SGWO: A Modified Grey Wolf Optimizer for Fog–Cloud Workflow Scheduling with Emphasis on Makespan and Cost Efficiency

Raouf Belmahdi[1*], Djamila Mechta[1], Saad Harous [2]
[1]LRSD Laboratory, Department of Computer Science, Faculty of Sciences, Setif 1 University Ferhat Abbas, Setif, Algeria
[2]Department of Computer Science, College of Computing and Informatics, University of Sharjah, Sharjah, UAE
E-mail: raouf.belmahdi@univ-setif.dz, mechtadjamila@univ-setif.dz, harous@sharjah.ac.ae

Keywords: IoT applications, fog-cloud infrastructure, task scheduling, grey wolf optimizer, makespan, cost efficiency

Received: December 10, 2024

The rapid deployment of IoT applications creates several challenges. To address these challenges, a new paradigm called Fog Computing extends Cloud Computing services to the network edge, which reduces latency, conserves energy, and saves bandwidth. One of the major issues in enhancing the IoT application's performance is the tasks' scheduling, specifically, how to allocate them to different resources distributed across Fog-Cloud Computing layers. In this paper, a scheduling algorithm named SGWO is proposed, which adapts the standard Grey Wolf Optimizer (GWO) algorithm by including an improvement aimed at optimizing the balance between the exploration and exploitation phases. This adaptation enables the efficient scheduling of dependent tasks, modeled as general workflows, by allocating them efficiently across a hybrid topology composed of fog and cloud nodes with the dual objectives of minimizing makespan and reducing resource utilization costs. A series of experiments were conducted in a static simulation environment using diverse workflow datasets under various topology configurations. The results demonstrate that SGWO converges quickly and delivers better performance in terms of both makespan and cost, achieving a fitness value higher than those obtained by PSO, SQGA, GA, MHEFT, and FCFS algorithms by 43.3%, 50.8%, 55.4%, 54.5%, and 57.7%, respectively, thereby confirming its effectiveness in scheduling for this type of infrastructure

Povzetek: Članek predstavi SGWO, izboljšan Grey Wolf Optimizer za razporejanje odvisnih nalog v fog–cloud okolju, ki z boljšim ravnotežjem med raziskovanjem in izkoriščanjem učinkoviteje minimizira stroške rabe virov kot primerjalni algoritmi.

1 Introduction

The Internet of Things (IoT) provides numerous benefits in several domains, such as automation, real-time decision making, and monitoring, and improves efficiency and productivity. These opportunities offered by IoT applications have generated rapid developments that led to extensive integration. Statistics show that by 2025 almost 75 billion IoT devices will be deployed in the world [13].

The high increase in the volume of data generated over communication networks has created several challenges in transmitting the data generated by these applications to remote Cloud data centers, to be processed and analyzed [20][45].

Several research works have mentioned some challenges related to the integration of IoT with Cloud Computing [6][10][23] such as IoT application presents a source of Big Data, hence the need for storage on the Cloud and complex data analysis [47]. A huge amount of data must be transferred, which requires high bandwidth [6][9] and increases latency [11], the infrastructure scalability is needed to support the interaction of applications with several deployed devices [6].

To address these issues and enhance the performance of this kind of application, new solutions were proposed in research works, among these solutions Fog Computing, which represents a new paradigm, is first proposed by Cisco in 2012, in order to overcome the limits of Cloud Computing for processing data produced by IoT applications in real-time.

Several definitions have been proposed such as Fog computing, according to Cisco, is a paradigm that brings Cloud computing and their services closer to the network's edge [1], and according to the OpenFog Consortium, the Fog Computing is a: "horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum" [12]. This new paradigm has offered a number of benefits to IoT applications, such as [21][3]: real-time communications, low latency, saving bandwidth and low energy consumption, etc.

1.1 The role of scheduling in fog computing

The amount of time it takes for tasks to be completed when an end user requests them. The use of Fog-Cloud comput-

ing's resources plays an essential role in enhancing the performance of this kind of application. Scheduling is one of the main strategies used to optimize different objectives and meet the different QoS requirements of IoT applications.

