELOA are focused on addressing the task scheduling problem in a cloud environment. On the exhaustive end of the spectrum lies the ELOA, which, unlike heuristic approaches, attempts to find the optimum solution by exploring all possible task assignments

### 3.1 Task Scheduling Problem

In cloud architecture, a data center is made up of several physical servers that run VMs based on user requests. Task scheduling in the cloud environment involves two main problems: selecting which host to deploy a VM and determining which VM to allocate to each task. To divide large-scale tasks into small subtasks, the Map/Reduce model is commonly used in the Cloud Environment. These smaller tasks are then assigned to Cloud servers based on their status and the quality-of-service (QoS) requirements. This structure of the cloud environment is represented in Figure 1.

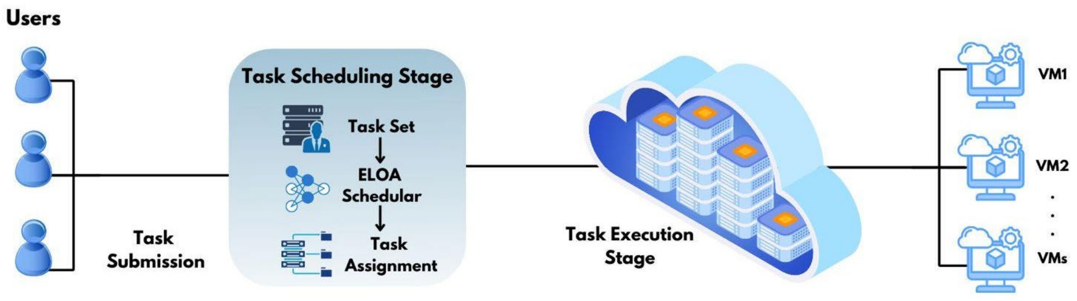


Figure 1. Abstract Representation of the Cloud Environment.

Implementing a proper scheduling algorithm is crucial to minimizing task completion time and associated costs. In this context, makespan can be considered an indicator of cost and used as an objective function.
VMs are generally specified by CPU, memory bandwidth, and storage. For instance, Amazon uses a Compute Unit (CU) to describe the CPU capacities. Since cloud services are billed by the number of hours consumed, CUs can be used to estimate the necessary execution time before requesting cloud resources. An EC2 Compute Unit (ECU) is equal to the CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor (Wen et al. 2015). Assume we have $m$ tasks $T = \{t_1, t_2, \dots, t_m\}$, and $n$ VMs $VM = \{vm_1, vm_2, \dots, vm_n\}$. Given that cloud services are paid for based on usage hours, CU can be used to estimate the needed execution time. The execution time $t_{ij}$ for task $t_i$ on

**VM $vm_j$ can be calculated as**:

$$t_{ij} = \frac{L_i}{C_j} \tag{1}$$

Where:

$$L_i: Length\ of\ task\ t_i, i \in \{1,2,\dots,m\}$$
$$C_j: CU\ of\ virtual\ machine\ vm_j, j \in \{1,2,\dots,n\}$$

The total occupation time for VM $vm_j$ can be calculated by:

$$T_j = \sum_{i=1}^{m} x_{ij} \cdot t_{ij} \tag{2}$$

Where $x_{ij} \in \{0,1\}$ and:

$$x_{ij} = \{1, \quad if\ t_i\ is\ allocated\ on\ vm_j\ 0, \quad otherwise$$

## 3.2 Heuristic Based Task Scheduling Algorithms

**Max-Min Algorithm**

Using the Max-Min algorithm, the minimum completion time of each unmapped task across all computers is determined. The task with the maximum of these minimum completion times is then selected. The Max-Min algorithm is particularly suitable for problems with a large number of tasks that have shorter execution times, more than those with longer execution times, as it may lead to a rise in the makespan for submitted tasks (Aref et al. 2022). The Max-Min algorithm can be implemented with the following steps:

*Step 1:* Calculate Completion time for each task on each VM:

$$C_{ij} = S_j + t_{ij} \tag{3}$$

Where:

$$C_{ij}: Completion\ time\ of\ task\ t_i\ on\ VM\ vm_j$$

$$S_j: Start\ time\ of\ VM\ vm_j$$
$$t_{ij}: Execution\ time\ of\ task\ t_i\ on\ VM\ vm_j$$

