www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 16(4), Sep-Dec 2025, 399-415. ISSN: 2007-1558. https://doi.org/10.61467/2007.1558.2025.v16i4.983

Experimental Study of Optimization Algorithms for Resource Allocation in Cloud Computing

Elizabeth D. Franco[1], *Claudia G. Gómez Santillán*[1], *Laura Cruz Reyes*[1], *Nelson Rangel Valdez*[1], *Gilberto Rivera Zarate*[2]
[1] TecNM Instituto Tecnológico de Ciudad Madero, México
[2] Universidad Autónoma de Ciudad Juárez, México
elizabeth.franco@outlook.es;{claudia.gs;laura.cr;nelson.rv }@cdmadero.tecnm.mx; gilberto.rivera@uacj.mx

Abstract. Cloud computing is essential for executing scientific workflows, offering scalable resources to process large volumes of data. However, efficient resource allocation remains challenging due to fluctuating demand and inadequate planning, which can increase makespan and, consequently, energy consumption. In this context, Infrastructure as a Service (IaaS) is primarily used, enabling the rental of virtual machines (VMs) with different characteristics. This study identifies the factors that influence makespan improvement when executing workflows from diverse disciplines and with varying characteristics. The factors evaluated are workflow size, workflow structure, and the number of virtual machines required. The algorithms considered include four heuristics and one metaheuristic, specifically a genetic algorithm (GA). Among the factors, the most influential in cloud computing is the number of VMs: increasing the number of VMs reduces makespan up to a point, exhibiting the law of diminishing returns.

Keywords: Cloud computing, scientific workflows, study of factors, makespan

Article Info
Received February 25, 2025
Accepted July 6, 2025

1 Introduction

Cloud Computing (CC) has established itself as a fundamental pillar supporting various work and social domains worldwide. Among these domains, science has particularly benefited from the management of scientific workflows in CC, offering scalable and flexible resources (Alam et al., 2024).

A workflow (WF) can be defined as a set of activities that must be executed in a specific order to achieve a given objective. Scientific WF, in turn, are used to solve complex problems, process large volumes of data, conduct simulations, and perform intensive analyses, all with the aim of generating knowledge across various scientific disciplines or areas (Escott et al., 2022).

In the context of implementing scientific WF in the cloud, Infrastructure as a Service (IaaS) is commonly employed. Under this model, those responsible for executing scientific WF can rent virtual machines (VM) of different types and costs, depending on their computational resource characteristics, such as RAM, storage, GPU, and bandwidth. These machines enable WF execution but also present significant challenges (Kamanga et al., 2023; Li et al., 2024; X. Zhang, 2024)

One of the main challenges lies in the fluctuations in resource demand required to execute these WF. Such behavior can lead to resource shortages, forcing machines to operate for prolonged periods and creating the need to expand data centers. This expansion results in increased energy consumption, raising significant environmental and economic concerns (Al-Wesabi et al., 2022; Thakur & Goraya, 2022). Consequently, current research focuses on optimizing WF management to improve energy efficiency and ensure a more effective allocation of available resources.

These challenges highlight the need for efficient strategies to manage resource allocation in CC, particularly in the context of scientific WF. In this regard, makespan optimization emerges as a key objective, as it maximizes the utilization of available resources and minimizes execution time (Mikram et al., 2024; H. Zhang & Zheng, 2023). Therefore, although much of the literature has focused on comparing optimization strategies, this study is based on the premise that the performance of resource allocation in cloud environments does not depend solely on the chosen algorithm but is deeply conditioned by structural factors of the problem, such as the topology of the Directed Acyclic Graphs (DAG), as well as by the characteristics of the computational environment, including the number and type of available virtual machines. Recognizing and integrating these elements into the experimental design is essential for obtaining robust conclusions and truly efficient solutions.

In other words, the main contribution of this study lies in identifying the factors that directly influence performance, specifically the makespan, during the execution of WF with different topologies. The factors analyzed include the size of the WF, its internal structure or topology, and the number of VM required for execution. To achieve this, five algorithmic strategies were employed four heuristics widely recognized in the literature and a modified genetic algorithm (GA) specifically designed to address this problem. The goal is to establish a solid starting point that will enable future research to develop even more efficient solutions for optimizing scientific WF in CC environments.

2 Fundamental Concepts

As previously mentioned, this document analyzes five types of scientific WF corresponding to different disciplines. Each of these WF presents varying topologies according to their representation in DAGs, which can be observed in Fig. 1 to Fig. 5.

2.1 Scientific workflows

Cybershake, shown in Fig. 1 is a scientific workflow developed by the Southern California Earthquake Center (SCEC). Its main objective is to simulate and analyze seismic activity in specific regions by modeling how seismic waves propagate through different types of geological terrain (Bezdan et al., 2021; Bothra et al., 2021; Konjaang & Xu, 2021; Nasonov et al., 2015; Wang et al., 2020; Xie et al., 2024).

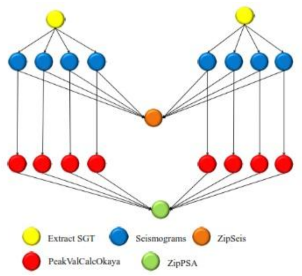

Fig. 1. Scientific WF for the Cybershake application.

Epigenomics, the graphical representation of the Epigenomics workflow is shown in Fig. 2. This benchmark is part of the projects conducted by the University of Southern California (USC). Originating in the field of bioinformatics, this scientific WF aims to analyze epigenomic data, focusing on genetic changes that do not involve modifications to the DNA sequence. These changes are considered essential for understanding cellular regulation and functions under different conditions (Wangsom et al., 2019; Xie et al., 2024; Zeedan et al., 2023).

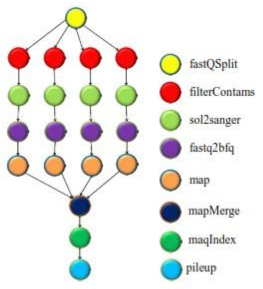

Fig. 2. Scientific WF for the Epigenomics application.

LIGO Inspiral, shown in Fig. 3, is a widely used workflow benchmark derived from the Laser Interferometer Gravitational-Wave Observatory (LIGO) application. It focuses on the detection of gravitational waves—particularly those produced by the inspiral collision of black holes or neutron stars—and enables complex calculations and predictions rooted in general relativity and astrophysics (Nasonov et al., 2017; Rodriguez & Buyya, 2014; Wang et al., 2020; Wangsom et al., 2019; Xie et al., 2024; Zeedan et al., 2023).

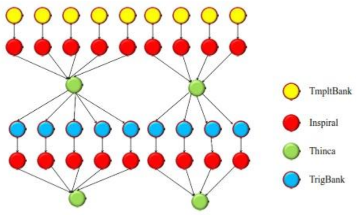

Fig. 3. Scientific WF for the LIGO Inspiral application.

Montage, the WF Montage shown in Fig. 4 is an astronomical application that enables the generation of image mosaics from astronomical observations covering large regions of the sky by combining data from multiple telescopes or observations. This process allows for the study of large-scale structures in the universe (Nasonov et al., 2017; Rodriguez & Buyya, 2014; Wang et al., 2020; Wangsom et al., 2019; Xie et al., 2024; Zeedan et al., 2023).

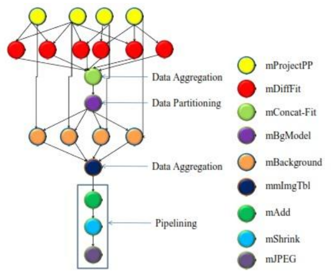

Fig. 4. Scientific WF for the Montage application.

