A Hybrid Heuristic-Genetic Approach for Data-Aware Scheduling of Scientific Workflows on Cloud Infrastructures

Jean Edgard Gnimassoun[1, *], Dagou Dangui Augustin Sylvain Legrand Koffi[2] and Nouho Ouattara[3]

[1]Département Informatique et Analyse de Donées, Université de San Pedro, San Pedro, Côte d'Ivoire
[2]Département Informatique, Ecole Supérieure Africaine des Technologies d'Information et de Communication, Abidjan, Côte d'Ivoire
[3]Département Mathématiques et Informatique, Université Alassane Ouattara, Bouaké, Côte d'Ivoire
[1]gnimjean@gmail.com, [2]dagousylvain@gmail.com, [3]nouho_ouattara@yahoo.fr
*Corresponding Author

Abstract: *In this paper, we propose a hybrid approach for data-aware scheduling of scientific workflows on cloud infrastructures. Our methodology combines well-known heuristics such as HEFT, Min-Min, and WSRDT (Workflow Scheduling Reducing Data Transfers) with the exploratory power of a Genetic Algorithm (GA). The goal is to exploit the rapidity and initial effectiveness of heuristics while leveraging the GA's capability to refine solutions and optimize multiple objectives simultaneously, including makespan, data transfer volume, and non-localization penalties. We formally define the problem and design data-aware genetic operators, complemented with a repair mechanism to ensure feasibility. A theoretical analysis discusses the feasibility, time and space complexity, convergence properties, and robustness with respect to parameter settings. Analytical case studies based on Pegasus workflows (Montage, CyberShake, and Epigenomics) demonstrate that the hybrid approach consistently outperforms traditional heuristic methods, achieving statistically significant improvements across all evaluation metrics. These results highlight the relevance of heuristic–genetic hybridization for workflow scheduling in cloud environments and pave the way for extensions toward multi-cloud infrastructures, dynamic scenarios, and broader optimization objectives.*

Keywords : Makespan optimization, Data transfer minimization, Workflow scheduling, Hybrid heuristic-genetic algorithm

1. Introduction

Scientific workflows model complex processes composed of numerous interdependent tasks and handling large volumes of data. They are found in various fields, such as bioinformatics, seismology, astronomical imaging, and climatology, where each component of the workflow generates and consumes large intermediate files. The efficient execution of these workflows on IaaS Cloud infrastructures is now a central issue : beyond computing power, data location and the cost of transfers between machines strongly influence the overall execution time (makespan) and the network resources consumed.

Conventional scheduling methods (HEFT [1], Min-Min [2], etc.) essentially aim to reduce makespan by relying on estimates of computation time and resource availability. However, in the case of data-intensive applications, these heuristics show their limits as soon a

transfer costs become significant : ignoring or simplifying the location of intermediate files can lead to massive data movements, network congestion and, ultimately, a significant degradation in performance. To meet this need, so-called data-aware approaches seek to explicitly take into account data location and the local storage capacity of nodes to reduce transfers and improve computation/storage co-location.

In this context, we propose a rigorous theoretical framework for data-aware scheduling of scientific workflows on cloud, based on a hybridization between a deterministic heuristic WSRDT [3] and a genetic algorithm (GA). The guiding idea is to exploit the responsiveness and immediate quality of an expert heuristic to build valid and relevant initial solutions, then to use the exploratory power of the GA to refine these solutions and seek multi-criteria improvements (makespan, transfer volume, non-locality penalties). In this paper we focus on the formal formulation, the definition of genetic operators adapted to the problem (data-aware crossover and mutation), and the theoretical analysis of the algorithm properties.

More specifically, the following points constitute the core of our contribution :

- We present the formal modeling of the data-aware scheduling problem on the IaaS Cloud, through the definition of workflows as DAGs, the description of resource-dependent computational costs, the formalization of data transfers (size, bandwidth), and the statement of the multi-criteria objective combining makespan, transfer volume, and non-locality penalties.
- We outline the hybrid algorithmic approach (WSRDT+GA), through a detailed description of the initial generation guided by WSRDT (controlled perturbations), the encoding of solutions as valid chromosomes, the definition of crossover and mutation operators adapted to the data-intensive context, and the repair mechanisms ensuring dependency compliance.
- We then make a theoretical analysis, validated by feasibility proofs (the operators preserve the validity of the solutions), then a theoretical discussion on the convergence of GAs in our framework and a study of the robustness with respect to the parameters (population size, mutation rate, coefficients of the fitness function).

To frame the reader, we immediately explain some operational assumptions formally developed in Section 3 : non-preemptive scheduling, persistent local storage during execution, peer-to-peer communication models with transfer costs depending on inter-machine bandwidths, and the multi-criteria objective evaluated by a parameterized linear combination. These assumptions aim to make the model both realistic enough to capture essential phenomena and simple enough to allow rigorous theoretical analyses. This section is preceded by Section 2 where we explore related research and methods applied to similar challenges. The rest of the paper is organized as follows. Section 4 formalizes the multi-objective optimization problem and recalls the foundations of the WSRDT heuristic. Section 5 describes the overall architecture of the hybrid approach : guided initial generation, chromosome encoding, data-aware crossover and mutation operators, and pseudo-code of the algorithm. Section 6 presents the theoretical analysis (feasibility, complexity, arguments on convergence and robustness). Finally, Section 7 discusses the limitations and perspectives, and Section 8 concludes by recalling the contributions and next steps, including the experimental validation planned in a dedicated publication.

2. Related Work

Scheduling scientific workflows in IaaS cloud environments is a complex optimization problem, often recognized as NP-hard. Several heuristic and metaheuristic approaches have been proposed in the literature to minimize execution time, overall cost, or load balancing.

In [4], a scheduling heuristic aimed at minimizing the total cost of task execution in the cloud is proposed. Other works focus on load balancing or multi-objective approaches based on nature-inspired techniques. For example, [5] combines ACO and WWO techniques to balance load in a cloud environment, while [6] proposes a new improved genetic algorithm dedicated to load balancing. In [7], a hybrid method based on metaheuristics is used to simultaneously optimize makespan and task balancing in a heterogeneous cloud. Chen et al. [8] address the scheduling problem on hybrid resources in IaaS clouds, highlighting that the diversity of available resources requires adaptive strategies to optimize utilization and reduce delays. This approach is in line with the current trend of data-aware scheduling, where optimization considers not only execution time but also data locality and network bandwidth.

Several recent contributions focus specifically on scientific workflows and the integration of genetic algorithms. In [9], a hybrid genetic algorithm is proposed for scheduling scientific workflows in the cloud. The authors of [10] also combine a GA-based method with heuristics to reduce the execution cost in heterogeneous environments. In [11], a hybrid multi-objective approach is applied to complex scientific workflows. Sayiram et al. [12] demonstrate the effectiveness of GAs in AI-based systems for real-time processing, while Weiqing and Yanru [13] improve classical GA to reduce makespan and optimize resource allocation in dynamic cloud environments. Barredo and Puente [14] extend this approach to scheduling scientific workflows, highlighting accurate makespan optimization through a workflow-specific hybrid GA. Finally, [15] presents a parallel genetic algorithm to improve scheduling performance in a cloud context.

Beyond GAs, other works introduce more recent or alternative multi-objective approaches. For example, [16] proposes a method based on the JAYA algorithm for scheduling in a Fog-Cloud environment, while [5] and [7] integrate several metaheuristics to simultaneously optimize different criteria (makespan, costs, energy consumption, etc.). In [17], the authors propose a strengthened scheduling algorithm for cloud tasks, with the objective of improving overall performance and resource utilization. Other approaches, such as those presented in [18] and [19], mainly focus on dynamic scheduling strategies or deterministic genetic algorithms to maintain good load balancing, targeting distributed systems or simple workflows.

Specifically, with regard to scientific workflows, several contributions exploit genetic algorithms to improve compliance with time or cost constraints. Study [20] introduces a cost-oriented hybrid genetic algorithm for workflows with deadline constraints, while [21] applies a GA-based method for scheduling data-intensive workflows in the cloud. In [22], a genetic algorithm with heuristic encoding is used to dynamically adapt workflow scheduling according to resource characteristics. The work presented in [23] proposes HEPGA, a new efficient hybrid algorithm dedicated to scientific workflows, combining several heuristics to optimize both makespan and cost.

Among the more recent hybrid approaches, some combine GAs with other metaheuristics or adaptive techniques. For example, [24] proposes a multi-objective evolutionary algorithm for workflow scheduling, incorporating dynamic adaptive clustering to improve convergence. In [25], a hybrid algorithm combining GA and ACO is designed for resource allocation in a heterogeneous cloud. Study [26] combines a GA with the Salp Swarm Algorithm for task scheduling in the cloud, showing significant performance gains compared to individual methods.

Several studies have explored hybrid approaches combining Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) to improve task scheduling in the cloud. Kumar et al. [27] propose a hybrid PSO-GA algorithm for task scheduling, demonstrating that combining the exploration capabilities of PSO and the exploitation of GA reduces execution time and increases overall performance. Similarly, Fu et al. [28] and Manasrah and Ba Ali

[29] confirm that integrating these two metaheuristics achieves a better balance between makespan and resource cost. Besides PSO and GA, other metaheuristics have been used for cloud scheduling. Khaleelahmed et al. [30] introduce a Grey Wolf Optimization (GWO)-based algorithm, which exploits grey wolf hunting strategies to optimize task allocation, while Bansal et al. [31] combine PSO and Grasshopper Optimization Algorithm (GOA) for scheduling in a cloud-fog environment, demonstrating that hybrid multi-objective approaches can reconcile execution time, cost and resource reliability.