Step 2: Determine the minimum completion time for each task:

$$min_j(C_{ij}), \quad for\ each\ i\ \epsilon\ \{1,2,\ldots,m\} \tag{4}$$

Step 3: Select the task with the maximum value among these minimum completion times:

$$max_i\left(min_j(C_{ij})\right) \tag{5}$$

Step 4: Allocate the selected task to the corresponding VM and update the start time.

$$S_j = C_{ij} \tag{6}$$

**B. Min-Min**

The Min-Min algorithm prioritizes tasks with the overall minimum completion time at each step and allocates each unmapped task to the machine that offers the minimum completion time. Since the Min-Min algorithm increases the wait time for large tasks by prioritizing smaller tasks, it is more useful when the number of smaller tasks is greater than the number of larger tasks. The Min-Min algorithm can be implemented with the following steps:

*Step 1:* Calculate Completion time for each task on each VM using Equation (3)

Step 2: Determine the minimum completion time for each task using Equation (4)

Step 3: Select the task with the minimum value among these minimum completion times:

$$min_i\left(min_j(C_{ij})\right) \tag{7}$$

Step 4: Allocate the selected task to the corresponding VM and update the start time using Equation (6).

While the Min-Min algorithm is efficient and yields a relatively low makespan, and the Max-Min algorithm achieves better load balancing, both fail to guarantee optimal solutions due to their heuristic nature. For small-scale problems, ELOA can be employed to explore all possible task allocations, ensuring the optimal solution.

**C. Adaptive Incremental Genetic Algorithm**

Genetic Algorithm (GA) is a bio-inspired algorithm based on Darwin's Theory of Evolution and the principle of natural genetics (Duan et al. 2018). In the GA, **fitness value is used for the selection**. With the ***greater fitness value***, **individuals have a higher probability of entering the next generation**. In the crossover operation, individuals inherit characteristics from both parents. This process helps in preserving good traits. Also, the mutation operation contributes to producing new individuals. During GA, **chromosomes representing candidate solutions** are generated with these genetic operations. In AIGA, some tasks are selected from independent task sets for scheduling at each iteration. This approach helps reduce the computation time compared to a standard GA by limiting the number of tasks processed. The pseudocode of the AIGA is represented as Algorithm 1 (Duan et al. 2018).

**Algorithm 1. AIGA**

**Input:** Population size (Np), Maximum iterations (max_iter), Objective function, Control parameters, Task list (T), Virtual Machine (VM) types, Interval size
**Output:** The global best solution

```
1:  Select a subset of tasks from T, where the number of tasks equals the interval size.
2:  Initialize population P with the number of Np individuals.
3:  Evaluate each individual in P.
4:  Set iteration = 0.
5:  While iteration < max_iter do:
6:      Calculate the probability to be chosen of each individual.
7:      Select individuals for the new population P' based on their probabilty.
8:      Sort P' in descending order of fitness.
```

```
9:     Compute mutation and crossover rates for each individual in P'.
10:     For each individual in P':
11:        If rand(0,1) < crossover_rate, perform crossover.
12:        If rand(0,1) < mutation_rate, perform mutation.
13:     End for
14:     Update the global best solution if a better one is found.
15:     iteration = iteration + 1.
16: End while
17: Update the occupation time of each virtual machine.
```

## 3.2 Exhaustive Learning Optimization Algorithm

This approach essentially tries every possible combination to reach the final optimal solution. With progress in technology and an increase in computer processing speed, ELOA has become more practical. This study focuses on the viewpoint of finding optimal solutions for task scheduling in the cloud environment. Algorithm 2 represents the pseudocode of ELOA.

**Algorithm 2. ELOA**
**Input:** Execution time matrix, task list, VM list
**Output:** The best task order and minimum makespan

```
1: Initialize the best makespan as infinity
2: for each permutation of the task order do
3:     Initialize VM times to zero
4:     for each task in the current order do
5:         Select the VM with the minimum (makespan + execution time of task)
6:         Assign a task to the selected VM
7:         Update the VM's makespan
8:     end for
9:     Calculate the makespan of the current task order
10:     if current makespan < best makespan then
11:         Update the best makespan
12:         Save the current task order as the best order
13:     end if
14: end for
15: Return the best task order and best makespan
```

In Algorithm 2, task orders are created randomly using permutations. Then, each task in the task order is assigned to the VM, which minimizes the sum of the machine's current makespan and the task's execution time. After assigning a task, the makespan of the selected VM is updated. This process is repeated for every task in the task sequence. Once all tasks in the order are assigned to VMs, the makespan for that order is calculated. If this new makespan is smaller than the best makespan found so far, it is updated as the best makespan. The process is then repeated for all possible task orders, and after all trials, the optimal makespan is determined.