Sipth, This WF in Fig. 5 originates from the SIPHT (Searching for Intergenic sRNA Pathways and Transcriptional Terminators) application, which aims to identify genes that encode small non-coding RNAs (sRNAs) in bacterial genomes (Rodriguez & Buyya, 2014; Wang et al., 2020; Wangsom et al., 2019; Xie et al., 2024; Zeedan et al., 2023).

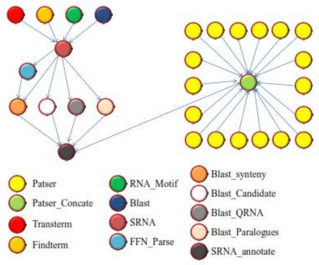

Fig. 5. Scientific WF for the Sipht application.

It is worth noting that the selection of the five presented DAGs (Cybershake, Epigenomics, LIGO Inspiral, Montage, and Sipht) is not arbitrary; rather, it reflects their broad acceptance as representative benchmarks in the scientific WF literature (Nasonov et al., 2017; Rodriguez & Buyya, 2014; Wang et al., 2020; Wangsom et al., 2019; Xie et al., 2024; Zeedan et al., 2023). Each one covers a distinct domain seismic simulations, bioinformatics analysis, gravitational astrophysics, astronomical image processing, and bacterial genomics. These DAGs allow for the evaluation of algorithms under diverse graph structures and computational characteristics, including variations in parallelism, task sizes, internal dependencies, and levels of complexity. Detailing their elements is essential because they reflect different scenarios of cloud resource allocation, ensuring that the conclusions drawn are not limited to a single domain but are generalizable to high-demand computational problems across multiple disciplines.

2.2 Cloud Simulator

A widely used development framework in the literature as a CC simulator is WorkflowSim 1.0. Developed by the University of Southern California and based on CloudSim, it adds WF management features such as task clustering, as well as generation and real-time failure monitoring according to Bambrik (2020), it enables the study of WF scheduling (Andreoli et al., 2024; Chen & Deelman, 2012; X. Zhang, 2024). Due to its ability to model and simulate algorithms in CC, it can be characterized as an Infrastructure as a Service (IaaS) platform through the creation of VM (Konjaang & Xu, 2021).

2.3 Evaluated Algorithmic Strategies from the Literature

The experimental study presented in this document was conducted using four heuristics widely cited in the literature Min-Min, Max-Min, First-Come, First-Served (FCFS), and Round Robin. These heuristics are available within the WorkflowSim 1.0 development framework (Chen & Deelman, 2012; *WorkflowSim/WorkflowSim-1.0*, 2013/2025). Additionally, a genetic algorithm (GA) was evaluated, based on approaches found in the literature and specifically adapted for the makespan optimization problem in scientific WF. The following sections provide a detailed description of the five algorithms mentioned.

Min-Min Algorithm for Scientific WF Optimization. As mentioned at the beginning of this section, the heuristics are part of the example developments within the WorkflowSim 1.0 framework. The implementation and interpretation of the Min-Min algorithm are presented in the algorithm.

Algorithm 1. Min-Min (*WorkflowSim/WorkflowSim-1.0*, 2013/2025).

```
Input: List of cloudlets, List of VM
Output: List of scheduled cloudlets, set VM state to busy for VM executing tasks.
1     Initialize hasChecked as an empty list.
2     Get the size of the cloudlet list
3     Clear the hasChecked list.
4     For each i = 0 to the size of the cloudlet list:
5         Set minIndex = 0 and minCloudlet = null.
6         For each j = 0 to the size of the cloudlet list:
7             If hasChecked[j] == false,
8                 cloudlet [j] = minCloudlet
9                 j = minIndex
10                Break
11        If minCloudlet = = null,
12            Break
13        For each j = 0 to the size of the cloudlet list:
14            If hasChecked[j] == true, continue.
15            Get the length of cloudlet[j].
16            If cloudlet[j].length < minCloudlet:
17                cloudlet [j] = minCloudlet
18                j = minIndex
19        hasChecked[minIndex] = true
20    Get the size of the VM list.
21      Initialize firstIdleVm = null.
22      For each j = 0 to the size of the VM list:
23        If la VM [j] = = idle
24            VM j = firstIdleVm
25            Break
26      IF firstIdleVm = = null
27          Break
28      For each j = 0 to the size of the VM list:
29        If VM[j] == idle && MIPS > firstIdleVm
30          VM[j] = firstIdleVm
31    Set firstIdleVm state to busy.
32    Assign the VmId of firstIdleVm to minCloudlet.
33    Add minCloudlet to the list of scheduled tasks.
```

.

In Algorithm 1, lines 1 to 3 initialize an empty list called hasChecked, as well as obtain the number of jobs. Meanwhile, lines 4 to 19 identify the cloudlet with the smallest length that has not yet been scheduled; this can be considered the core process of the algorithm. It is important to highlight that the essence of the Min-Min algorithm lies in iteratively selecting the shortest task and assigning it to the most suitable available VM, aiming to optimize the overall completion time. Lines 22 to 25 identify idle VM, followed by the selection of the best available VM among all idle VM, which occurs in lines 28 to 30. Finally, in lines 31 to 33, the selected cloudlet is assigned to the chosen VM, its state is updated to busy, and the assignment is restricted within the scheduled task list

Max-Min Algorithm for Scientific WF Optimization. The interpretation of the Max-Min algorithm implementation is presented in Algorithm 2.

Algorithm 2. Max-Min ((*WorkflowSim/WorkflowSim-1.0*, 2013/2025).

```
Input: Task List (CloudletList), Virtual Machines List (VmList)
Output: Scheduled Task List (ScheduledList), Task assignment to VM.
1     Initialize the hasChecked list as an empty list.
2     Obtain the size of the task list (CloudletList).
3     Clear hasChecked.
4     For each task in the task list:
5         hasChecked = false
6     Repeat for each unprocessed task in CloudletList:
7         maxÍndice =0
8         maxTarea = null.
9         For each task in CloudletList:
10            If the task has not been marked as hasChecked
11                maxCloudlet = cloudlet
12                maxIndex = j
13                Break
14        End For
15        If maxCloudlet = = null
16            Break
17        For each remaining task in CloudletList
18            If cloudlet.length > maxTarea.length
19                maxCloudlet = cloudlet
20                maxIndex = j
21        End For
22        hasChecked.maxIndex = true.
23        Obtaining the size of the VM list
24        Initialize firstIdleVm = null
25        For each j = 0 to the size of the VM list:
26            If  VM [j] = = idle
27                 VM[j] = firstIdleVm
28                Break
29         If firstIdleVm = = null
30            Break
31         For each j = 0 to the size of the VM list:
32            If VM [j] == idle && y MIPS > firstIdleVm
33               VM[j] = firstIdleVm
34        Set the state of firstIdleVm to busy.
35        Assign the VmId of the firstIdleVm to the maxCloudlet.
36        Add maxCloudlet to the scheduled task list
37    End Repeat
```

The core of Algorithm 2 (Max-Min) is in lines 4 to 21, where it identifies and selects the longest unprocessed task for execution, prioritizing the distribution of larger tasks to available resources. From lines 6-14, the algorithm identifies the first unprocessed task in the CloudletList, setting it as an initial point for determining the task with the longest length.

In lines 17 to 21, the CloudletList is traversed again to check if the current task has not been processed (i.e., hasChecked[j] == false) and if its length is greater than the length of the task stored in maxTask. In the following lines (22 to 37), the task selected is assigned to the best available VM, and its state is updated to busy, completing the scheduling of tasks.

FCFS for Scientific Workflow Optimization. The defining characteristic of the FCFS algorithm is that it processes tasks strictly in the order they arrive in the queue, ensuring a first-come, first-served approach without considering task size or complexity. That is, the first task to arrive is the first to be executed, the second to arrive is the second to be executed, and so on. This is reflected in Algorithm 3, adapted to assign tasks to available resources as follows.