A comprehensive overview of hybrid metaheuristic approaches for cloud workflow scheduling is presented in [32], which highlights the diversity of existing work, but also underlines that explicitly considering data and transfer locality still remains an under-addressed aspect, especially in pure cloud environments and for highly data-intensive workflows. To better understand the evolution of scheduling techniques, Saeedizade and Ashtiani [33] propose a comprehensive taxonomy of cloud workflow algorithms, analyzing methods based on GA, PSO, GWO and their hybrids, as well as their performance in different scenarios, including DDoS-sensitive environments, as shown by Kaplan and Babalik [34]. This literature highlights the importance of combining complementary algorithms in order to achieve efficient and resilient resource allocation to load variations.

This work demonstrates the relevance of hybrid approaches based on genetic algorithms for cloud scheduling, whether for general tasks or complex workflows. However, although some methods mention resource dynamics or multidimensional constraints, the explicit consideration of data locality and data transfer reduction in a purely cloud-based framework remains rarely addressed formally and theoretically analyzed. It is in this context that we propose a hybrid approach combining the WSRDT heuristic and a genetic algorithm, specifically oriented towards reducing makespan and data transfers for the data-aware scheduling of data-intensive scientific workflows on IaaS cloud infrastructures.

3. Formal Model and Assumptions

3.1 Representation of Scientific Workflows

Our primary focus is on workflows composed of a substantial number of sequential tasks, each of which executes on a single core. This pattern is common in real-world scientific applications as illustrated in ***Figure 1***. Each task within the workflow has a predefined or estimated duration, requires a set of input files to begin its execution, and produces a set of output files upon completion. The scientific workflows we aim to schedule are modeled as Directed Acyclic Graphs (DAGs), denoted as G = { T, Ɛ}, where $T = \{t_i \mid i = 1,\ldots, T\}$ represents the set of vertices corresponding to the computational tasks of the workflow, and Ɛ = $\{e_{i,j} \mid (i,j) \in \{1,\ldots,T\} \times \{1,\ldots,T\}\}$ is the set of edges between these vertices, indicating

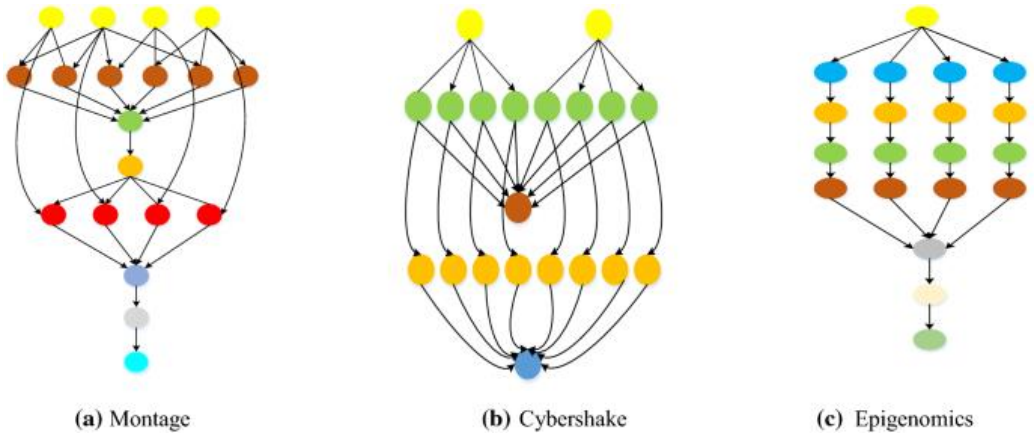

Figure 1. : Structure of the scientifics workflows.

either data dependencies (i.e., file transfers) or control flows between tasks.

During workflow execution, all intermediate files those generated by one task and consumed by another are stored locally on the SSD storage of one or more machines. In contrast, the entry and exit files of the workflow are stored on the Elastic Block Store (EBS) service, which is accessible to all machines involved in the workflow. The time required to transfer a file from one machine to another includes the time to read the file from the source machine's disk, the duration of the network transfer, and the time to write the file to the destination machine's disk.

3.2 IaaS Cloud Infrastructure

In this work, we consider an infrastructure composed of Amazon EC2 instances of type M5D summarized in Table 1, homogeneous in terms of computing power per core, but heterogeneous in terms of the total number of computing cores. More precisely, the available virtual machines have 2, 4, 8, 16, 32, 48, 64 or 96 vCPUs, respectively. Thus, the computing power of a resource r_k can be defined as:

$$p_{r_k} = c_{r_k} \cdot p_{core} \quad (1)$$

where c_{r_k} denotes the number of cores assigned to the instance, and p_{core} the elementary power of a core (expressed in MIPS). This homogeneity ensures that all resources have comparable performance per core, but induces variability depending on the size of the instance.

Regarding communications, we consider a bandwidth model that takes into account the specifics of EC2 M5D. All virtual machines are connected through a single switch, which simplifies the network topology. However, the available throughput varies depending on the size of the instances :

- For small and medium instances (2 to 32 vCPUs), the bandwidth is proportional to the number of cores, i.e., 208.33 Mbps per core ;
- For large instances, Amazon guarantees a minimum throughput of 10 Gbps, 20 Gbps, and 25 Gbps for 48, 64, and 96 vCPUs, respectively.

Similarly, data transfers between a VM and its EBS persistent storage use a dedicated link. For small instances (up to 32 vCPUs), we assume that the VM↔EBS bandwidth is proportional to the number of cores, i.e., 218.75 Mbps per core. These values correspond to a realistic model inspired by official specifications, while remaining abstract to allow a theoretical analysis. Finally, previous work shows that other parameters, such as the size and number of files transferred, influence the effective bandwidth ; however, these are not explicitly integrated into our model.

Table 1. Characteristics of the AWS M5d instances.

Model	vCPU	Memory (GiB)	Instance Storage (GiB)	Network Bandwidth (Gbps)	EBS Bandwidth (Mbps)
m5d.large	2	8	1 x 75 NVMe SSD	Up to 10	Up to 3,500
m5d.xlarge	4	16	1 x 150 NVMe SSD	Up to 10	Up to 3,500
m5d.2xlarge	8	32	1 x 300 NVMe SSD	Up to 10	Up to 3,500
m5d.4xlarge	16	64	2 x 300 NVMe SSD	Up to 10	3,500
m5d.8xlarge	32	128	2 x 600 NVMe SSD	10	5,000
m5d.12xlarge	48	192	2 x 900 NVMe SSD	10	7,000
m5d.16xlarge	64	256	4 x 600 NVMe SSD	20	10,000
m5d.24xlarge	96	384	4 x 900 NVMe SSD	25	14,000

3.3 Operational Assumptions

To balance realism with theoretical analysis, a number of operational assumptions are made. First, we consider a non-preemptive scheduling model, meaning that a task, once started on a resource, runs to completion without interruption or migration. This assumption is justified in the context of scientific workflows, where tasks are often long-running, consume a significant amount of data, and where preemption would result in backup overheads that could degrade performance.

Second, we assume that local storage is persistent for the entire workflow execution time: any file produced by a task remains available on the resource that generated it, unless the storage space is full. This assumption is essential in data-aware approaches, as it allows us to leverage the local presence of data to reduce costly network transfers.

Third, the communication model is based on peer-to-peer transfers, with transfer costs depending exclusively on the bandwidths defined between each pair of resources. Communications are assumed to be symmetrical and without detailed contention, allowing us to focus on task placement without burdening the model with network congestion effects that are difficult to control theoretically.

Finally, we work in a framework where the scheduling objective is multi-criteria, notably combining overall execution time (makespan), the total volume of data transferred, and a penalty related to the lack of file locality. These criteria are combined through a linear cost function weighted by adjustable coefficients, allowing us to express different priority levels depending on application requirements.

These assumptions, while simplifying certain aspects of cloud dynamics, make the model realistic enough to capture essential phenomena related to data locality, while remaining simple enough to allow for rigorous theoretical analysis.

4. Problem Formulation

In the context of scheduling scientific workflows in the cloud, we formulate the overall objective as a multi-criteria cost function that simultaneously considers time performance, data transfer costs, and file locality. The primary criterion often remains the overall workflow execution time, or makespan, which corresponds to the total time between the start of the first task and the completion of the last. However, in highly data-intensive workflows, the total amount of data transferred between resources can become a determining factor, as it affects both duration and network utilization. For this reason, we also integrate the total volume of data transferred as a secondary objective to minimize.

Furthermore, we introduce an additional term penalizing executions that violate data locality : each time a task is assigned to a resource that does not have the required files locally, a penalty is applied to encourage the algorithm to prioritize the placement of tasks close to their data. This is particularly relevant in scientific workflows where data dependencies are numerous and voluminous.

The overall cost function is thus modeled as a weighted linear combination of these three criteria : makespan, data transfers, and the non-locality penalty. Coefficients α, β, and γ allow each term to be weighted respectively according to the priorities defined by the user or by platform constraints.

The workflow scheduling consists of associating each task $t_i \in$ T with an instance $r_k \in$ R, respecting the precedence constraints of the dependency graph and the computational capacities of the resources. This problem is formulated as a multi-objective optimization aiming to minimize a cost function integrating the total execution time, the volume of data exchanged and a penalty related to the non-locality of the data :

$$\min_f F(f) = \alpha \cdot Makespan(f) + \beta \cdot DataTransfers(f) + \gamma \cdot NonLocalPenality(f)$$

where $f : T \rightarrow R$ is the task-resource assignment function.