## 4. Data Collection

The VM (instance) specifications used in this study reference the Amazon Elastic Compute Cloud schema, as detailed in Table 1 (Duan et al. 2018). The table presents the VM types along with their corresponding CU.

Table 1. VM Types and CU

| VM No. | Instance Type | CU |
|---|---|---|
| 1 | c3: large | 8 |
| 2 | c3: xlarge | 16 |
| 3 | c3: 2xlarge | 31 |
| 4 | c3: 3xlarge | 62 |
| 5 | c3: 4xlarge | 132 |
| 6 | c4: xlarge | 7 |
| 7 | c4: 2xlarge | 14 |
| 8 | c4: 3xlarge | 28 |
| 9 | c4: 4xlarge | 55 |
| 10 | c4: 8xlarge | 108 |

For the analysis, ten tasks with lengths in minutes [814.0,447.0,610.0,319.0, 876.0, 692.0, 663.0, 631.0, 626.0, 655.0] are utilized in this study (Duan et al. 2018). Using Equation (1), the execution times required by each VM to complete the assigned tasks are calculated, with the results displayed in Table 2 (Duan et al. 2018).

Table 2. Execution Time of Each VM

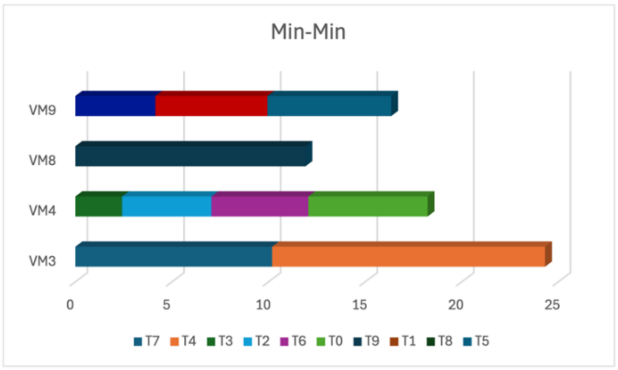
| | | **Virtual Machines (VMs)** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **VM0** | **VM1** | **VM2** | **VM3** | **VM4** | **VM5** | **VM6** | **VM7** | **VM8** | **VM9** |
| **Tasks** | **0** | 101.75 | 50.88 | 26.26 | 13.13 | 6.17 | 116.29 | 58.14 | 29.07 | 14.8 | 7.54 |
| | **1** | 55.88 | 27.94 | 14.42 | 7.21 | 3.39 | 63.86 | 31.93 | 15.96 | 8.13 | 4.14 |
| | **2** | 76.25 | 38.13 | 19.68 | 9.84 | 4.62 | 87.14 | 43.57 | 21.79 | 11.09 | 5.65 |
| | **3** | 39.88 | 19.94 | 10.29 | 5.15 | 2.42 | 45.57 | 22.79 | 11.39 | 5.8 | 2.95 |
| | **4** | 109.5 | 54.75 | 28.26 | 14.13 | 6.64 | 125.14 | 62.57 | 31.29 | 15.93 | 8.11 |
| | **5** | 86.5 | 43.25 | 22.32 | 11.16 | 5.24 | 98.86 | 49.43 | 24.71 | 12.58 | 6.41 |
| | **6** | 82.88 | 41.44 | 21.39 | 10.69 | 5.02 | 94.71 | 47.36 | 23.68 | 12.05 | 6.14 |
| | **7** | 78.88 | 39.44 | 20.35 | 10.18 | 4.78 | 90.14 | 45.07 | 22.54 | 11.47 | 5.84 |
| | **8** | 78.25 | 39.13 | 20.19 | 10.1 | 4.74 | 98.43 | 44.71 | 22.36 | 11.38 | 5.8 |
| | **9** | 81.88 | 40.94 | 21.13 | 10.56 | 4.96 | 93.57 | 46.79 | 23.39 | 11.91 | 6.06 |