Algorithm 3. FCFS ((*WorkflowSim/WorkflowSim-1.0*, 2013/2025).

```
Input: List of Cloudlets (CloudletList), List of Virtual Machines (VmList).
Output: List of Scheduled Cloudlets (ScheduledList).
1     For each Cloudlet in CloudletList:
2         set stillHasVm = false
3         For each VM in VmList
4             If VM.state = IDLE
5                 set stillHasVm = true
6                 change VM.state to BUSY
7                 Assign Cloudlet.VMId to VM.Id
8                 Add Cloudlet to ScheduledList
9                 Break
11        If stillHasVm = false
12            Break
```

Lines 3 through 10 of Algorithm 3 begin by iterating over each VM in the VmList to find a VM that can run the current job. To do this, it verifies that the VM's state is IDLE; if so, the stillHasVm flag is set to true, indicating that a VM is available, and then the VM's state is changed to BUSY to be reserved for the current job.

Then, the VM's identifier VM.Id is assigned to the cloudlet, linking the virtual machine to the job for execution. The Cloudlet is then added to the scheduled task list ScheduledList, marking it as scheduled. Once the job is assigned to a virtual machine, the algorithm exits the inner loop ParacadaVM, as it is no longer necessary to search for a virtual machine for the current task.

Round Robin Algorithm for Scientific WF Optimization. The Round Robin algorithm is a preemptive scheduling method that treats all processes equally, assigning each a fixed time slice in a cyclic manner to ensure fairness and balance across tasks. This algorithm is presented below (Algorithm 4).

Algorithm 4. Round Robin ((*WorkflowSim/WorkflowSim-1.0*, 2013/2025).

```
Input: List of tasks (cloudletList) and list of virtual machines (vmList).
Output: List of scheduled tasks (scheduledList), where each task is assigned to a virtual machine.
1     Initialize vmIndex = 0.
2     Get cloudletList and its size.
3     size = cloudletList.size()
4     Sort(cloudletList, CloudletListComparator())
5     vmList = getVmList()
6     Sort(vmList,  VmListComparator()).
7     For each cloudlet task in cloudletList, to size:
8         Obtain the current cloudlet task.
9         Initialize firstIdleVm = null.
10        For each VM in vmList:
11            If vm.state = VM_STATUS_IDLE:
12                Assign vm to firstIdleVm.
13                Break
14        If firstIdleVm is null:
15            Break
16        Change firstIdleVm.state to VM_STATUS_BUSY
17        Assign the ID of firstIdleVm to the current cloudlet task
18        Add the cloudlet to scheduledList.
19         vmIndex = (vmIndex + 1) % size of vmList.
```

Lines 7 to 19 of Algorithm 4 perform task assignment to VM using the Round Robin approach. It iterates through the cloudletList, processing each task to assign it to an available virtual machine. For each task, it looks for a VM in an idle state in the vmList. If a free VM is found, it assigns it to the task, changes its state to busy, and binds the task to that machine. If no available VM is found, the task assignment is stopped. The assigned task is added to the scheduled tasks list, and the virtual machine index is updated in a circular manner, maintaining a balanced task assignment cycle.
Genetic Algorithm. Is one of the most popular algorithms in the literature and, in the words of several authors, is considered one of the first stochastic algorithms inspired by Darwinian theory. This algorithm is regarded as suitable for solving optimization problems and, despite its stochastic nature, offers a probability-based approach that reliably approximates the global optimum by maintaining and evolving the best solutions across generations, thereby enhancing future outcomes. As mentioned earlier, the GA is based on Darwinian theory and consists of four important methods, described as follows

Initial Population, in the literature, the initial population is commonly created randomly and is considered a set of solutions, also known as a set of individuals, where each individual is represented by a chromosome. In the words of Coello Coello (2016), this chromosome is traditionally represented as a binary string. Likewise, each position of the chromosome is called a gene, and the value within that position is called an allele. Selection, inspired by natural selection, it is mentioned that the fittest individuals have a greater probability of obtaining food and reproducing. Similarly, in GA, this mechanism is employed to provide a probability of selecting or rejecting an individual to create a new generation in a way that is favorable to the expected goal.

Crossover (Recombination), this mechanism involves combining the alleles of the selected individuals. That is, the crossover method is performed with the individuals obtained from the selection, and the feature taken from nature is the mixing of genes from a pair of individuals to produce a new chromosome, traditionally involving two individuals (offspring solutions). Mutation it is an evolutionary operator that involves altering one or more alleles after the creation of the offspring solutions. The GA and its last three methods, also known as genetic operators, require parameters to control the probability with which certain processes are applied to the solution population. This is known as parameter configuration (Gómez Santillán, 2009).

2.4 Parameter Configuration

In the words of Gómez Santillán (2009), when addressing a specific problem through the implementation of a metaheuristic, it is essential to consider two main stages the representation scheme and the definition of the objective function. These stages allow the problem to be properly structured within the chosen metaheuristic framework and require the selection of appropriate values for the parameters involved, aiming for an optimal representation of the problem and efficient resolution.

Additionally, the combination of the values assigned to the parameters is called the parameter configuration, while the process of identifying and selecting the best possible configuration is known as parameter tuning.Gómez Santillán (2009) explains that there are two types of parameter adaptation. The first is known as parameter tuning, which is set statically before the algorithm's execution and remains constant throughout its operation. The second is called parameter control, which is applied dynamically during the algorithm's execution, adjusting to the needs that arise in the process.

Some authors in the literature, such as Romero-Ocaño *et al.* (2018), mention that parameter fine-tuning can be defined as an offline configuration, while parameter control is considered an online configuration. The classification of parameter tuning is shown in Fig. 6.

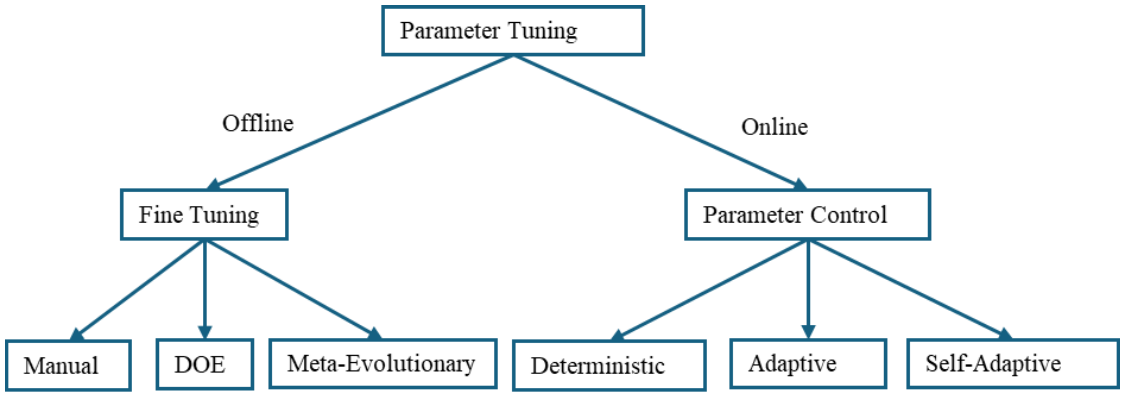

Fig. 6. General parameter adjustment settings (Gómez Santillán, 2009).

As shown in Fig. 6, Gómez Santillán (2009) presents that parameter fine-tuning can be carried out using three different techniques manually, by testing various configurations and selecting the one that offers the best performance; through meta-evolution, using an auxiliary algorithm for optimization; or with Design of Experiments (DOE), applying statistical tests to improve decision-making (Gómez Santillán, 2009).

In general, although some authors in the literature, such as De Jong (2007) and Rosas *et al.* (2024), mention that the no free lunch theorem expresses that there is no optimal set of parameters that can solve all optimization problems, they also emphasize that at least having a good initial configuration is important, especially when addressing real-world problems. This is because algorithms tend to rely on specific configurations.