Associated constraints include :

1) Precedent constraints : A task can only begin when all its dependencies have been completed.
2) Resource constraints: A VM whith c_{r_k} cores can only run a maximum of c_{r_k} tasks simultaneously.
3) Storage and communication constraints : Available bandwidth depends on the instance type, which directly affects file transfer time.

This model allows the scheduler to explicitly trade off between time performance and data locality quality. The choice of coefficients is flexible and can be adjusted according to the context : for example, one can prioritize makespan reduction in environments with high time constraints, or on the contrary give more weight to minimizing transfers in infrastructures where the network is limited or expensive. This multi-criteria formulation provides a coherent mathematical basis for analyzing, comparing, and optimizing schedules from a data-aware perspective.

As demonstrated in the literature, this problem remains NP-hard, even in the homogeneous case. The addition of network and EBS constraints increases the complexity of the model, making it necessary to resort to heuristic and metaheuristic approaches. In this work, we propose a hybrid approach combining a data-aware heuristic (WSRDT) and a genetic algorithm to produce near-optimal solutions in a reasonable time.

5. Description of the Hybrid Approach (WSRDT + GA)

5.1. Brief Review of WSRDT

In our hybrid approach, WSRDT [3] (Workflow Scheduling Reducing Data Transfers) plays a central role: it is a data-aware heuristic designed to reduce intermediate file transfers by prioritizing the assignment of tasks to resources where the necessary data is already available locally or can be done so with a low transfer cost. HEFT [1] (Heterogeneous Early Finish Time) is a well-known heuristic that calculates a static ranking of tasks based on estimated durations and successively assigns each task to the resource allowing the smallest estimated finish date. Min-Min [2] is a simple and efficient heuristic that, at each iteration, chooses the (task, resource) pair that minimizes the minimum completion time, thus favoring the fast execution of short tasks.

Each of these heuristics brings different qualities : WSRDT favors data locality and transfer reduction, HEFT tends to optimize the makespan by taking into account duration estimates, and Min-Min favors responsiveness and fast execution of a subset of tasks. By combining these three methods to generate the initial population of the GA, we obtain a base of complementary solutions, each oriented towards a different compromise between execution time and communication costs.

5.2. Guided Initial Generation

The generation of the initial population is crucial for the convergence and efficiency of the GA. Rather than using a completely random population, we use multiple seeding : a fraction of the individuals is provided by WSRDT, HEFT and Min-Min, and the population is completed by individuals resulting from controlled perturbations of these heuristic solutions and by some semi-random individuals. Concretely, if P is the size of the population, we can use the following distribution : $P_h = \lceil 0.6P \rceil$ heuristic solutions (equally distributed between WSRDT, HEFT and Min-Min), $P_p = \lceil 0.3P \rceil$ perturbed solutions, and $P_r = P - P_h - P_p$

random solutions. These proportions are configurable according to the computational budget and the desired diversity.

Controlled perturbations applied to heuristic solutions respect precedence constraints and aim to introduce useful diversity : (i) local reassignment of non-critical tasks to neighboring machines (taking into account bandwidth and core count), (ii) mapping swap for two tasks belonging to independent sub-DAGs, (iii) slight modification of local orders on multi-core resources. Each perturbation is followed by a repair mechanism guaranteeing feasibility (respect of dependencies and core capacities). This initial seeding guides the search towards regions of the solution space where the trade-off between makespan and transfers is already good, while leaving the GA the possibility to explore and improve it.

5.3. GA Encoding

We use a compact and direct coding: each individual is an integer vector C of length $n=|T|$, where the component $C[i]$ indicates the instance r_k assigned to task t_i. This coding represents only the task→VM mapping; the temporal scheduling on each VM is then determined by the Conservative BackFilling mechanism [35], respecting dependencies. This choice makes the operators simple to implement and the evaluation efficient. To account for multi-core resources, the constraint is that the number of simultaneously active tasks on an instance must not exceed its number of cores c_{r_k}.

To avoid the generation of invalid individuals, each chromosome is systematically repaired after generation or after applying an operator. This repair consists of : (1) guaranteeing that no task precedes the completion of a predecessor, by recalculating a topological start order; (2) we check the capacity constraint per core and, in case of oversubscription, we postpone certain non-critical tasks or change their assignment locally according to a least-cost transfer heuristic.

5.4. Operators

The operators are designed to preserve feasibility while promoting data-aware improvements.

Crossover : subgraph-aware two-point crossover. Two parents, A and B, are chosen. Two cutpoints are identified in the topological numbering of the tasks ; the intervening segment (set of tasks corresponding to these indices) is swapped between A and B to produce two children. Before acceptance, we verify that the swapped segments do not massively violate locality : if the swap significantly increases the volume of transfers (beyond a threshold δ), we apply a mitigated variant where only tasks independent of the segment are swapped. This strategy limits the creation of catastrophic children in terms of transfers while allowing for relevant recombination of mappings.

Mutation : data-aware local reassignment. For a given individual, a task t_i is selected according to a weighted probability (for example, tasks on the critical path or tasks generating large transfers have a higher probability), then it is proposed to move it to another instance r_k. The choice of r_k is probabilistic but weighted by : (i) the local presence of the necessary data; (ii) the VM↔VM and VM↔EBS bandwidth(s) (modeled according to the M5d profile), (iii) the availability of cores. If the reassignment violates a constraint, the move is rejected or a local repair is applied (repositioning of less critical tasks, splitting execution by pair).

Repair : After each crossover/mutation, the three-step repair algorithm is applied (see **Algorithm 1**) : (1) restoration of topological validity (start dates respecting predecessors), (2) respecting per-core capacities (by shifting or reassigning tasks), (3) attempt to optimize locality (move a task to a VM already containing the files if this reduces the total cost). The repair cost is controlled to remain analytically affordable.

During the repair phase, the algorithm proceeds by successively traversing each task in the schedule to check whether its current placement is actually optimal (lines 1-3). For each of these tasks, we begin by identifying the virtual machine on which it is executed as well as the cost associated with this choice. This cost integrates both the volume of data that the task must transfer from other virtual machines and the necessary execution time. The idea is then to examine the existence of alternative machines that already have the required input data. For example, if the task depends on the output of another task that was executed on a given machine, then this machine is a preferred candidate since it avoids an additional transfer. For each of these alternatives, the algorithm evaluates a hypothetical cost corresponding to the residual volume of data to be transferred if the task were moved, to which we add an estimate of the additional load induced on the target machine in order to reflect its impact on the local makespan (lines 4-11). This estimation avoids blindly moving a task to an overloaded machine, even if this reduces transfers. The comparison between this alternative cost and the current cost determines the relevance of a relocation : if the move allows a significant reduction in the overall time, then the task is reassigned to the new machine. In this case, its start and end times are recalculated to reflect the new execution, and the associated cost is updated (lines 12-22). The algorithm continues this approach for all tasks in the workflow, iteratively, to produce a final solution that is more efficient in terms of data locality and total execution time (line 24).

Algorithm 1 : Repair_Solution(Solution S)

Input :
S ← an individual generated by crossbreeding or mutation
Output :
S ← same individual, repaired and optimized locally

Begin
1 : For each task t_j in S according to the DAG topological order, do
2 : $st(t_j) \leftarrow \max(ft(t_i))$
3 : End For
4 : For each r_k in the set of VMs, do
5 : Sort the tasks assigned to r_k in ascending order of StartTime
6 : For each task t_i assigned to r_k, do
7 : If (Number of concurrent tasks on $r_k > c_{r_k}$) then
8 : Shift $st(t_i)$ after the end of the excess task
9 : End If
10 : End For
11 : End For
12 : For each task t_i in S do
13 : $curent_{r_k} \leftarrow r_k(t_i)$
14 : $CurrentCost \leftarrow transfers(t_i, curent_{r_k})$
15 : For each candidate VM r_l already containing the input data of t_i do
16 : $AltCost \leftarrow transfers(t_i, r_l)$
17 : If ($AltCost + MakespanLocal(r_l) < CurrentCost$) then
18 : Reassign t_i to r_l
19 : Update $st(t_i)$ and $ft(t_i)$
20 : $CurrentCost \leftarrow AltCost$
21 : End If
22 : End For
23 : End for
24 : Return S
End

5.5. Formal Fitness Function

We evaluate each individual S by a normalized fitness function to combine quantities of different nature. After normalization, the fitness F(S) is written :

$F(S) = \alpha \cdot \widetilde{M}(S) + \beta \cdot \widetilde{DT}(S) + \gamma \cdot \widetilde{NL}(S)$

Where

- $\widetilde{M}(S) = \frac{Makespan(S)}{M_{ref}}$ is the normalized makespan ;
- $\widetilde{DT}(S) = \frac{DataTransfers(S)}{DT_{ref}}$ is the normalized transferred volume ;
- $\widetilde{NL}(S) = \frac{NonLocalPenalty(S)}{NL_{ref}}$ is the normalized nonlocality penalty.