## 5. Results and Discussion

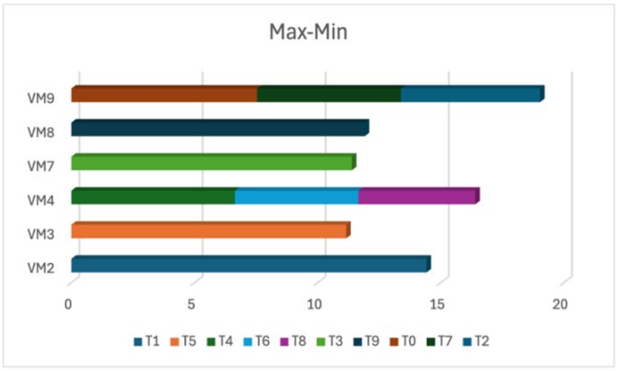
### 5.1 Numerical Results

The task orders generated by the Min-Min, Max-Min, AIGA, and ELOA are [3,1,2,8,7,9,6,5,0,4], [4,0,5,6,9,7,8,2,1,3], [0,2,1,4,5,8,3,6,9,7] and [0, 2, 1, 5, 9, 4, 3, 6, 7, 8] respectively. Furthermore, the makespan for these algorithms is 24.31, 19.5, 18.12, and 17.5, respectively.

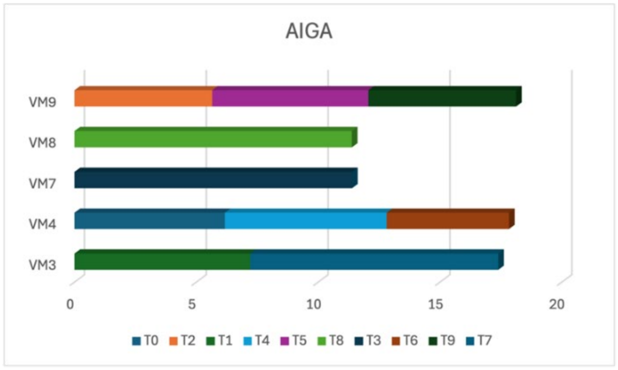
As the experiment demonstrates, ELOA achieves improvements of 28.01%, 10.26%, and 3.42% compared to the Min-Min, Max-Min, and AIGA algorithms. Figure 2 illustrates the task assignment schemes to virtual machines and corresponding makespan values for the Min-Min, Max-Min, AIGA, and ELOA algorithms.



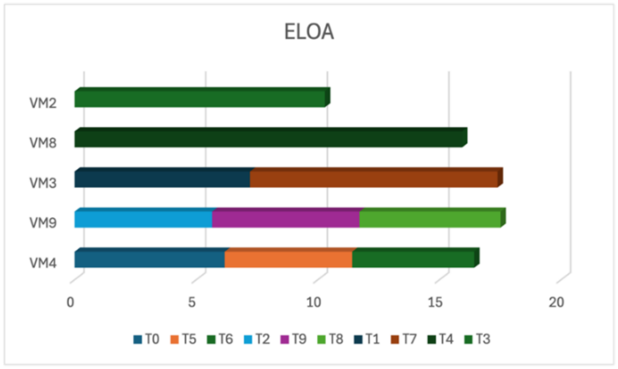
*a) Min-Min*



*b) Max-Min*



*c) AIGA*



*d) ELOA*

Figure 2. Assignment of Tasks to VM Schemes and Makespan Using Algorithms

## 5.2 Validation

For a more comprehensive evaluation of the previously mentioned algorithms for the scheduling of task operations under cloud computing, a series of extensive sets of experiments was conducted using randomly generated diverse task lengths. These task lengths are detailed in Table 3, where each row represents a unique test case scenario, and each column corresponds to the length of a specific task. With this structured evaluation approach, a critical analysis of each algorithm's efficiency and robustness in terms of varying computational environments has been conducted.

Table 3. Task Lengths for the Test Cases Used in the Performance Evaluation of Algorithms