2.5 Design of Experiments

The design of experimental (DOE) allows for parameter tuning. Additionally, it is described as a highly relevant methodology in the literature, as it encompasses a set of statistical tools used to plan, execute, classify, analyze, and interpret experiments efficiently (Gómez Santillán, 2009; Jankovic et al., 2021; Montgomery, 2017; Wu & Hamada, 2011). Its main objective is to identify the environments and conditions that produce valid or optimal and impartial results. Moreover, (Gómez Santillán, 2009; Jankovic et al., 2021; Montgomery, 2017) mention that the process involves multiple steps, such as

Identification and formulation of the problem this is the first step and allows defining the objective of the experiment, as well as identifying the variables of interest. Selection of factors, levels, and response variables in this step, the factors or independent variables to manipulate are selected, and the specific values (levels) of these factors are determined, while the response variables define how the experiment's result will be measured. Choice of experimental design in this phase, the most appropriate type of design for the experiment is selected, which depends on the experiment's objectives and constraints.

Execution of the experiment as the name suggests, this is the phase where the experiments are carried out based on the factors selected in the previous phases. Analysis of statistical data with the data collected from the previous phase, this step involves analyzing it using statistical tools. Drawing conclusions and recommendations finally, conclusions are drawn from the data analysis, and recommendations are made to optimize the process or system, based on the obtained results.

2.6 Experimental Strategy Factorial Design

One of the experimental designs is the factorial design, which is an approach where two or more factors are studied simultaneously, rather than analyzing one factor at a time. This allows for evaluating not only the individual effects of each factor on the response variable but also their interactions (Gómez Santillán et al., 2009; Montgomery, 2017). According to Montgomery (2017), there are different types of factorial designs, such as full factorial design, fractional factorial design, and 2k factorial design.

The full factorial design tests all possible combinations of factors and levels, providing the maximum amount of information about the main effects and interactions between factors. In other words, according to Romero- Ocaño *et al.* (2018), a full factorial design performs random executions of all possible combinations that can be formed from the levels of the factors under study, allowing for the analysis of both the individual impact of each factor on the response variable and the effects of interactions between factors on that variable (Gómez Santillán et al., 2009). The effect of a factor is defined as the variation in the response variable caused by a change in the level of the factor. This effect, known as the main effect, refers to the direct influence of each primary factor within the experiment (Gómez Santillán et al., 2009).

2.7 Evaluation Metric

In the literature, some authors, such as Alsadie (2021), Ghorbannia Delavar and Aryan (2014), and Zeedan et al. (2022), among others, mention that *makespan* is considered a key metric in optimization problems and is applicable to the case study presented in this experimental research, which focuses on the optimization of scientific WF. This metric reflects the total time required to complete a set of tasks or jobs in a system, ensuring that the available resources in CC are used optimally. In the context of cloud resources and scientific WF, the makespan is expressed as the finishing time of the last task within the workflow execution. Therefore, reducing the makespan directly contributes to improving system performance and enhancing the efficiency of resource utilization. Consequently, it is widely adopted as a benchmark to compare the performance of different scheduling algorithms. A shorter makespan indicates a faster completion of workflows, which is crucial in environments where time-sensitive results are required. Furthermore, reducing this metric contributes not only to efficiency but also to lowering the operational costs associated with cloud infrastructures.

$$Makespan = \underset{i}{max}\left(T_{\{completation,i\}}\right)$$

Donde:
$T_{\{completation,i\}}$: corresponds to the completion time of the last task on resource i

3 Solution Methodology

As mentioned in the introduction, the issue of fluctuations in the demand for resources required for the execution of these WF in a CC environment is a broad and significant topic of interest, as it directly impacts the fulfillment of service levels and other aspects. Additionally, this problem is considered NP-Hard (Farid et al., 2023; Xu & Abnoosian, 2022; X. Zhang, 2024).

For this reason, we decided to conduct an experimental study that not only includes the heuristics presented in subsection 2.3 but also incorporates a metaheuristic, in this case, a genetic algorithm (GA). This choice is based on two main reasons methodological consistency the selected heuristics are already implemented and validated within WorkflowSim 1.0, ensuring uniformity in the experimental conditions.

Complementarity of approaches while heuristics allow for fast and efficient decision-making, their main limitation is the tendency to become trapped in local optima. The GA, widely studied in the literature, offers an evolutionary approach that enriches the analysis by contrasting it with classical heuristics.

We acknowledge that a more balanced set (for example, three heuristics and three metaheuristics) could provide a more comprehensive comparison. However, this initial setup aims to establish a solid baseline for future research. In subsequent work, we plan to include other evolutionary and bio-inspired algorithms, such as PSO, ACO, and DE, to broaden the study's scope.

Including these five algorithms enables an in-depth evaluation of critical factors such as resource scalability, WF size, or DAG topology. This approach is key to developing more efficient resource allocation strategies, making this study highly relevant in the field. Following the objective of this section, two subsections are presented. The first introduces the GA implemented within the WorkflowSim 1.0 development framework as Algorithm 5, while the second subsection presents the experimental design, which, as mentioned throughout this document, forms an essential part of the present study.

3.1 Genetic Algorithm for Scientific WF Optimization

In this subsection, we present the GA implemented in this work. Although its design is inspired by classical approaches reported in the literature, its implementation was specifically adapted to the problem of workflow scheduling in cloud computing environments. In particular, the modifications focus on the solution representation and the design of evolutionary operators, with the aim of minimizing the makespan, which is the central metric for evaluating system performance.

As part of the details of the GA (Algorithm 5), it begins by retrieving the list of jobs (cloudletList), along with the creation of the VM in the simulation environment. This corresponds to lines 1 and 2. In lines 3 to 7, the initial population is created with random numbers corresponding to the existing VM. In general, lines 8 to 25 carry out the algorithm's evolutionary procedure, which consists of three operators that can be seen as three stages

Selection stage, the selection operator is applied, whereby, based on a defined percentage, a proportion of individuals from the population is chosen to be used as parents for the next generation. The crossover operator is applied, considering the crossover percentage, which determines how many of the selected parent individuals will participate in recombination and which genes will form the chromosomes of the offspring. The mutation operator is applied, whose goal is to maintain diversity within the population. Through a defined probability, it is determined whether a gene within a descendant individual will be altered.

Finally, in line 26, each individual, both offspring and parents, is evaluated, and subsequently, in lines 28 to 35, the selected cloudlet is assigned to the chosen VM, its status is updated to busy, and the assignment is registered in the list of scheduled tasks. For the GA parameters, an offline parameter tuning was performed, with the values presented in Table 1.

Table 1. Configuration GA.

Parameter	Detail
Population Size	100
Selection Operator	Roulette
Crossover Operator	Uniform
Mutation Operator	Reciprocal
Selection Percentage	0.5
Crossover Percentage	0.5
Mutation Percentage	0.05

Algorithm 5. Genetic algorithm.