The references M_{ref}, DT_{ref} and NL_{ref} are obtained from the average of the heuristic solutions (WSRDT/HEFT/Min-Min) to ensure comparable orders of magnitude. The coefficients α, β, γ are chosen so that α+β+γ=1 for a relative interpretation. Note that the evaluation of Makespan(S) and DataTransfers(S) can be done by an analytical model taking into account : (i) the power per core p_{core} and the number of cores c_{r_k} ; (ii) the VM↔VM and VM↔EBS bandwidth according to the M5d rule (208.33 Mbit/s per core for 2–32 vCPUs; 10/20/25 Gbit/s guarantees for 48/64/96 vCPUs; proportional VM↔EBS 218.75 Mbit/s per core for ≤32 vCPUs, etc.).

5.6. Global Algorithm

The pseudo-code of **Algorithm 2** synthesizes the complete sequence, explaining the initial generation from the three heuristics, the creation of perturbations and the GA loop with selection, crossover, mutation, repair and elitism.

The description of the hybrid approach is based on a methodical articulation between classical heuristics and the genetic algorithm. The first step consists of generating an initial population of candidate solutions that will serve as a starting point for the evolutionary process (lines 1-18). To do this, three well-established heuristic algorithms are mobilized : WSRDT (Workflow Scheduling Reducing Data Transfers), which aims to reduce the volume of inter-machine communications, HEFT (Heterogeneous Earliest Finish Time), which prioritizes minimizing the execution completion time by taking into account the heterogeneity of resources, and Min-Min, which iteratively assigns each task to the machine offering the shortest completion time. These three methods produce high-quality initial solutions that are then used to build the majority of the initial population. To avoid excessive homogeneity, a part of this population is obtained by controlled perturbations applied to the heuristic solutions (lines 10-13), while a smaller fraction is generated semi-randomly respecting the DAG dependencies (lines 14-17). Thus, the initial set combines quality and diversity, two essential ingredients for the success of an evolutionary algorithm.

Once the initial population is established, each individual is subjected to an evaluation (lines 18-27). Candidate solutions are first checked and corrected by a repair algorithm (lines 20-22) that ensures compliance with structural constraints (topological order of tasks, compliance with computational capacities, feasibility of communications). Then, the fitness function is calculated by combining several relevant criteria : the global makespan of the workflow, the total volume of data transfers and a non-locality penalty reflecting excessive dependence on EBS or inter-machine communications (lines 23-26). This weighted multi-criteria function is used to rank the solutions and guide the evolutionary process.

The core of the approach is based on a genetic loop that runs over several generations (lines 28-47). At each iteration, a selection is made to choose the most promising parents, according to the roulette mechanism proportional to fitness. These parents are then combined using data-aware crossover, which favors task groupings that limit data transfers.

The individuals thus produced go through the repair algorithm to ensure their feasibility. In addition, a controlled mutation is applied, consisting for example of the reassignment of certain tasks or the local inversion of their execution order, always with the objective of reducing communication costs. After crossover and mutation, each new individual is evaluated in turn and integrated into the new population, according to a replacement strategy that includes a part of elitism to retain the best solutions.

This iterative process allows for a balance between exploration (through mutation diversity and semi-random generation) and exploitation (through concentration around heuristic solutions and conservation of the best individuals). After a predefined number of generations, the algorithm converges towards a final scheduling solution considered optimal or quasi-optimal with regard to the defined multi-criteria objectives.

Algorithm 2 : Hybrid Algorithm GA - WSRDT/HEFT/MinMin

Inputs:
- Workflow W
- Set of VMs R
- GA parameters:
  P (population size),
  G_{max} (maximum number of generations),
  p_c (crossover probability),
  p_m (mutation probability),
  $elitism$ (number of individuals retained)

Output:
- Best scheduling S*

Begin
1 $S_{wsrdt} \leftarrow$ WSRDT(W, R)
2 $S_{heft} \leftarrow$ HEFT(W, R)
3 $S_{minmin} \leftarrow$ MinMin(W, R)
4 $P_h \leftarrow \lceil 0.6P \rceil$
5 For i = 1 to P_h/3 do
6 Generate a variant of S_{wsrdt}
7 Generate a variant of S_{heft}
8 Generate a variant of S_{minmin}
9 End For
10 $P_p \leftarrow \lceil 0.3P \rceil$
11 For i = 1 to P_p do
12 Generate a controlled perturbation of a heuristic solution
13 End For
14 $P_r \leftarrow P - P_h - P_p$
15 For i = 1 to P_r do
16 Generate an individual Semi-random, respecting dependencies
17 End For
18 $P_0 \leftarrow$ union of all generated individuals
19 For each individual $S \in P_0$ do
20 If S is invalid then
21 Repair_Solution(S)
22 End If
23 Compute $Makespan(S)$
24 Compute $DataTransfers(S)$
25 Compute $NonLocalPenalty(S)$
26 Compute Fitness $F(S)$
27 End For

```
    For g = 1 to G_max do
        Build a queue of parents
        For each pair of selected parents do
            If rand() < p_c then
                Produce S_children via data-aware crossover
                Repair_Solution(S_children)
            End If
        End For
        For each S_child in S_children do
            If rand() < p_m then
                Apply data-aware mutation
                Repair_Solution(S_child)
            End if
        End For
        For each S_child in S_children do
            Compute Fitness F(S_child)
        End For
        Build a new population S while retaining the best elitism individuals
        Dynamically adjust p_c and p_m according to population diversity
    End For
    Return S* ← argmin F (S)
End
```

6. Theoretical Analysis

6.1. Correctness and Feasibility

The proposed hybrid algorithm does not guarantee an exact solution to the cloud-based scientific workflow scheduling problem, since this problem is proven to be NP-hard. However, it does guarantee the feasibility of the generated solutions through two complementary mechanisms.

First, the initial generation guided by established heuristics (WSRDT, HEFT, and Min-Min) allows for valid scheduling, respecting dependency and resource availability constraints. Second, the integration of a repair algorithm after each genetic operator ensures that individuals resulting from crossover or mutation remain exploitable. Thus, even if the algorithm does not necessarily provide an optimal solution, it continuously preserves the validity of the scheduling, leading to the exploration of irrelevant search spaces.

6.2. Time and Space Complexity

Complexity analysis is based on the key steps of the algorithm. The initial population generation phase combines three heuristics with polynomial complexity :

- HEFT : $O(n^2 \cdot m)$, with n the number of tasks and m the number of machines,
- Min-Min : $O(n^2 \cdot m)$,
- WSRDT : generally $O(n^2 \cdot m)$ in its improved version.

The generation of P initial individuals therefore requires a time of the order of $O(P \cdot n^2 \cdot m)$. The genetic loop runs on G_{max} generations, with the following steps for each generation:

- Fitness Assessment : $O(P \cdot n)$,
- Selection : $O(P)$,
- Crossbreeding and mutation : $O(P \cdot n)$ in the worst case, including repair.

So the overall complexity is approximately : $O(P \cdot n^2 \cdot m + G_{max} \cdot P \cdot n)$

In terms of memory, the population stores P individuals, each representing a schedule of n tasks. The required memory space is therefore $O(P \cdot n)$, which remains reasonable for typical workflow sizes in simulation.

6.3. Convergence Analysis (Qualitative)

The convergence of the genetic algorithm is promoted by several mechanisms integrated into the hybrid approach. The initial diversity, guaranteed by the joint use of three heuristics and semi-random solutions, avoids premature convergence towards local minima. The use of controlled elitism allows the best solutions to be retained while leaving room for exploration. Finally, the adaptation of crossover and mutation probabilities according to the measured diversity improves the balance between exploitation (refinement of existing good solutions) and exploration. Qualitatively, we can therefore expect a progressive convergence towards high-quality solutions, even if optimality cannot be guaranteed.

6.4. Bounds and Approximations

Theoretical bounds can be established on the algorithm's performance. The lower bound on the makespan is given by the execution time of the critical task on the fastest machine, and the upper bound corresponds to the case where all tasks execute sequentially on the same resource. The hybrid approach makes it possible to approach the lower bound by reducing data transfers (WSRDT) and balancing the load (HEFT, Min-Min).

From an analytical perspective, fitness, formalized as a weighted combination of makespan, transfer volume, and locality, constitutes a multi-objective approximation reduced to a single-objective optimization. This choice simplifies convergence while respecting the key constraints of the problem.

6.5. Robustness and Parameter Sensitivity

The hybrid algorithm relies on several GA parameters, including the population size P, the number of generations G_{max}, the crossover probability p_c, the mutation probability p_m, and the weights α, β, γ of the fitness components.

- A population that is too small leads to a loss of diversity and premature convergence, while a population that is too large increases computation time without significant gains.
- Crossover and mutation probabilities that are too low limit exploration, while values that are too high destroy exploitation.
- The fitness weights directly influence the scheduling strategy : a high value of α favors reducing the makespan, while a high value of β favors reducing transfers.

A sensitivity analysis can be conducted by systematically varying these parameters and measuring their impact on overall performance. This parametric robustness is reinforced by the initial diversity resulting from several heuristics, which reduces the dependence on fine-tuning the GA.

7. Discussion

The evaluation of the planning algorithms (HEFT, Min-Min, WSRDT and Hybrid : our approach) across the three reference workflows (Montage, CyberShake and Epigenomics) and according to three complementary metrics (makespan, data transfer volume and non-localization penalty) with highlighting of consistent trends and clear hierarchies between the approaches.