Scheduling tasks is a challenging problem in Fog computing [26][2], which represents an NP-hard problem [16][43][14], impacting the optimization of application performance. The hierarchical architecture of Fog computing allows resource distribution across multiple layers, contributing to the complexity of this issue. To improve performance based on these types of application requirements, we require efficient scheduling algorithms that are grounded in the theory of optimization [27]. These algorithms ensure the proper assignment of tasks from an application to available infrastructure resources [15], to meet the end-user requests effectively.

In this work, we introduce a metaheuristic scheduling algorithm, named SGWO, which is based on the modified GWO algorithm adapted to address the scheduling problem. The proposed algorithm SGWO, enhances the scheduling of workflow-type tasks deployed on Fog-Cloud computing infrastructure, aiming to optimize total execution time or makespan and the overall execution cost associated with resource utilization. In addition, the proposed scheduling method takes into account the trade-off balance between the makespan and the total cost. Furthermore, we conducted a series of experiments to evaluate the performance of the proposed SGWO scheduling algorithm against existing algorithms, including PSO, GA, SQGA, MHEFT and FCFS.

The remainder of the paper is structured as follows. In Section 2, we describe the system model and the relationships between its layers. In Section 3, we define the formulation of the proposed scheduling problem and the objective function that we seek to optimize. In Section 4, we discuss the new scheduling approach and describe the proposed SGWO algorithm. We present a series of experiments and evaluations of the SGWO algorithm in Section 5, and a discussion of the results obtained. Finally, we conclude this work in Section 6.

1.2 Related work

One of the main challenges in improving the performance of applications built on the Fog-Cloud computing infrastructure is scheduling. In the literature, many research attempts have been made [28][18] to solve this problem. Most of these studies use heuristics and metaheuristics algorithms to address this type of problem [7][19][22].

Workflow scheduling often relies on heuristic strategies, with HEFT (Heterogeneous Earliest Finish Time) standing out as a widely recognized solution, this algorithm consists of two stages: the task prioritizing phase and the processor selection phase, with the aim of minimizing the earliest finish time (EFT). The classic version of the HEFT algorithm aims to minimize the total execution time only. A new algorithm called EHEFT-R (Enhanced Heterogeneous Earliest Finish Time based on Rule) is proposed by the authors of article [48], which addresses the multi-objective scheduling problem in cloud computing by integrating three criteria (makespan, energy consumption, QoS) in the form of a weighted sum to calculate the rank function of phase one. The experimental results show that this algorithm outperforms existing approaches such as HEFT, NSGA-II, QL-HEFT, etc.

One of the most used scheduling algorithms is the genetic algorithm [7][25]. A scheduling model, called EASM for cloud computing [29], aimed at optimizing energy consumption, execution time and Service Level Agreement (SLA) violations. It operates in two phases, a preprocessing phase that classifies tasks according to their delay constraints, followed by an optimization phase based on an adaptive genetic algorithm. Experimental results show that EASM outperforms Round Robin and a naive version of the genetic algorithm that does not take deadlines into account.

A task scheduling algorithm called CAG that uses the genetic algorithm is proposed in [34]. It is executed at the central scheduling node of the fog layer and decides whether to perform a task in the cloud or fog layer. This aims to lower execution costs and improve the meeting of deadlines for task responses. The results indicate that the proposed algorithm outperforms Round Robin and Minimum Response Time Algorithms.

A hybrid strategy based on prioritizing tasks and a genetic algorithm (PGA) is proposed in [17]. It chooses the best node to execute a task, which optimizes both the amount of energy used and the total execution time. A novel scheduling algorithm for workflow-type tasks called SQGA, using the quantum approach to improve the effectiveness of the genetic algorithm is introduced in [8]. It reduces the overall execution time of IoT applications. The results show that the proposed algorithm outperforms traditional GA and FCFS algorithms.

Other research studies utilize scheduling algorithms derived from swarm intelligence techniques, notably the Particle Swarm Optimization (PSO) approach, which has been widely used for diverse optimization problems [25]. The authors of [36] developed an expanded particle swarm optimization (EPSO) approach with an extra gradient method, to reduce execution time and increase resource effectiveness. The results show that the EPSO method proposed is more effective than the conventional PSO algorithm. Some research works focus on the hybridization of the PSO algorithm, in the work [44], the authors develop a hybrid algorithm between GA and PSO named GA-PSO, to reduce an application's makespan and energy usage in the fog computing environment. The outcomes show that the suggested hybrid algorithm GA-PSO outperforms the two traditional algorithms GA and PSO. In [40], the authors also develop a hybrid GA-PSO algorithm for dynamic resource allocation aimed at minimizing the makespan. Their results demonstrate that the hybrid algorithm is more effective compared to standalone GA and PSO algorithms.