Input: Cloudlet list, VM list, population size, crossover rate *Pc*, mutation rate *Pm*, number of generations *nGenerations*

Output: List of scheduled cloudlets, VM status set to busy for the VM executing tasks

```
1.  cloudletList []= getCloudletList ()
2.  vmList = getVmList ()
3.  Initialize populationList []. Length
4.  For i = 0 to populacition.lenght()
5.      Para j = 0 hasta cloudletList. lenght ()
6.          Solution [j] = generate random solution (vmList)
7.      Add population[i] = solution;
8.  For generation = 0 to nGeneraciones
9.      initialize fitnessList []
10.     For each individual in the population
11.         fitnessList = calculateFitness (individuo, cloudletList, vmList)
12.     Initialize matingPool
13.     For i = 0 to population.lenght
14.         parent1 = matingPool[i]
15.         parent2= matingPool[i+1]
16.         child1, child2 = crossover (parent1, parent2, pc)
17.         offspringList.add(child1)
18.         offspringList.add(child2)
19.     For child= 0 to offspringList.size()
20.         Mutate (child, Pm, vmList.size())
21.     offspringFitnessList = []
22.     For child = 0 to offspringList.size()
23.         Fitness = evaluateFitness(child,cloudletList,vmList)
24.         offspringFitnessList.add(fitness)
25.     populationList = replacePopulation (populationList, fitnessList, offspringList, offspringFitnessList)
26.     bestIndividual = findBestIndividual(populationList, fitnessList)
27.     scheduledCloudlets = []
28.     For i = 0 to cloudletList.size()-1
29.         cloudlet = cloudletList[i]
30.         vmIndex = bestIndividual[i]
31.         vm = vmList[vmIndex]
32.         assignVM(cloudlet, vm)
33.         vm.state = BUSY
34.         scheduledCloudlets.add(cloudlet)
35.     Return scheduledCloudlets
```

3.2 Experimental Desing

In the context of scientific WF and the research outlined in this document, Table 2 and Table 3 are presented below to detail the characteristics of each DAX, which in this work refers to a Directed Acyclic Graph expressed in XML. These XML-based workflow descriptions, introduced in subsection 2.1, are classified according to their size, serving as the basis to guide the presentation of the experimental results. Furthermore, within the WorkflowSim 1.0 development framework, a total of 20 VM and two data centers were established. The characteristics of these resources are presented in Table 4 and Table 5, respectively.

Table 3. DAX size classification based on number of jobs.

Jobs	Size Classification
24 - 30	Small
46 - 60	Medium
100	Large
997 - 1000	Extra Large

Table 2. Characteristics of DAX.

DAX	No. Jobs	No. Task
CyberShake	30	90
	50	154
	100	312
	1000	3004
Epigenomics	24	69
	46	134
	100	297
	997	2969
LIGO Inspiral	30	163
	50	278
	100	546
	1000	5549
Montage	25	134
	50	290
	100	618
	1000	6472
Sipht	30	1927
	60	3976
	100	6047
	1000	64174

Table 4. Characteristics of each VM.

VM Instance	Number
Image Size	10000 MB
RAM	512 MB
MIPS	1000
Bandwidth	1000
Number of CPUs	1
VMM name	Xen

Table 5. Characteristics of each data center.

DC Instance	Number
Storage	1000000 MB
RAM	2048 MB
MIPS	2000
Bandwidth	10000
Number of PEs per CPU core	20
System Architecture	x86
Operating System	Linux
Hypervisor	Xen

4 Experimentation and Results

This section presents the experimental design as well as the results obtained. The former allows us to describe the elements and characteristics that make up the experimentation, while the latter exposes the findings. Following the guidelines described in section three, the objective is to identify the relevant factors that could enable the design of more efficient strategies for resource allocation.

4.1 Analysis and Results

As the first part of the experimentation, we present a multifactorial analysis using the non-parametric Scheirer-Ray-Hare test (rank-based multifactorial ANOVA), applied to the normalized average makespan data. For this experimentation, the following main factors were established i) DAG Topology referred to as *DAX*; ii) DAG Size referred to in this article as *DAXSize*; iii) Number of VM referred to as *NumVMs*; iv) Algorithm Additionally, significant interactions between the factors were identified. A summary of the results is shown in Table 6.

The results in Table 6 reveal that all algorithms show highly significant effects from the main factors, with p-values < 0.0001. This indicates that the type and size of the instance, as well as the number of VM, strongly influence algorithm performance. The DAX × DAXSize interactions are consistent and statistically significant across all algorithms, suggesting that the combination of problem characteristics affects how the methods behave. The DAX × Number of VM interactions are significant only for Min-Min, Max-Min, and FCFS, but not for Round Robin or GA, possibly reflecting differences in how these algorithms exploit virtualized resources. The DAXSize × Number of VM interactions are significant for Max-Min, FCFS, Round Robin, and GA, but marginal for Min-Min, pointing to possible variations in scalability depending on the algorithm.

Table 2. Summary of non-parametric multifactorial analysis results

Effect / Algorithm	Min-Min	Max-Min	FCFS	RoundRobin	GA
C(DAX)	p ≈ 2.7e-28	p ≈ 3.3e-32	p ≈ 1.7e-28	p ≈ 1.4e-27	p ≈ 1.4e-26
C(DAXSize)	p ≈ 2.5e-32	p ≈ 3.5e-36	p ≈ 9.9e-33	p ≈ 3.4e-31	p ≈ 2.1e-32
C(NumVMs)	p ≈ 1.3e-08	p ≈ 1.2e-14	p ≈ 7.8e-10	p ≈ 6.1e-08	p ≈ 3.2e-13
DAX × DAXSize	p ≈ 1.1e-09	p ≈ 3.9e-11	p ≈ 2.1e-09	p ≈ 1.1e-08	p ≈ 1.3e-10
DAX × NumVMs	p ≈ 0.025	p ≈ 0.0021	p ≈ 0.0165	p ≈ 0.359	p ≈ 0.095
DAXSize × NumVMs	p ≈ 0.095	p ≈ 1.0e-05	p ≈ 0.0126	p ≈ 0.0064	p ≈ 0.0039

These findings highlight that not only individual factors but also their combinations differentially impact algorithmic performance. The analysis suggests that adjusting environment parameters (such as the number of VM) must be carefully considered, along with both the nature of the instance and the algorithm used. As the second part of the analysis, to detail the specific levels within each factor responsible for the differences, Dunn's post-hoc tests were applied, adjusted with Holm's correction for multiple comparisons. The results are shown in Table 7.

Table 3. Summary of post-hoc test results

Factor	Pairs with p < 0.05	Min-Min p-value	Max-Min p-value	FCFS p-value	RoundRobin p-value	GA p-value
DAX	Epigenomics–CyberShake	0.009	0.0038	0.0202	0.0079	0.0266
	Epigenomics–Montage	0.000025	0.000016	0.000016	0.000016	0.00023
	Montage–Sipht	0.00091	0.0027	0.00088	0.00068	0.00396
	LIGO–Montage	0.0495	0.0495	0.0607	0.053	0.0367
DAXSize	Extra Large–Large	0.00077	0.00029	0.00054	0.00047	0.00022
	Extra Large–Medium	0.000002	0.000001	0.000002	0.000002	4.40E-07
	Extra Large–Small	1.90E-09	3.70E-09	1.70E-09	3.50E-09	1.30E-09
	Large–Small	0.0315	0.0795	0.0382	0.0577	0.0621
NumVMs	None (all p = 1.0)	1	1	1	1	0.56–1.0

As shown in Table 7, the results indicate for *DAX*, the most differentiated pairs were mainly between Epigenomics and Montage, Epigenomics and CyberShake, and Montage versus Sipht; For *DAXSize,* the *Extra Large* group showed statistically significant differences against all other sizes, highlighting its role as an extreme case; For *NumVMs*, none of the algorithms showed significant comparisons between levels (all p-values ≥ 0.55), reinforcing that this factor was not decisive in the studied scenarios. To conclude the experimentation analysis, we also examined the main effects plots and interaction plots.

Fig. 7 presents the main effects plots per algorithm, where the Y-axis represents the normalized average makespan using the Min-Max scaler strategy.In the first section, corresponding to DAX, we observe that the Epigenomics DAG exhibits a higher average makespan compared to the other scientific workflows, suggesting that Epigenomics has greater resolution complexity. This behavior is consistent across all evaluated algorithms.