| | | **Length of Tasks** | | | | | | | | | | | | **Length of Tasks** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **Scenarios** | **1** | 273 | 796 | 928 | 486 | 719 | 319 | 920 | 280 | 812 | 160 | **Scenarios** | **11** | 264 | 944 | 417 | 949 | 966 | 214 | 813 | 554 | 246 | 224 |
| | **2** | 109 | 205 | 244 | 298 | 425 | 977 | 659 | 882 | 709 | 784 | | **12** | 620 | 710 | 365 | 634 | 408 | 759 | 238 | 831 | 876 | 239 |
| | **3** | 239 | 971 | 649 | 479 | 949 | 164 | 720 | 964 | 164 | 570 | | **13** | 771 | 273 | 867 | 570 | 449 | 494 | 486 | 544 | 208 | 243 |
| | **4** | 489 | 432 | 749 | 859 | 210 | 705 | 304 | 134 | 300 | 703 | | **14** | 644 | 632 | 379 | 674 | 295 | 398 | 114 | 299 | 773 | 682 |
| | **5** | 938 | 726 | 144 | 565 | 611 | 130 | 593 | 836 | 924 | 411 | | **15** | 307 | 135 | 681 | 469 | 728 | 114 | 847 | 138 | 739 | 220 |
| | **6** | 475 | 968 | 873 | 516 | 776 | 454 | 140 | 421 | 591 | 236 | | **16** | 875 | 672 | 634 | 914 | 850 | 311 | 223 | 184 | 572 | 281 |
| | **7** | 800 | 936 | 987 | 214 | 285 | 360 | 986 | 524 | 174 | 220 | | **17** | 374 | 150 | 148 | 112 | 117 | 980 | 635 | 952 | 120 | 614 |
| | **8** | 746 | 749 | 928 | 338 | 992 | 164 | 719 | 111 | 245 | 670 | | **18** | 393 | 319 | 981 | 299 | 715 | 849 | 563 | 491 | 576 | 421 |
| | **9** | 910 | 650 | 346 | 165 | 621 | 777 | 114 | 475 | 557 | 255 | | **19** | 726 | 767 | 357 | 406 | 972 | 281 | 164 | 988 | 872 | 922 |
| | **10** | 505 | 863 | 437 | 782 | 335 | 805 | 239 | 440 | 707 | 232 | | **20** | 653 | 325 | 548 | 326 | 256 | 188 | 484 | 609 | 902 | 383 |

*Note: The table displays lengths for each of the 10 tasks across the 20 different test cases used in evaluating the algorithm performances.*

The makespan of all the algorithms under the test cases (mentioned in Table 3) is illustrated in Figure 3. In the figure, the x-axis represents the test case indexes, while the y-axis represents the makespan values of each case. It can clearly depict that ELOA consistently outperforms the others, with Max-Min closely following its results. This visualization offers key insights into the relative efficiency of these algorithms, assisting in the selection of optimal task-scheduling strategies.

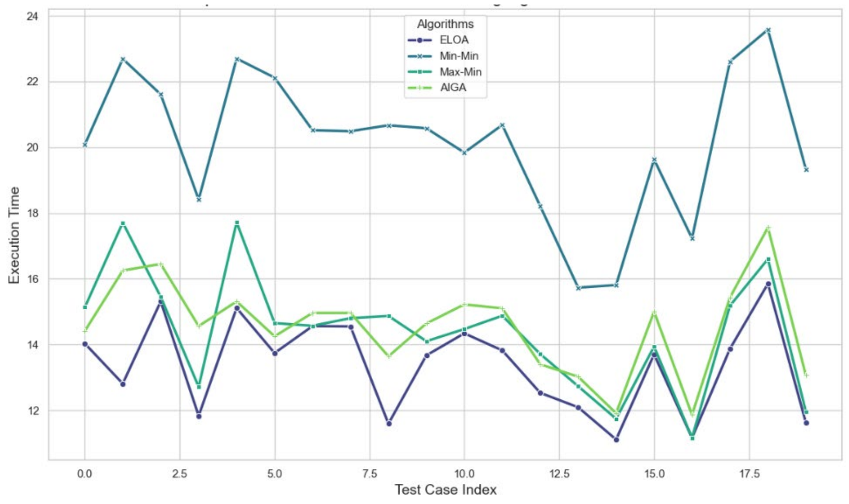


Figure 3. Comparative Performance of Algorithms Across Test Cases

Figure 4 presents a pairwise scatter plot illustrating the performance outcomes of each algorithm across multiple test scenarios. This visualization effectively distinguishes the performance correlations and variations between the algorithms variations across the diverse conditions set forth in the experiments.