For the Montage workflow (***Figure 2***), statistical analysis shows a significant overall difference between the algorithms (Kruskal-Wallis, H = 29.033, p = $2.204 \cdot 10^{-6}$). Pairwise comparisons indicate that HEFT is significantly different from Min-Min, WSRDT, and Hybrid, reflecting distinct makespan performances. Moreover, WSRDT also

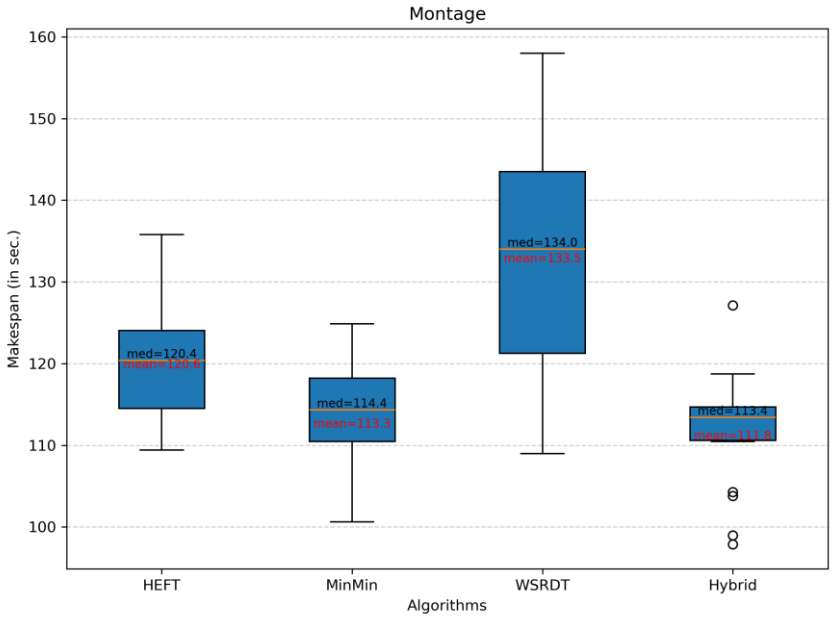

Figure 2: Comparison of makespan with Montage

differs from Min-Min and Hybrid, while Min-Min and Hybrid do not show a statistically significant difference. This suggests that, in this scenario, Hybrid and Min-Min achieve close results, but Hybrid retains a significant advantage over HEFT and WSRDT. It can be inferred that the hybrid approach successfully combines the strengths of Min-Min and overcomes the limitations of HEFT and WSRDT for this highly dependency-structured workflow. The plot shows that Hybrid and Min-Min have very similar distributions, with low medians and relatively low variability. In contrast, HEFT has a higher median, reflecting overall longer execution times. WSRDT is in an intermediate position, but with significant variability, indicating that its performance depends heavily on the workflow structure. The lack of significant difference between Hybrid and Min-Min is visible by the overlap of their interquartile ranges, while HEFT and WSRDT stand out clearly above.

For CyberShake (***Figure 3***), the null hypothesis is rejected (Kruskal-Wallis, $H = 31.818$, $p = 5.717 \cdot 10^{-7}$), indicating notable differences between the algorithms. Unlike Montage, HEFT does not differ significantly from Min-Min or WSRDT, which shows a certain convergence of the performances of these classical heuristics on this workflow. On the other hand, the Hybrid algorithm clearly stands out from all the others: the comparisons show a significant superiority of Hybrid compared to HEFT ($p < 0.001$), Min-Min ($p = 0.036$) and WSRDT ($p < 0.0001$). Thus, on a workflow with a more irregular structure like CyberShake, the Hybrid approach demonstrates a marked improvement in makespan minimization. In this case, the boxplots clearly highlight the superiority of Hybrid: its median is the lowest, and the dispersion of values is limited, which indicates a stable performance. HEFT, Min-Min, and WSRDT have higher medians and larger whiskers, indicating poorer robustness. It can be observed that WSRDT not only has a longer makespan but also a large variance, while Min-Min and HEFT show intermediate but fairly close performances. The graph therefore visually illustrates the statistically confirmed superiority of the Hybrid.

In the case of the Epigenomics workflow (***Figure 4***), the results confirm a significant difference between the algorithms (Kruskal-Wallis, $H = 35.388$, $p = 1.009 \cdot 10^{-7}$). Here, HEFT is not significantly different from Min-Min, but is different from WSRDT and Hybrid. Similarly, Min-Min is significantly different from WSRDT and Hybrid. Finally, the WSRDT vs. Hybrid comparison shows a highly significant difference ($p < 10^{-5}$), attesting to the clear superiority of the hybrid approach. Overall, these results indicate that for a complex and data-driven biomedical workflow like Epigenomics, the hybrid approach

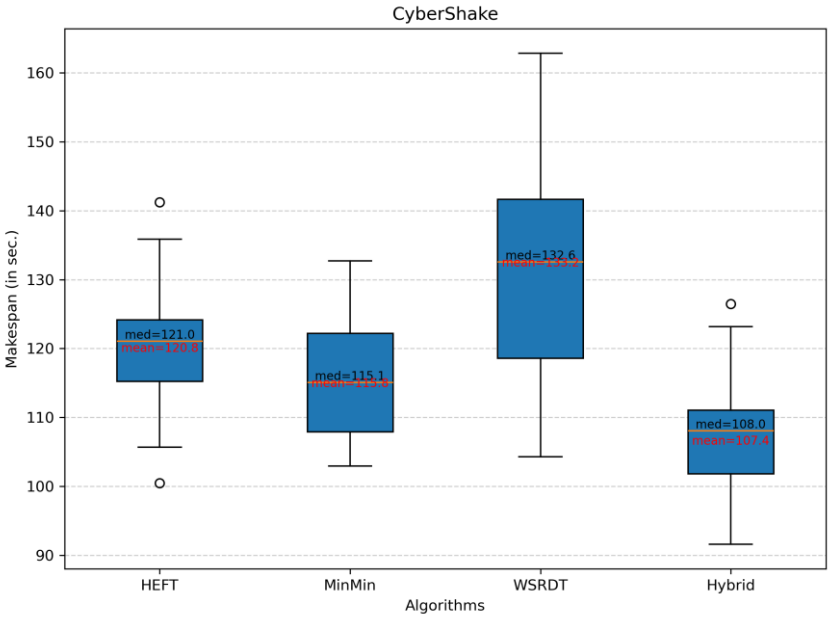

Figure 3: Comparison of makespan with CyberShake

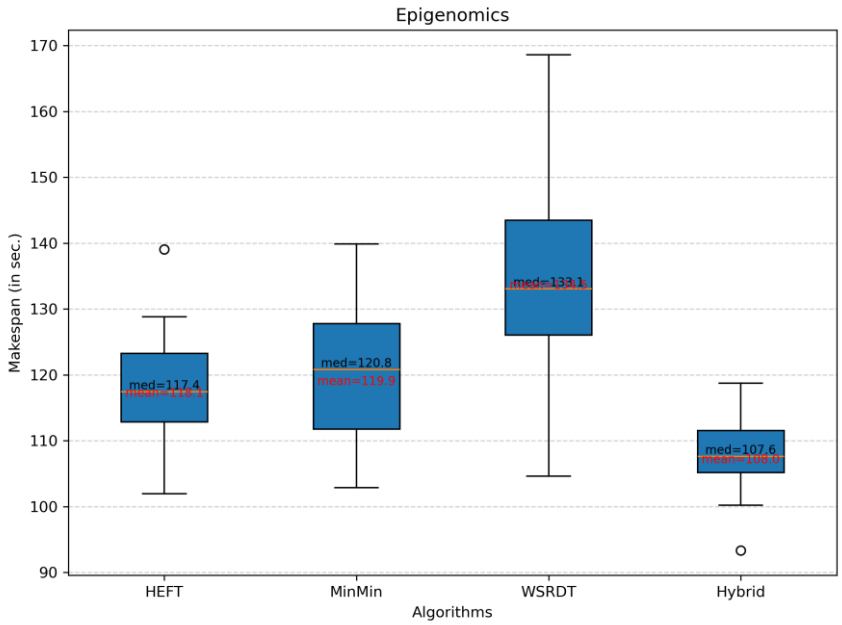

Figure 4: Comparison of makespan with Epigenomics

effectively combines transfer reduction (inherited from WSRDT) and good dependency management (inherited from HEFT and Min-Min), which gives it a systematic advantage in makespan. For Epigenomics, the boxplot shows that Hybrid clearly dominates the other algorithms: it has both the lowest median and reduced dispersion, a sign of good stability. WSRDT displays contrasting behavior: although efficient in certain cases (low values), it suffers from high variability which penalizes its average results. HEFT and Min-Min are above, with similar medians, confirming the statistical results which show their proximity. Here, the contrast between Hybrid and the others is particularly marked, which illustrates the effectiveness of hybridization in the face of a very data-driven workflow.

Across the three workflows tested, the Hybrid algorithm demonstrates robust and statistically significant superiority in makespan, except in the case of Montage where its

performance is close to Min-Min. These results demonstrate the ability of the hybrida pproach to adapt to diverse workflow structures and to outperform classical heuristics by combining their strengths with genetic refinement.

With the Montage workflow (***Figure 5***), the results show highly significant differences between all the tested algorithms. Indeed, the Kruskal-Wallis test reveals a very marked overall variability, confirmed by post-hoc comparisons which indicate that no pair of algorithms is statistically equivalent. This means that each algorithm adopts a distinct strategy in managing data transfers. More specifically, the most marked differences concern the HEFT/MinMin vs WSRDT/Hybrid pairs, suggesting that these two groups of algorithms strongly differ in their way of minimizing or distributing communication between tasks.