According to the studies listed in Table 1.2, most of the scheduling techniques used in fog and cloud comput-

Table 1: Summary of scheduling approaches in existing studies

Reference	Infrastructure	Workload Type	Objective Function	Optimization Strategy	Key Results
[48]	Cloud	Workflow	Makespan, Energy, QoS	Enhances HEFT by adding rule-based task prioritization, a weighted multi-objective rank function, and improved processor selection	EHEFT-R outperforms HEFT, NSGA-II, and QL-HEFT across multiple performance metrics
[29]	Cloud	Workflow	Execution Time, Energy, SLA violations	An EASM algorithm based on an adaptive GA is proposed to improve task scheduling by combining a prior classification according to deadlines.	EASM outperforms Round-Robin and naive GA in terms of timeliness and efficiency.
[34]	Fog/Cloud	Bag-of-Tasks	Cost, Meeting of Deadlines	A CAG approach, based on a cost-sensitive GA, optimizes task assignment between Fog and Cloud	CAG outperforms Round Robin and minimum response time algorithms.
[17]	Fog/Cloud	Bag-of-Tasks	Execution Time,	A hybrid PGA strategy combines task prioritization and a genetic algorithm	PGA outperforms other bio-inspired algorithms, including AMO and Po2c.
[8]	Fog/Cloud	Workflow	Makespan	A proposed SQGA algorithm based on a quantum approach, improves the efficiency of conventional genetic algorithms.	SQGA outperforms traditional GA and FCFS.
[36]	Fog/Cloud	Bag-of-Tasks	Execution Time, Resource Effectiveness	A hybrid EPSO method combining PSO optimization with proximal gradient methods is proposed to optimize scheduling.	EPSO outperforms the GA and PSO algorithms compared (TCaS, Ideal PSO, Modified PSO).
[44]	Fog	Bag-of-Tasks	Makespan, Energy	A GA-PSO approach combines the properties of genetic and particle swarm optimization algorithms for hybrid optimization.	Hybrid algorithm outperforms its two basic components, GA and PSO
[40]	Manufacturing System	Workflow	Makespan	A hybrid GA-PSO approach is proposed for scheduling in manufacturing systems with dynamic resources and constrained sequences.	Hybrid algorithm outperforms basic GA and PSO algorithms

ing environments are based on metaheuristic algorithms, notably genetic algorithms and particle swarm optimization [7]. However, these methods have certain limitations [31] [39] [30], such as a frequently long convergence time, difficulty in balancing exploration and exploitation, and an increased risk of convergence to a local optimum.

A recent metaheuristic algorithm called Grey Wolf Optimizer (GWO) [31], is proposed as an alternative to address these limitations, which imitates the hunting mechanism and leadership hierarchy of Grey wolves in nature. This algorithm established a special parameter that enables the balancing of exploration and exploitation. Benchmark results show that the GWO algorithm performs better than the popular PSO algorithm [24], Differential Evolution (DE) [42], Gravitational Search Algorithm (GSA) [37], etc. In [32], the novel mGWO method, which is a modification of the GWO algorithm, employs a new exponential decay function, to optimize the algorithm's convergence while maintaining a good balance between exploration and exploitation.

According to this, we propose a new GWO-based solution called SGWO, adapted to the context of scheduling in fog and cloud computing environments. The aim is to leverage the advantages of this algorithm and evaluate its performance in comparison with the scheduling methods most widely used in this field. Moreover, the SGWO algorithm considers dependencies between tasks, thereby adding complexity to the scheduling process.

2 System architecture

The distributed paradigm known as Fog computing, organized as a hierarchical architecture, extends cloud computing capabilities to the edge of the network [21]. This type of architecture, illustrated in Figure 1, demonstrates the overall architecture and decomposition of the scheduling system. Figure 1 highlights the main components and the various interactions between them, providing a clear overview of the system's structure. Each part and its role are detailed further in the following sections.