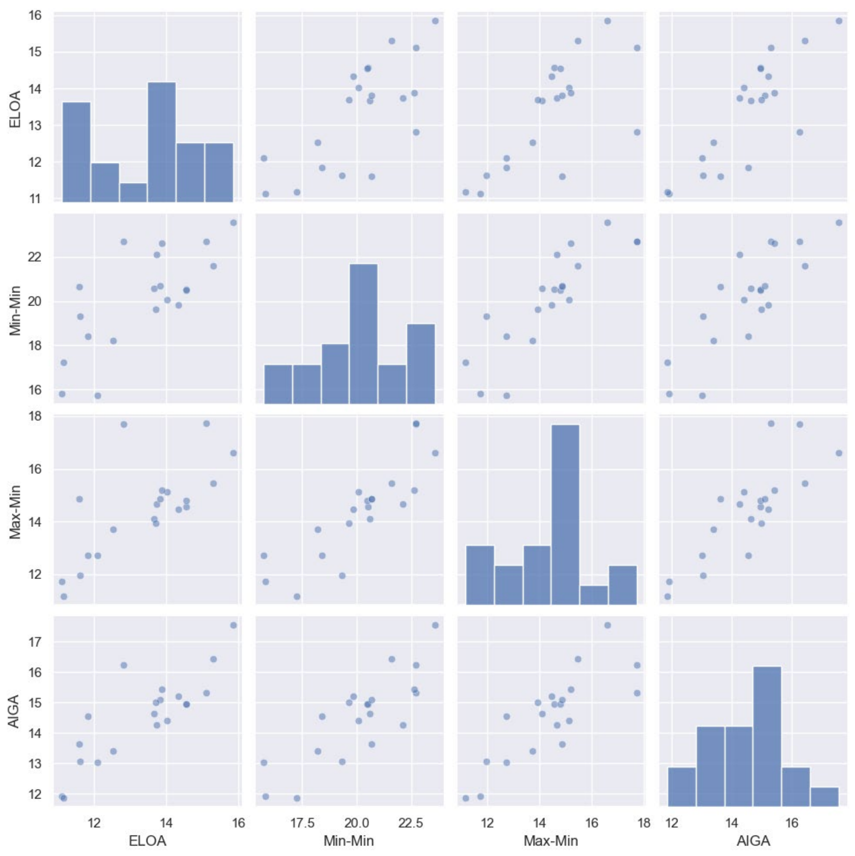


Figure 4. Comparison of Task Scheduling Algorithms

For a further comparison of the difference between the Min-Min, Max-Min, and AIGA algorithms relative to ELOA, a boxplot was created, as represented in Figure 5. This visualization accentuates the outliers, median, and quartiles of performance variations, thereby delineating the spread and central tendency of each dataset. Additionally, it facilitates a comparative analysis that aids in evaluating the robustness and consistency of each algorithm across various computational scenarios. According to boxplot analysis, it can be observed that the difference between the Min-Min Algorithm and ELOA results shows higher variation than the Max-Min Algorithm and AIGA compared to ELOA. Max-Min and AIGA algorithms give closer results to ELOA. However, for interpreting outliers in the boxplot analysis, it can be observed that, despite their heuristic nature, Min-Min does not exhibit significant outliers, but Max-Min and AIGA may show larger deviations from ELOA in some cases.