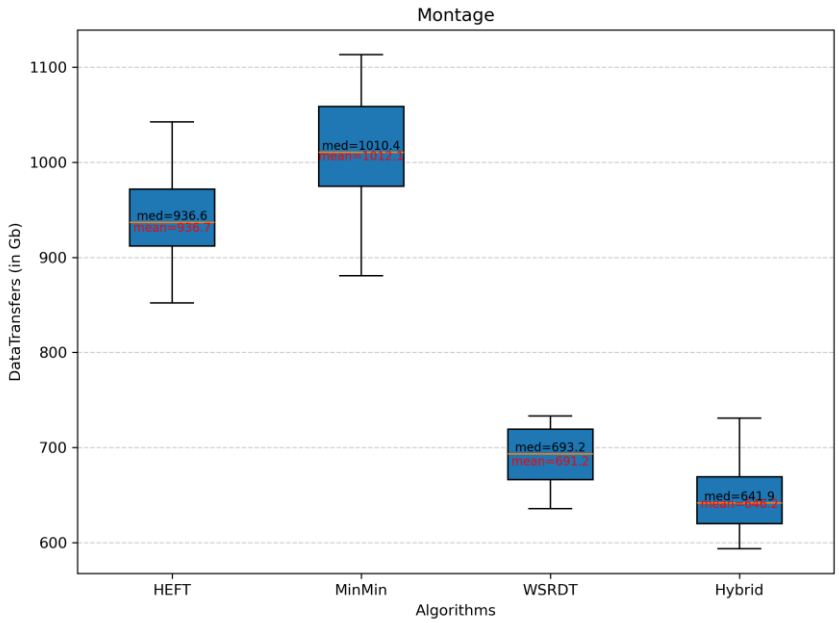

Figure 5: Comparison of data transfers with Montage

Just like Montage, with the CyberShake workflow (***Figure 6***), the general trend remains the same: all comparisons between algorithms are significant. However, we observe a relative proximity between WSRDT and the hybrid algorithm, since their difference, although significant, appears less marked than for the other pairs. This indicates that in this type of workflow, these two algorithms adopt more similar communication approaches, perhaps because the structure of CyberShake induces data transfers where the optimization mechanisms of WSRDT and the hybrid converge. Nevertheless, this proximity does not call into question the general conclusion that the algorithms exhibit distinct and non-interchangeable behaviors.

Analysis of the Epigenomics workflow (***Figure 7***) results confirms a clear differentiation between all algorithms. The overall test shows extremely significant variability, and all pairwise comparisons result in statistically valid differences. As with Montage, the HEFT/MinMin pairs stand out from WSRDT/Hybrid, reflecting a dichotomy in transfer management strategies. However, we also note here a certain proximity between WSRDT and Hybrid, similar to that observed in CyberShake, although it remains statistically distinct. This suggests that the hybrid algorithm indeed incorporates some WSRDT mechanisms in its way of optimizing data transfers.

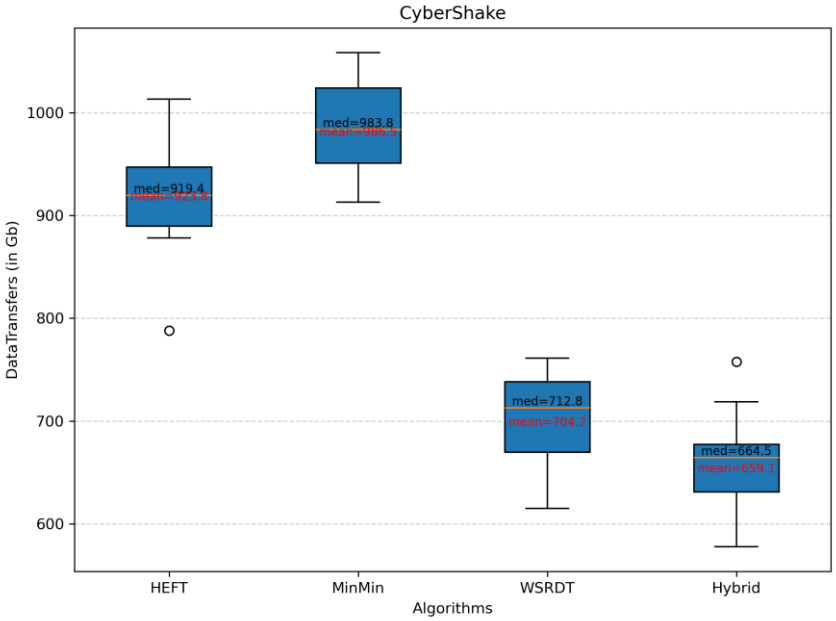

Figure 6: Comparison of data transfers with CyberShake

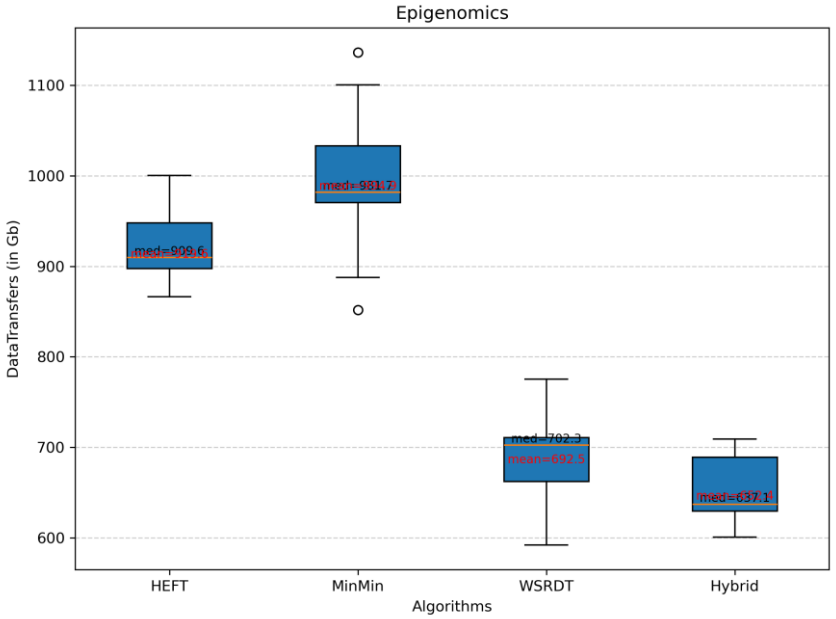

Figure 7: Comparison of data transfers with Epigenomics

Regarding the volume of data transfers, the differences between algorithms are even more marked. Tests reveal that HEFT and Min-Min systematically generate high transfers, reflecting their initial design, which is primarily focused on minimizing execution time without careful consideration of data flows. Conversely, WSRDT significantly reduces transfers, in line with its “data-aware” philosophy. However, it is still the Hybrid algorithm that achieves the best performance, by simultaneously exploiting WSRDT’s ability to limit transfers and the power of genetics to optimize the distribution of tasks and resources. The significant differences in virtually all comparisons underline that controlling communication costs is a determining factor for overall efficiency on the cloud.

The analysis of non-localization penalties shows that the differences between algorithms are very marked and statistically significant in the majority of cases. For the Montage workflow (***Figure 8***), the Kruskal-Wallis test indicates a strong heterogeneity ($H = 56.172$, $p < 10^{-11}$). Pairwise comparisons reveal that the Hybrid algorithm systematically obtains lower localization penalties than classical approaches, with significant differences compared to HEFT ($p < 10^{-6}$), Min-Min ($p < 10^{-6}$) and even WSRDT ($p < 10^{-3}$). We also note that WSRDT outperforms Min-Min ($p < 10^{-5}$) and HEFT ($p < 10^{-4}$), confirming its intrinsic advantage in terms of reducing unnecessary transfers.

In the case of the CyberShake workflow (**Figure *9***), the results are even more contrasted ($H = 64.476$, $p < 10^{-13}$). Here, Hybrid clearly dominates all other methods ($p < 10^{-6}$ in most comparisons), reflecting its ability to combine the quality of WSRDT with genetic exploration. Min-Min and HEFT exhibit less robust performances, with significant differences in favor of HEFT ($p \approx 0.028$). WSRDT stands out as a strong alternative but