On the other hand, CyberShake and Montage show lower makespan values, indicating less complex applications, possibly due to shorter tasks or fewer interdependencies. For *DAXSize*, workflows classified as Extra Large have significantly higher execution times (makespan) compared to large, medium, and small sizes. This is expected given the greater number of tasks and dependencies in larger graphs, which increases both complexity and completion time.

Finally, regarding the number of VM (NumVMs), we observe that as the number of VM increases, the average makespan decreases. However, this reduction becomes less pronounced beyond ten virtual machines, suggesting that the scalability benefit has a limit. This could be compared to the law of diminishing returns (Brue, 1993). This law states that, beyond a certain point, adding more resources to a system does not produce proportional benefits. In our experiments, this saturation occurs around 10 VM, after which the benefits in makespan reduction stabilize. This phenomenon is due to the fact that additional resources do not eliminate the intrinsic complexity of the task graph or the critical dependencies between jobs. Additionally, the interaction plots were analyzed, as they allow us to understand how two or more factors jointly influence a response variable.

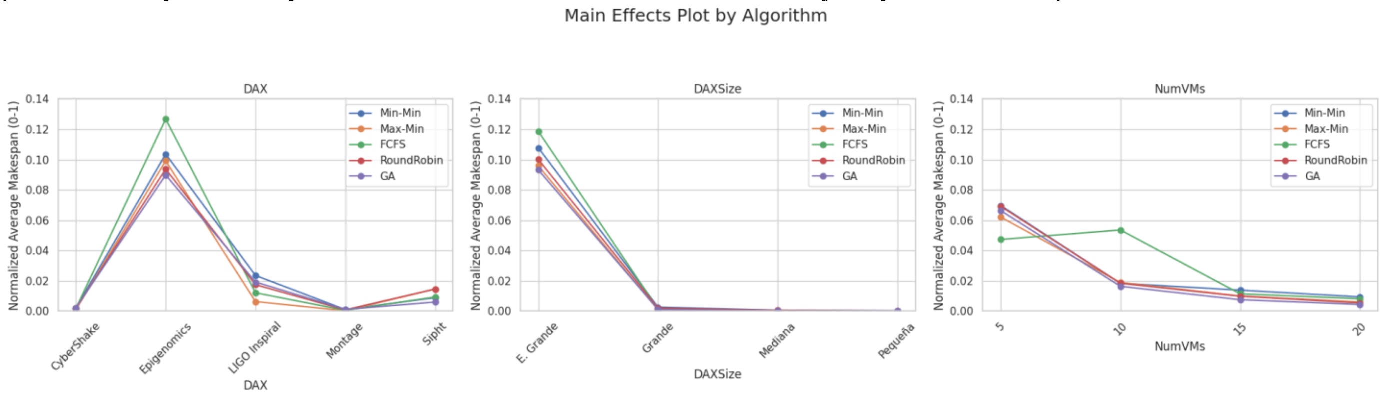

Fig. 7. Main effects plot by algorithm

In Fig. 8, the interaction between DAX type and size (DAXSize) is evaluated. We observe that for larger DAXsizes (Extra Large), the makespan increases significantly. Furthermore, the Epigenomics DAX consistently shows a higher makespan compared to other DAX, even at the same size. This reinforces the results seen in the main effects plots, where both

Epigenomics and larger DAXsizes already displayed higher makespan values. The interaction between DAXtype and size indicates that both factors jointly influence the makespan.

In the second set of plots, the interaction between the number of VM and DAXtype is assessed. We observe that Epigenomics and LIGO Inspiral show improvements in average makespan as the number of virtual machines increases, but this improvement plateaus after ten virtual machines. This suggests that although increasing the number of virtual machines benefits all DAX, there is no strong interaction between DAX type and the number of VM. That is, the number of virtual machines has an effect independent of DAX type.

The interaction between workflow size and number of virtual machines reveals that for larger DAXsizes (Extra Large), the makespan is significantly higher. However, the impact of increasing the number of virtual machines is greatest when moving from five to ten machines. Beyond ten virtual machines, the makespan stabilizes, suggesting that adding more resources does not provide meaningful improvements. This behavior exemplifies the law of diminishing returns, where scalability benefits have a limit. For smaller DAXsizes, the impact of adding more virtual machines is less significant, as the tasks are already well distributed.

Finally, when evaluating all the interactions presented in Fig. 8, we can note that there is no significant synergistic relationship between the factors. In other words, although each factor (DAX type, DAX size, and number of virtual machines) individually influences the makespan, their combined effects do not generate strong interactions beyond what is already explained by the main effects.

5 Conclusions

As described throughout this document, the presented study focused on evaluating the factors that affect the performance of five optimization algorithms for resource allocation in CC using scientific WF from various disciplines. The main conclusions are as follows Impact of *WFSize* WF classified as *Extra Large* exhibit significantly higher execution times, regardless of the algorithm used. This highlights the need for more advanced strategies to handle the complexity of these cases.

Algorithm Performance the Min-Min and Max-Min algorithms showed consistent performance in minimizing makespan compared to other approaches, making them ideal for workflows with tasks of variable length. The Round Robin algorithm demonstrated balanced performance but was less efficient in scenarios with more complex tasks. The GA stood out for its adaptability and superior performance in large and complex workflows, although at the cost of higher computational time.

However, one notable takeaway is that when designing more efficient strategies, it is recommended to employ heuristics for small-sized DAGs due to their speed and simplicity. However, since large DAGs pose greater challenges, it becomes necessary to turn to more advanced metaheuristics. Additionally, a potentially interesting alternative Additionally, a potentially interesting alternative is the implementation of parallel algorithms, as the structure of the Epigenomics DAG tends to be the most complex to solve. We believe this is due to its high degree of parallelism, represented in its graph, which could be exploited to optimize workflow execution in CC environments.