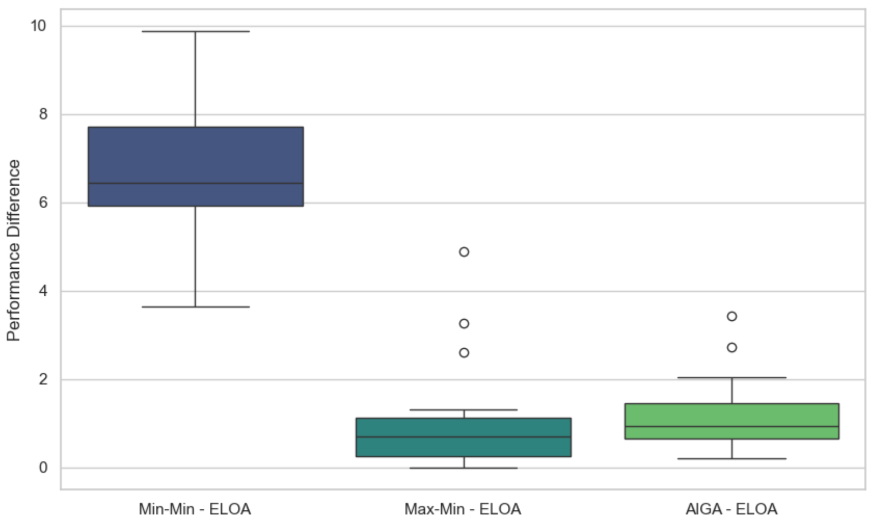


Figure 5. Boxplot Analysis of Makespan Differences Among Algorithms

## 6. Conclusion

To minimize the computational costs and conserve energy, efficient allocation of resources and the minimization of the time required to complete all the tasks, i.e. makespan, is crucial in the realm of cloud task scheduling. Among the strategies employed to address these challenges are the heuristic algorithms Min-Min and Max-Min, as well as the bio-inspired AIGA. While ELOA demands substantial computational power, it excels in smaller projects where sufficient resources are available. In order to identify the most efficient task order while ensuring the best possible outcome, this algorithm takes a constructive approach by thoroughly evaluates all possible task sequence. However, its exhaustive nature makes it computationally intensive. Our findings indicate that while heuristic-based algorithms like Min-Min, Max-Min, and AIGA operate efficiently and help control computing costs, they don't always guarantee the optimal solution. For smaller-scale projects where precision is paramount, ELOA is often the superior choice, as it rigorously evaluates all potential solutions to find the best one. Among the heuristic approaches, AIGA performs closely to ELOA, while the Max-Min algorithm also proves to be a strong alternative. Notably, the makespan difference between Min-Min and ELOA can fluctuate more significantly than that between Max-Min and ELOA, highlighting the varying performance of these strategies across different cloud computing scenarios
Further research could explore hybrid algorithms that merge the efficiency of heuristic methods with the thoroughness of exhaustive approaches. These hybrids may provide a well-balanced trade-off between computational efficiency and solution accuracy, especially for medium- to large-scale applications. Additionally, evaluating these algorithms in edge computing environments—where data is processed closer to its source—could be advantageous, given the growing role of edge computing in reducing latency for cloud services.

## References

Ullah, A., Alomari, Z., Alkhushayni, S., Al-Zaleq, D., Bany Taha, M., and Remmach, H., "Improvement in task allocation for VM and reduction of Makespan in IaaS model for cloud computing," Cluster Computing, vol. 27, no. 8, pp. 11407–11426, 2024. https://doi.org/10.1007/s10586-024-04539-8.

Zhang, J., Cheng, L., Liu, C., Zhao, Z., and Mao, Y., "Cost-aware scheduling systems for real-time workflows in cloud: An approach based on Genetic Algorithm and Deep Reinforcement Learning," Expert Systems with Applications, vol. 234, p. 120972, 2023. https://doi.org/10.1016/J.ESWA.2023.120972.

Naz, I., Naaz, S., Agarwal, P., Alankar, B., Siddiqui, F., and Ali, J., "A Genetic Algorithm-Based Virtual Machine Allocation Policy for Load Balancing Using Actual Asymmetric Workload Traces," Symmetry, vol. 15, no. 5, 2023. https://doi.org/10.3390/sym15051025.

Gulbaz, R., Siddiqui, A. B., Anjum, N., Alotaibi, A. A., Althobaiti, T., and Ramzan, N., "Balancer Genetic Algorithm-A Novel Task Scheduling Optimization Approach in Cloud Computing," 2021. https://doi.org/10.3390/app.

Sahu, D., Sharma, S., Dubey, V., and Tripathi, A., "Cloud Computing in Mobile Applications," International Journal of Scientific and Research Publications, vol. 2, no. 8, 2012.

Gajbhiye, A., and Shrivastva, K. M. P., "Cloud computing: Need, enabling technology, architecture, advantages and challenges," Proceedings of the 5th International Conference on Confluence 2014: The Next Generation Information Technology Summit, pp. 1–7, 2014. https://doi.org/10.1109/CONFLUENCE.2014.6949224.

Radu, L.-D., "Green Cloud Computing: A Literature Survey," Symmetry, vol. 9, no. 12, p. 295, 2017. https://doi.org/10.3390/sym9120295.

Rasheed, H., "Data and infrastructure security auditing in cloud computing environments," International Journal of Information Management, vol. 34, no. 3, pp. 364–368, 2014. https://doi.org/10.1016/J.IJINFOMGT.2013.11.002.

Mell, P., and Grance, T., "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, Special Publication 800-145, September 2011.

Duan, K., Fong, S., Siu, S. W. I., Song, W., and Guan, S. S.-U., "Adaptive Incremental Genetic Algorithm for Task Scheduling in Cloud Environments," Symmetry, vol. 10, no. 5, p. 168, 2018. https://doi.org/10.3390/sym10050168.