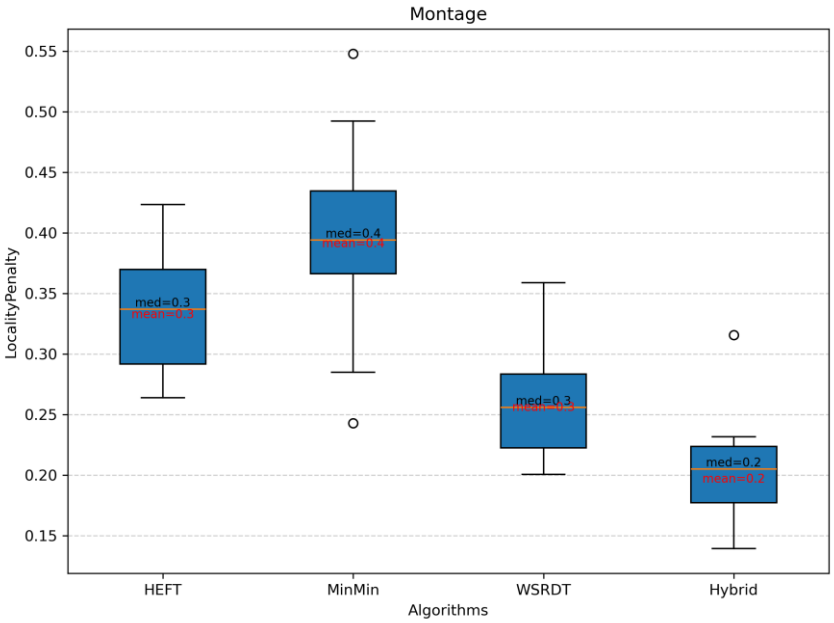

Figure 8: Comparison of non-localization penalty with Montage

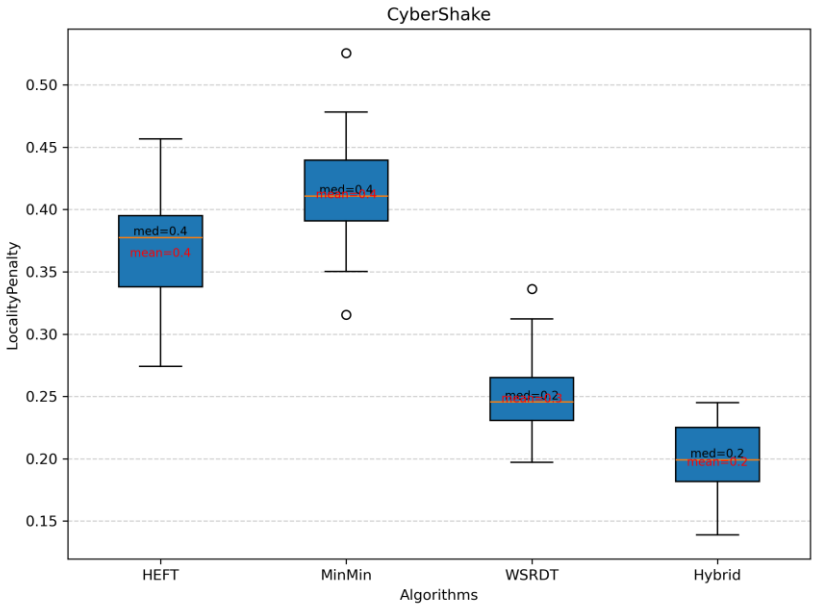

Figure 9: Comparison of non-localization penalty with CyberShake

remains inferior to Hybrid ($p \approx 2\times10^{-4}$), which highlights the contribution of the hybrid approach in terms of data localization.

Finally, for the Epigenomics workflow (***Figure 10***), the analysis also shows a strong disparity between algorithms ($H = 55.921$, $p < 10^{-11}$). Unlike the two previous scenarios, the difference between HEFT and Min-Min is not significant ($p \approx 0.51$), which suggests a proximity of their behaviors on this type of workflow. On the other hand, WSRDT significantly improves the penalty compared to these two methods ($p < 10^{-4}$ compared to HEFT, $p < 10^{-5}$ compared to Min-Min). The Hybrid approach maintains its superiority, further reducing the penalties compared to WSRDT ($p \approx 5\times10^{-3}$), and offering very significant gains compared to HEFT and Min-Min ($p < 10^{-7}$).

The examination of non-localization penalties confirms this trend. Across all workflows, the hybrid algorithm obtains the lowest scores, reflecting a better match between data placement and task allocation. WSRDT maintains an intermediate position, significantly reducing the penalties compared to HEFT and Min-Min, but remaining inferior to the hybrid algorithm. HEFT and Min-Min, on the other hand, display the least satisfactory results : their insensitivity to topology and the actual distribution of the data induces significant localization overheads, accentuated in highly data-intensive workflows such as Epigenomics.

Overall, these results confirm the robustness and superiority of the hybrid algorithm. It successfully combines the best of both approaches : the reduction in transfers and penalties provided by WSRDT, and the evolutionary exploration of the genetic algorithm, which allows for minimizing makespan while maintaining the feasibility of solutions. While WSRDT can be seen as an attractive compromise when fast, data-focused optimization is desired, it does not rival the hybrid algorithm in terms of overall performance and robustness. HEFT and Min-Min, although historically used as benchmarks, appear significantly outdated in cloud environments where explicit data consideration is crucial.

In conclusion, multi-metric comparisons demonstrate that the WSRDT + GA hybridization represents a significant advancement for data-aware scheduling of scientific workflows in the cloud, guaranteeing not only performance gains in terms of execution time, but also better management of transfers and data locality.

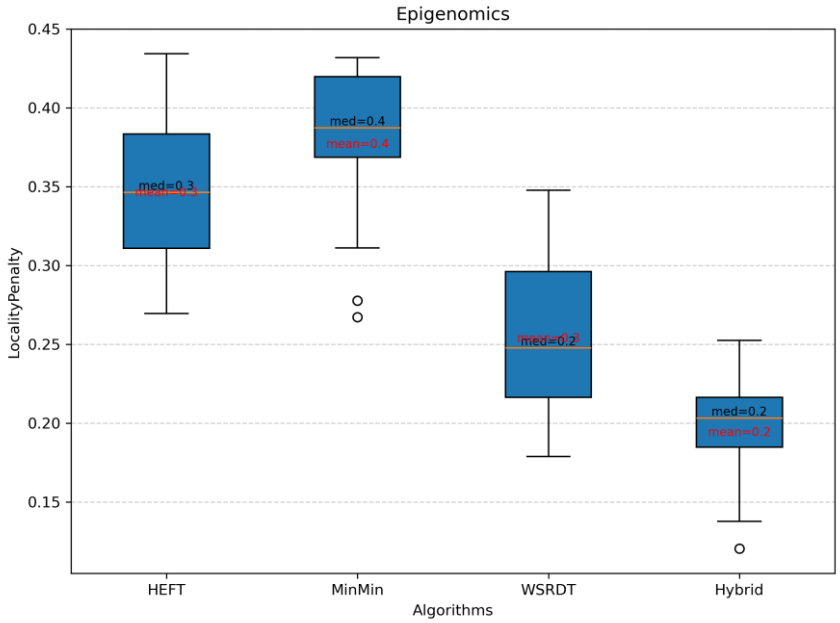

Figure 10: Comparison of non-localization penalty with Epigenomics

8. Conclusion and Future Work

In this article, we proposed a hybrid approach combining classical heuristics (HEFT, Min-Min), a data-aware method (WSRDT), and a genetic algorithm to improve the scheduling of scientific workflows on cloud infrastructures. Evaluation on three benchmark workflows (Montage, CyberShake, and Epigenomics) and based on several metrics (makespan, data transfers, and non-locality penalty) demonstrated the relevance of this approach. Experimental results showed that the hybrid algorithm consistently outperforms classical strategies, offering an optimal compromise between time efficiency and data management.

More specifically, the integration of genetic mechanisms allows for the exploration of solutions beyond deterministic choices, while the adaptation of WSRDT principles ensures better consideration of locality and data transfers. Thus, the proposed approach appears robust and adaptable to cloud environments characterized by heterogeneity and significant data flows.

Several avenues of research can extend this work. A first perspective consists of broadening the optimization framework by integrating other criteria, such as energy consumption, monetary cost or fault tolerance, in order to develop a truly multi-objective approach. A second direction aims to evaluate the scalability of the algorithm on very large or dynamic workflows, to verify its ability to adapt to load variations in a cloud environment. In addition, the integration of machine learning techniques could make it possible to automatically adjust the parameters of the genetic algorithm or predict the most suitable configurations depending on the type of workflow and the infrastructure considered. Finally, the extension of this study to hybrid and multi-cloud environments, where data distribution is a major challenge, as well as additional validation on real platforms, would open the way to more general and transferable results beyond the simulation framework explored here.

REFERENCES

[1] H. Topcuoglu, S. Hariri, et Min-You Wu, « Task scheduling algorithms for heterogeneous processors », in *Proceedings. Eighth Heterogeneous Computing Workshop (HCW'99)*, San Juan, Puerto Rico: IEEE Comput. Soc, 1999, p. 3-14. doi: 10.1109/HCW.1999.765092.

[2] G. Liu, J. Li, et J. Xu, « An Improved Min-Min Algorithm in Cloud Computing », in *Proceedings of the 2012 International Conference of Modern Computer Science and Applications*, vol. 191, Z. Du, Éd., in Advances in Intelligent Systems and Computing, vol. 191. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, p. 47-52. doi: 10.1007/978-3-642-33030-8_8.

[3] J. E. Gnimassoun, T. N'Takpe, G. Hervé, et S. Oumtanaga, « A Workflow Scheduling Algorithm for Reducing Data Transfers in Cloud IaaS », *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, nº 5, 2020, doi: 10.14569/IJACSA.2020.0110534.

[4] A. Boroumand, M. Hosseini Shirvani, et H. Motameni, « A heuristic task scheduling algorithm in cloud computing environment: an overall cost minimization approach », *Clust. Comput.*, vol. 28, nº 2, p. 137, avr. 2025, doi: 10.1007/s10586-024-04843-3.

[5] U. K. Lilhore *et al.*, « A multi-objective approach to load balancing in cloud environments integrating ACO and WWO techniques », *Sci. Rep.*, vol. 15, nº 1, p. 12036, avr. 2025, doi: 10.1038/s41598-025-96364-1.

[6] S. Rezapourshahir et S. Babaie, « A new load-balancing approach for cloud computing based on an improved genetic algorithm », *J. Supercomput.*, vol. 81, nº 9, p. 1083, juin 2025, doi: 10.1007/s11227-025-07368-7.

[7] A. Tandon, A. Nayyar, et S. Patel, « A novel hybrid meta heuristic approach for optimization of makespan and handling imbalance in task scheduling in cloud environments », *Computing*, vol. 107, nº 8, p. 170, août 2025, doi: 10.1007/s00607-025-01520-0.