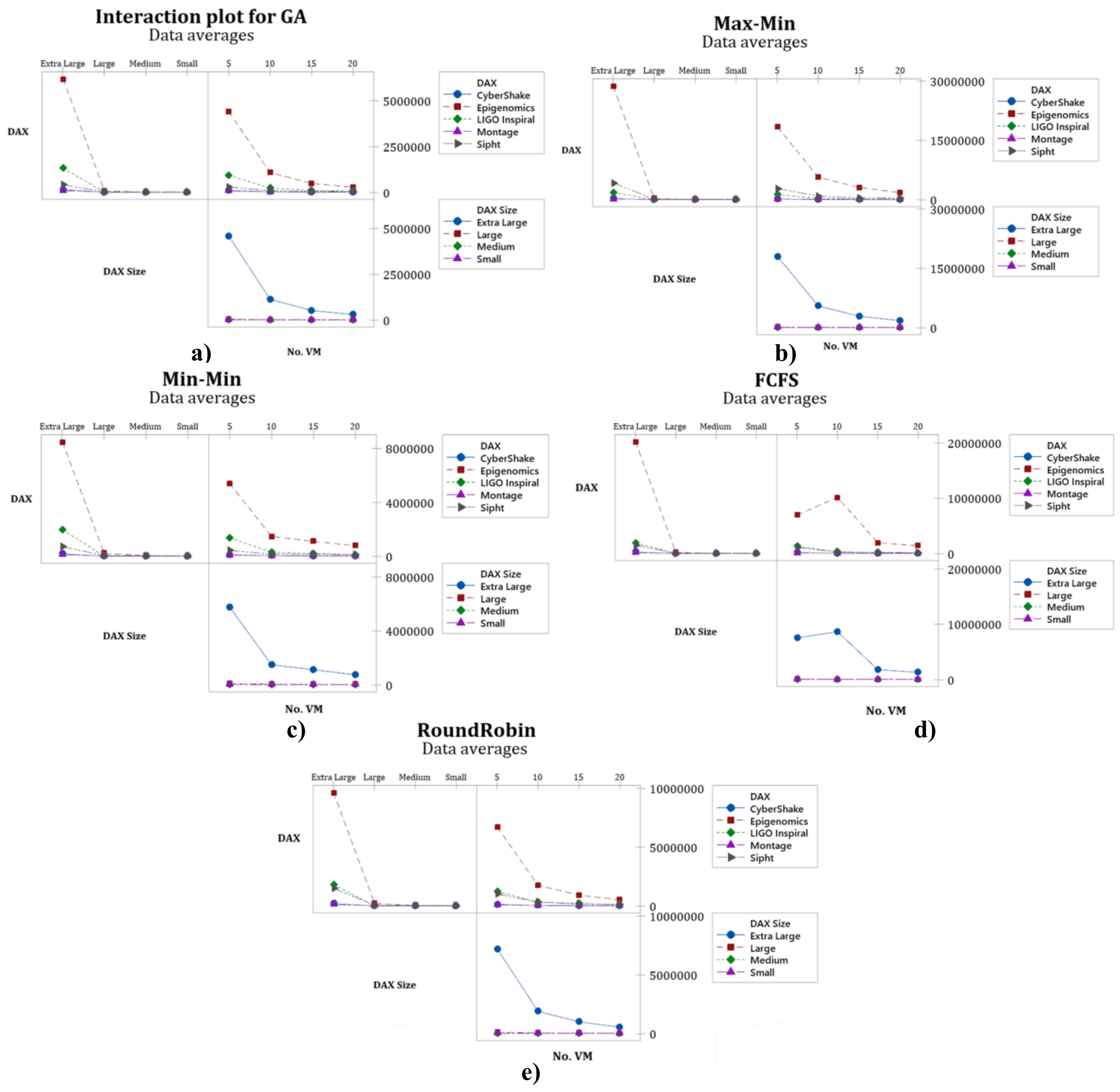

Fig. 8. Interaction plot by algorithm. a) Interaction plot for GA, b). Interaction plot for Max-Min; c) Interaction plot for Min-Min; d) Interaction plot for FCFS; e) Interaction plot for RoundRobin

Effect of the Number of VM: Increasing the number of VM significantly reduces the makespan up to a certain point, stabilizing around ten machines. This suggests that more efficient resource allocation could avoid unnecessary overprovisioning. Specific Workflows the Epigenomics workflow consistently showed the highest makespan values, underscoring the need for customized or adaptive algorithms to address its particular characteristics. Based on the above observations, we can emphasize that the importance of the factors influencing this kind of experimentation lies not only in the algorithm itself, but also in the fundamental characteristics of the problem, such as the topology of the DAG and the computing infrastructure, including the number and type of VM available. Ignoring these structural and contextual factors can significantly limit the potential improvements that any algorithm, no matter how advanced, can offer. Therefore, achieving truly efficient solutions requires a combined approach that balances algorithmic design with a careful understanding of workflow structure and cloud environment capabilities.

Furthermore, we propose expanding the analysis toward coevolutionary strategies that could combine different optimization approaches working simultaneously, adapting to the changing characteristics of the workflows. By integrating multiple algorithms that evolve in parallel, efficiency in resource allocation could be improved, especially in complex scenarios. These coevolutionary strategies could be particularly useful for addressing large-scale, highly complex problems, such as those posed by workflows with a high degree of parallelism or more complex dependency structures. Additionally, combining different algorithms could allow a more thorough exploration of the solution space, reducing the likelihood of becoming trapped in local optima.

Finally, regarding the experimental study, additional performance indicators could be incorporated, such as cost, resource usage efficiency, or impact on quality of service. It would be valuable to extend this study by incorporating other DAGs with different characteristics or even expanding the analysis by tuning the parameters in the GA, which could contribute to obtaining more precise and representative results of real-world operating conditions.

Acknowledgments

EDF gratefully acknowledges the CONAHCyT grant for doctoral studies (Grant 930472). The authors are also thankful for the support obtained from: Tecnológico Nacional de México through the Instituto Tecnológico de Ciudad Madero, for the use of the Laboratorio Nacional de Tecnologías de Información (LaNTI), and for the support received through the TecNM project 'Design of an intelligent virtual agent for decision support with impact on the navigation of electric vehicles'.

References

Alam, M., Shahid, M., & Mustajab, S. (2024). Security prioritized multiple workflow allocation model under precedence constraints in cloud computing environment. *Cluster Computing*, *27*(1), 341-376. https://doi.org/10.1007/s10586-022-03819-5

Alsadie, D. (2021). TSMGWO: Optimizing Task Schedule Using Multi-Objectives Grey Wolf Optimizer for Cloud Data Centers. *IEEE Access*, *9*, 37707-37725. https://doi.org/10.1109/ACCESS.2021.3063723

Al-Wesabi, F. N., Obayya, M., Hamza, M. A., Alzahrani, J. S., Gupta, D., & Kumar, S. (2022). Energy Aware Resource Optimization using Unified Metaheuristic Optimization Algorithm Allocation for Cloud Computing Environment. *Sustainable Computing: Informatics and Systems*, *35*, 100686. https://doi.org/10.1016/j.suscom.2022.100686

Andreoli, R., Zhao, J., Cucinotta, T., & Buyya, R. (2024). *CloudSim 7G: An Integrated Toolkit for Modeling and Simulation of Future Generation Cloud Computing Environments* (No. arXiv:2408.13386). arXiv. http://arxiv.org/abs/2408.13386

Bambrik, I. (2020). A Survey on Cloud Computing Simulation and Modeling. *SN Computer Science*, *1*(5), 249. https://doi.org/10.1007/s42979-020-00273-1

Bezdan, T., Zivkovic, M., Antonijevic, M., Zivkovic, T., & Bacanin, N. (2021). Enhanced Flower Pollination Algorithm for Task Scheduling in Cloud Computing Environment. En A. Joshi, M. Khosravy, & N. Gupta (Eds.), *Machine Learning for Predictive Analysis* (pp. 163-171). Springer. https://doi.org/10.1007/978-981-15-7106-0_16

Bothra, S. K., Singhal, S., & Goyal, H. (2021). Deadline-Constrained Cost-Effective Load-Balanced Improved Genetic Algorithm for Workflow Scheduling. *International Journal of Information Technology and Web Engineering (IJITWE)*, *16*(4), 1-34. https://doi.org/10.4018/IJITWE.2021100101

Chen, W., & Deelman, E. (2012). WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. *2012 IEEE 8th International Conference on E-Science*, 1-8. https://doi.org/10.1109/eScience.2012.6404430

Coello Coello, C. A. (2016, mayo). *Introoducción a la computación Evolutiva (Notas del curso)*.

De Jong, K. (2007). (PDF) Parameter Setting in EAs: A 30 Year Perspective. En *Parameter Setting in Evolutionary Algorithms* (Vol. 54, pp. 1-18). Springer Science & Business Media. https://www.researchgate.net/publication/227027773_Parameter_Setting_in_EAs_a_30_Year_Perspective

Escott, K.-R., Ma, H., & Chen, G. (2022). Transfer Learning Assisted GPHH for Dynamic Multi-Workflow Scheduling in Cloud Computing. En G. Long, X. Yu, & S. Wang (Eds.), *AI 2021: Advances in Artificial Intelligence* (pp. 440-451). Springer International Publishing. https://doi.org/10.1007/978-3-030-97546-3_36

Farid, M., Lim, H. S., Lee, C. P., & Latip, R. (2023). Scheduling Scientific Workflow in Multi-Cloud: A Multi-Objective Minimum Weight Optimization Decision-Making Approach. *Symmetry*, *15*(11), Article 11. https://doi.org/10.3390/sym15112047

Ghorbannia Delavar, A., & Aryan, Y. (2014). HSGA: A hybrid heuristic algorithm for workflow scheduling in cloud systems. *Cluster Comput*, *17*(1), 129-137. https://doi.org/10.1007/s10586-013-0275-6

Gómez Santillán, C. G. (2009). *Afinación Estática Global de Redes Complejas y Control Dinámico Local de la Función Tiempo de Vida en el Problema de Direccionamiento de Consultas Semánticas* [Instituto Politécnico Nacional/ Centro de Investigación en Ciencia Aplicada y Tecnología Avanzada, Unidad Altamira]. https://www.dropbox.com/s/o0u770p77clo2hx/TDClaudiaGS10.pdf?dl=1

Gómez Santillán, C. G., Reyes, L. C., Conde, E. M., Martinez, C. A., Lam, M. A. A., & Zezzatti, C. A. O. O. (2009). Performance Analysis of the Neighboring-Ant Search Algorithm through Design of Experiment. En E. Corchado, X. Wu, E.