Mahmood, A., Khan, S. A., and Bahlool, R. A., "Hard real-time task scheduling in cloud computing using an adaptive genetic algorithm," Computers, vol. 6, no. 2, 2017. https://doi.org/10.3390/computers6020015.

Liu, S., and Wang, N., "Collaborative Optimization Scheduling of Cloud Service Resources Based on Improved Genetic Algorithm," IEEE Access, vol. 8, pp. 150878–150890, 2020. https://doi.org/10.1109/ACCESS.2020.3016762.

Ebadifard, F., and Babamir, S. M., "A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment," Concurrency and Computation: Practice and Experience, vol. 30, no. 12, 2018. https://doi.org/10.1002/cpe.4368.

Laszewski, G. Y., "QoS Guided Min-Min Heuristic for Grid Task Scheduling," Journal of Computer Science and Technology, vol. 1, no. 4, 2003.

Mao, Y., Chen, X., and Li, X., "Max–Min Task Scheduling Algorithm for Load Balance in Cloud Computing," in Proceedings of the 2014 International Conference, pp. 457–465, 2014. https://doi.org/10.1007/978-81-322-1759-6_53.

Braun, T. D., Siegel, H. J., Beck, N., Bölöni, L. L., Maheswaran, M., Reuther, A. I., Robertson, J. P., Theys, M. D., Yao, B., Hensgen, D., and Freund, R. F., "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems," Journal of Parallel and Distributed Computing, vol. 61, no. 6, pp. 810–837, 2001. https://doi.org/10.1006/JPDC.2000.1714.

Wen, J., Lu, L., Casale, G., and Smirni, E., "Less Can Be More: Micro-managing VMs in Amazon EC2," in Proceedings of the 2015 IEEE 8th International Conference on Cloud Computing, pp. 317–324, 2015. https://doi.org/10.1109/CLOUD.2015.50.

Aref, I. S., Kadum, J., and Kadum, A., "Optimization of Max-Min and Min-Min Task Scheduling Algorithms Using G.A in Cloud Computing," in Proceedings of the 2022 5th International Conference on Engineering Technology and Its Applications (IICETA), pp. 238–242, 2022. https://doi.org/10.1109/IICETA54559.2022.9888542.

Hosseini Shirvani, M., "A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems," Engineering Applications of Artificial Intelligence, vol. 90, p. 103501, 2020. https://doi.org/10.1016/J.ENGAPPAI.2020.103501.

Kokash, N., "An Introduction to Heuristic Algorithms," Research Methodology Course, University of Trento, Italy, 2005.

## Biography

**Kıvılcım Naz Böke** is a bachelor's student in Industrial Engineering at Çankaya University, with a minor in Artificial Intelligence. She has gained practical experience through internships in production planning, process design, and ERP systems at various companies, including Kilim Mobilya, Serdar Plastik, and Turkish Aerospace. As the founder of the Sustainable Development Community, she is actively involved in sustainability initiatives. Her research focuses on optimizing task scheduling in cloud computing to improve resource utilization and efficiency.

**Syed Shah Sultan Mohiuddin Qadri** is an Assistant Professor in the Department of Industrial Engineering at Çankaya University, Ankara, Türkiye. He holds a Ph.D. in Industrial Engineering from Yasar University, Izmir, Türkiye. Dr. Qadri earned his BS and MS degrees in Applied Mathematics from the University of Karachi and NED

University of Engineering & Technology, Karachi, Pakistan, in 2010 and 2013, respectively. He has worked as a researcher on a TÜBİTAK-funded project, and his research interests include simulation optimization, intelligent transportation systems, heuristic optimization, and traffic modeling. Dr. Qadri is dedicated to advancing optimization techniques for solving complex real-world problems.

**Ahmet Kabarcık** is a Lecturer at the Department of Industrial Engineering at Çankaya University in Ankara, Turkey. He earned his PhD in Operations Research from the Turkish Military Academy in 2013. He completed his Bachelor's and Master's degrees in Computer Engineering at Çankaya University in 2002 and 2007, respectively. His research interests include network optimization and security, optimal database design and management, enterprise resource planning, object-oriented programming, and computer-aided design and manufacturing. Recently, he has developed a strong interest in artificial intelligence and machine learning and is actively working to enhance his knowledge in these fields.