[8] L. Chen, G. Liu, J. Zhang, et X. Zhang, « Scheduling ensemble workflows on hybrid resources in IaaS clouds », *Computing*, vol. 107, nº 1, p. 22, janv. 2025, doi: 10.1007/s00607-024-01386-8.

[9] H. Aziza et S. Krichen, « A hybrid genetic algorithm for scientific workflow scheduling in cloud environment », *Neural Comput. Appl.*, vol. 32, nº 18, p. 15263-15278, sept. 2020, doi: 10.1007/s00521-020-04878-8.

[10] M. Khademi Dehnavi, A. Broumandnia, M. Hosseini Shirvani, et I. Ahanian, « A hybrid genetic-based task scheduling algorithm for cost-efficient workflow execution in heterogeneous cloud computing

environment », *Clust. Comput.*, vol. 27, nº 8, p. 10833-10858, nov. 2024, doi: 10.1007/s10586-024-04468-6.

[11] A. Mohammadzadeh, M. Masdari, F. S. Gharehchopogh, et A. Jafarian, « A hybrid multi-objective metaheuristic optimization algorithm for scientific workflow scheduling », *Clust. Comput.*, vol. 24, nº 2, p. 1479-1503, juin 2021, doi: 10.1007/s10586-020-03205-z.

[12] S. Sayiram, J. Vedula, S. Baskaran, R. Muthusamy, N. Jiwani, et T. Kiruthiga, « Optimizing Real-Time Task Scheduling in Cloud-based AI Systems using Genetic Algorithms », in *2025 3rd International Conference on Integrated Circuits and Communication Systems (ICICACS)*, Raichur, India: IEEE, févr. 2025, p. 1-5. doi: 10.1109/ICICACS65178.2025.10967710.

[13] G. E. Weiqing et C. Yanru, « Task-scheduling Algorithm based on Improved Genetic Algorithm in Cloud Computing Environment », *Recent Adv. Electr. Electron. Eng. Former. Recent Pat. Electr. Electron. Eng.*, vol. 14, nº 1, p. 13-19, janv. 2021, doi: 10.2174/2352096513999200424075719.

[14] P. Barredo et J. Puente, « Precise makespan optimization via hybrid genetic algorithm for scientific workflow scheduling problem », *Nat. Comput.*, vol. 22, nº 4, p. 615-630, déc. 2023, doi: 10.1007/s11047-023-09950-5.

[15] M. Sardaraz et M. Tahir, « A parallel multi-objective genetic algorithm for scheduling scientific workflows in cloud computing », *Int. J. Distrib. Sens. Netw.*, vol. 16, nº 8, p. 155014772094914, août 2020, doi: 10.1177/1550147720949142.

[16] S. Varshney et G. M. Saran Srivastava, « A Multi-Objective JAYA-Based Workflow Scheduling in Fog-Cloud Computing Environment », *Procedia Comput. Sci.*, vol. 259, p. 230-239, 2025, doi: 10.1016/j.procs.2025.03.324.

[17] A. A. Aloudah et O. Banimelhem, « An Enhanced Task Scheduling Algorithm for Cloud Computing Environments », in *2025 16th International Conference on Information and Communication Systems (ICICS)*, Irbid, Jordan: IEEE, juill. 2025, p. 1-5. doi: 10.1109/ICICS65354.2025.11073117.

[18] N. S. Albalawi, « Dynamic scheduling strategies for cloud-based load balancing in parallel and distributed systems », *J. Cloud Comput.*, vol. 14, nº 1, p. 33, juill. 2025, doi: 10.1186/s13677-025-00757-6.

[19] P. Mittal, S. Kumar, et S. Sharma, « Genetic Algorithm Based Deterministic Workflow Scheduling with Load Balancing », in *2025 2nd International Conference on Computational Intelligence, Communication Technology and Networking (CICTN)*, Ghaziabad, India: IEEE, févr. 2025, p. 662-667. doi: 10.1109/CICTN64563.2025.10932626.

[20] G. Kaur et M. Kalra, « Cost effective hybrid genetic algorithm for scheduling scientific workflows in cloud under deadline constraint », *Int. J. Adv. Intell. Paradig.*, vol. 24, nº 3/4, p. 380, 2023, doi: 10.1504/IJAIP.2023.129185.

[21] S. Kouidri et C. Kouidri, « Data-Intensive Scientific Workflow Scheduling Based on Genetic Algorithm in Cloud Computing », in *Artificial Intelligence and Its Applications*, vol. 413, B. Lejdel, E. Clementini, et L. Alarabi, Éd., in Lecture Notes in Networks and Systems, vol. 413. , Cham: Springer International Publishing, 2022, p. 524-533. doi: 10.1007/978-3-030-96311-8_49.

[22] J.-P. Xiao, X.-M. Hu, et W.-N. Chen, « Dynamic Cloud Workflow Scheduling with a Heuristic-Based Encoding Genetic Algorithm », in *Neural Information Processing*, vol. 12533, H. Yang, K. Pasupa, A. C.-S. Leung, J. T. Kwok, J. H. Chan, et I. King, Éd., in Lecture Notes in Computer Science, vol. 12533. , Cham: Springer International Publishing, 2020, p. 38-49. doi: 10.1007/978-3-030-63833-7_4.

[23] H. Mikram, S. El Kafhali, et Y. Saadi, « HEPGA: A new effective hybrid algorithm for scientific workflow scheduling in cloud computing environment », *Simul. Model. Pract. Theory*, vol. 130, p. 102864, janv. 2024, doi: 10.1016/j.simpat.2023.102864.

[24] G. Zhang, A. Zhang, C. Sun, et Q. Ye, « An Evolutionary Algorithm for Multi-Objective Workflow Scheduling with Adaptive Dynamic Grouping », *Electronics*, vol. 14, nº 13, p. 2586, juin 2025, doi: 10.3390/electronics14132586.

[25] D. Mei, W. Su, Y. Shi, et Y. Jin, « Genetic Ant Colony Algorithm and Its Design and Research in Cloud Computing Platform Resource Scheduling », *Scalable Comput. Pract. Exp.*, vol. 26, nº 4, juin 2025, doi: 10.12694/scpe.v26i4.4612.

[26] M. Yousef, N. Hassouneh, et A. Sharieh, « Hybrid Metaheuristic-Based Cloud Task Scheduling Using Genetic Algorithm and Salp Swarm Algorithm », in *2025 International Conference on New Trends in Computing Sciences (ICTCS)*, Amman, Jordan: IEEE, avr. 2025, p. 406-412. doi: 10.1109/ICTCS65341.2025.10989323.

[27] R. Kumar, N. Singhal, et A. Chhabra, « Hybrid Optimization algorithm with the combination of PSO and genetic algorithm for task scheduling in cloud computing », *E-Learn. Digit. Media*, p. 20427530251331082, avr. 2025, doi: 10.1177/20427530251331082.

[28] X. Fu, Y. Sun, H. Wang, et H. Li, « Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm », *Clust. Comput.*, vol. 26, nº 5, p. 2479-2488, oct. 2023, doi: 10.1007/s10586-020-03221-z.

[29] A. M. Manasrah et H. Ba Ali, « Workflow Scheduling Using Hybrid GA-PSO Algorithm in Cloud Computing », *Wirel. Commun. Mob. Comput.*, vol. 2018, nº 1, p. 1934784, janv. 2018, doi: 10.1155/2018/1934784.

[30] S. Khaleelahmed *et al.*, « Task scheduling algorithm using grey wolf optimization technique in cloud computing environment », *Bull. Electr. Eng. Inform.*, vol. 14, nº 4, p. 2762-2771, août 2025, doi: 10.11591/eei.v14i4.7695.

[31] S. Bansal, B. S. Singla, et H. Aggarwal, « Workflow task scheduling in a cloud-fog environment: a hybrid PSO-GOA approach », *Int. J. Syst. Assur. Eng. Manag.*, janv. 2025, doi: 10.1007/s13198-024-02638-8.
[32] M. Bouqaffa et S. El Kafhali, « A Review of Hybrid Metaheuristic Algorithms for Workflow Scheduling in Cloud Computing », in *Advances in Intelligent Systems and Digital Applications*, vol. 1486, N. Gherabi, J. Kacprzyk, et S. Arezki, Éd., in Lecture Notes in Networks and Systems, vol. 1486. , Cham: Springer Nature Switzerland, 2025, p. 327-338. doi: 10.1007/978-3-031-95330-9_34.
[33] E. Saeedizade et M. Ashtiani, « Scientific workflow scheduling algorithms in cloud environments: a comprehensive taxonomy, survey, and future directions », *J. Sched.*, vol. 28, nº 1, p. 1-63, févr. 2025, doi: 10.1007/s10951-024-00820-1.
[34] F. Kaplan et A. Babalik, « Performance Analysis of Cloud Computing Task Scheduling Using Metaheuristic Algorithms in DDoS and Normal Environments », *Electronics*, vol. 14, nº 10, p. 1988, mai 2025, doi: 10.3390/electronics14101988.
[35] H. Casanova, A. Giersch, A. Legrand, M. Quinson, et F. Suter, « Versatile, scalable, and accurate simulation of distributed applications and platforms », *J. Parallel Distrib. Comput.*, vol. 74, nº 10, p. 2899-2917, oct. 2014, doi: 10.1016/j.jpdc.2014.06.008.