Oja, Á. Herrero, & B. Baruque (Eds.), *Hybrid Artificial Intelligence Systems* (pp. 662-669). Springer. https://doi.org/10.1007/978-3-642-02319-4_80

Jankovic, A., Chaudhary, G., & Goia, F. (2021). Designing the design of experiments (DOE) – An investigation on the influence of different factorial designs on the characterization of complex systems. *Energy and Buildings*, *250*, 111298. https://doi.org/10.1016/j.enbuild.2021.111298

Kamanga, C. T., Bugingo, E., Badibanga, S. N., & Mukendi, E. M. (2023). A multi-criteria decision making heuristic for workflow scheduling in cloud computing environment. *The Journal of Supercomputing*, *79*(1), 243-264. https://doi.org/10.1007/s11227-022-04677-z

Konjaang, J. K., & Xu, L. (2021). Multi-objective workflow optimization strategy (MOWOS) for cloud computing. *Journal of Cloud Computing*, *10*(1), 11. https://doi.org/10.1186/s13677-020-00219-1

Li, H., Tian, L., Xu, G., Cañizares Abreu, J. R., Huang, S., Chai, S., & Xia, Y. (2024). Co-evolutionary and Elite learning-based bi-objective Poor and Rich Optimization algorithm for scheduling multiple workflows in the cloud. *Future Generation Computer Systems*, *152*, 99-111. https://doi.org/10.1016/j.future.2023.10.015

Mikram, H., El Kafhali, S., & Saadi, Y. (2024). HEPGA: A new effective hybrid algorithm for scientific workflow scheduling in cloud computing environment. *Simulation Modelling Practice and Theory*, *130*, 102864. https://doi.org/10.1016/j.simpat.2023.102864

Montgomery, D. C. (2017). *Design and Analysis of Experiments*. John Wiley & Sons.

Nasonov, D., Melnik, M., & Radice, A. (2017). Coevolutionary Workflow Scheduling in a Dynamic Cloud Environment. En M. Graña, J. M. López-Guede, O. Etxaniz, Á. Herrero, H. Quintián, & E. Corchado (Eds.), *International Joint Conference SOCO'16-CISIS'16-ICEUTE'16* (pp. 189-200). Springer International Publishing. https://doi.org/10.1007/978-3-319-47364-2_19

Nasonov, D., Melnik, M., Shindyapina, N., & Butakov, N. (2015). Metaheuristic Coevolution Workflow Scheduling in Cloud Environment: *Proceedings of the 7th International Joint Conference on Computational Intelligence*, 252-260. https://doi.org/10.5220/0005599402520260

Rodriguez, M. A., & Buyya, R. (2014). Deadline Based Resource Provisioningand Scheduling Algorithm for Scientific Workflows on Clouds. *IEEE Transactions on Cloud Computing*, *2*(2), 222-235. https://doi.org/10.1109/TCC.2014.2314655

Romero-Ocaño, A. D., Cosío-León, M. A., Valenzuela-Alcaraz, V. M., Avilés-Rodríguez, G. J., Martínez-Vargas, A., Romero-Ocaño, A. D., Cosío-León, M. A., Valenzuela-Alcaraz, V. M., Avilés-Rodríguez, G. J., & Martínez-Vargas, A. (2018). Efecto de la calibración de parámetros mediante un diseño Taguchi L934 en el algoritmo GRASP resolviendo el problema de rutas de vehículos con restricciones de tiempo. *Computación y Sistemas*, *22*(2), 657-673. https://doi.org/10.13053/cys-22-2-2595

Rosas, L., Gómez Santillán, C. G., Rangel-Valdez, N., Morales-Rodríguez, M. L., Huacuja, H. F., Vargas, M., Rosas, L., Gómez Santillán, C. G., Rangel-Valdez, N., Morales-Rodríguez, M. L., Huacuja, H. F., & Vargas, M. (2024). Experimental Analysis of a Cooperative Coevolutionary Algorithm with Parameter Tuning for Multi-objective Problem Optimization with Uncertainty. *Computación y Sistemas*, *28*(3), 1291-1319. https://doi.org/10.13053/cys-28-3-5185

Thakur, A., & Goraya, M. S. (2022). RAFL: A hybrid metaheuristic based resource allocation framework for load balancing in cloud computing environment. *Simulation Modelling Practice and Theory*, *116*, 102485. https://doi.org/10.1016/j.simpat.2021.102485

Wang, Z., Zheng, W., Chen, P., Ma, Y., Xia, Y., Liu, W., Li, X., & Guo, K. (2020). A Novel Coevolutionary Approach to Reliability Guaranteed Multi-Workflow Scheduling upon Edge Computing Infrastructures. *Security and Communication Networks*, *2020*, e6697640. https://doi.org/10.1155/2020/6697640

Wangsom, P., Lavangnananda, K., & Bouvry, P. (2019). Multi-Objective Scientific-Workflow Scheduling With Data Movement Awareness in Cloud. *IEEE Access*, *7*, 177063-177081. https://doi.org/10.1109/ACCESS.2019.2957998

WorkflowSim/WorkflowSim-1.0. (2025). [Java]. WorkflowSim. https://github.com/WorkflowSim/WorkflowSim-1.0 (Obra original publicada en 2013)

Wu, C. F. J., & Hamada, M. S. (2011). *Experiments: Planning, Analysis, and Optimization*. John Wiley & Sons.

Xie, H., Ding, D., Zhao, L., Kang, K., & Liu, Q. (2024). A two-stage preference driven multi-objective evolutionary algorithm for workflow scheduling in the Cloud. *Expert Systems with Applications*, *238*, 122009. https://doi.org/10.1016/j.eswa.2023.122009

Xu, Y., & Abnoosian, K. (2022). A new metaheuristic-based method for solving the virtual machines migration problem in the green cloud computing. *Concurrency and Computation: Practice and Experience*, *34*(3), e6579. https://doi.org/10.1002/cpe.6579

Zeedan, M., Attiya, G., & El-Fishawy, N. (2022). A Hybrid Approach for Task Scheduling Based Particle Swarm and Chaotic Strategies in Cloud Computing Environment. *Parallel Process. Lett.*, *32*(1), 2250001. https://doi.org/10.1142/S0129626422500013

Zeedan, M., Attiya, G., & El-Fishawy, N. (2023). Enhanced hybrid multi-objective workflow scheduling approach based artificial bee colony in cloud computing. *Computing*, *105*(1), 217-247. https://doi.org/10.1007/s00607-022-01116-y

Zhang, H., & Zheng, X. (2023). Knowledge-driven adaptive evolutionary multi-objective scheduling algorithm for cloud workflows. *Applied Soft Computing*, *146*, 110655. https://doi.org/10.1016/j.asoc.2023.110655

Zhang, X. (2024). Optimizing scientific workflow scheduling in cloud computing: A multi-level approach using whale optimization algorithm. *Journal of Engineering and Applied Science*, *71*(1), 175. https://doi.org/10.1186/s44147-024-00512-9