In the proposed architecture, the different components are distributed over three main layers, which are fully connected, with each layer having its own characteristics and functionalities.

The first layer, the IoT layer, consists of a variety of IoT devices from diverse applications, such as connected cars, medical tools, and industrial sensors. These devices are equipped with sensors and actuators to collect data from the environment and perform pre-processing operations such as filtering and aggregation. Although this layer has limited resources and cannot meet the performance requirements of IoT applications on its own, it operates within a Fog-Cloud Computing architecture. The data collected by IoT devices is sent as queries to the upper layer, known as the Fog layer, for efficient processing.

In the proposed architecture, the second layer, the Fog layer, located above the IoT layer, has more resources and is responsible for managing and executing requests from the IoT layer locally and efficiently, to improve the execution time and reduce the cost of IoT applications.

The Fog layer consists of a set of fog nodes spread over the entire layer and a Fog broker. Fog nodes are physical devices interconnected by the network and equipped with computing, storage, and networking capabilities. They are usually more efficient than IoT devices and their role is to execute the different tasks of a workflow.

The Fog Broker serves as the main infrastructure node, located in the fog layer, and is responsible for the control and orchestration of all infrastructure operations. It is comprised of four primary components:

- Controller: This component is responsible for managing communication between the system's different parts. It breaks down requests of IoT devices into individual tasks of the workflow, and then collects and returns the results to the IoT devices. Additionally, it initiates and coordinates the scheduling of these tasks.

- Resources Monitor: The role of this component is to collect and monitor the status of various resources available on the Fog-Cloud computing infrastructure, such as fog nodes, cloud nodes, and their resources (CPU, storage, memory, network bandwidth, etc.), as well as to inform other system components of the status of available resources.

- Scheduler: This component is primarily responsible for determining which infrastructure node (fog node or cloud node) will execute each workflow task. Scheduling decisions are made based on available resources and an optimization algorithm designed to enhance application performance. This scheduler incorporates the SGWO algorithm, which effectively reduces both execution time and cost for the entire workflow. The output of the SGWO algorithm specifies which tasks should run on which nodes within the infrastructure.

- Task Dispatcher: The role of this component is to transmit and allocate the various workflow tasks to the different nodes of the infrastructure, which are assigned by the scheduler component.

The various resources of this layer are more efficient compared to the resources of the IoT layer, and the operational cost of most of these resources is free.

The third layer, the Cloud layer, offers powerful nodes deployed in remote data centers, generally connected via a WAN network. This layer provides extensive computation, storage, and memory capabilities, enabling the analysis of large amounts of data and the processing of complex tasks. In this case, its role is to execute workflow tasks assigned by the Scheduler component, knowing that the use of the resources of this layer generates costs.

Regarding the interaction between the various layers and components of the system, IoT devices first collect data through embedded sensors, then send it to the nearest fog node, which will redirect this request to the Broker. When the Broker receives the request, it calls on the Controller sub-component, which subdivides the request into tasks corresponding to a workflow, and then sends these generated tasks to the Scheduler for scheduling.

The Scheduler first consults the Resources Monitor subcomponent to receive information on the status of the resources available in the infrastructure, then uses the proposed SGWO scheduling algorithm to optimize the execution time and operating costs of the infrastructure resources. At the end of this phase, we obtain a list of dependent tasks assigned to the different nodes in the infrastructure, which will be forwarded to the Task Dispatcher sub-component. The Task Dispatcher will distribute the tasks to the appropriate nodes (Fog nodes or Cloud nodes) according to the assignment list provided by the Scheduler. Once tasks are running, the Task Dispatcher monitors their progress and communicates with the Scheduler to update resource status and ensure that tasks are executed efficiently.

After the various workflow tasks have been executed by the infrastructure nodes, the results are forwarded to the Controller sub-component, which returns the results to the IoT devices.

3 Problem formulation

IoT devices send many requests to the fog layer. To process these requests in an efficient manner, they are broken down into tasks, to be performed by the Fog-Cloud infrastructure's nodes. In this work, we focus on applications whose workloads are structured as workflows, where tasks have dependencies. We consider synthetic workflow types which include common execution patterns such as sequential and parallel, applicable to many IoT application areas. They are represented as a Directed Acyclic Graph or DAG, as illustrated in Figure 2, with the notation $G = (T, E)$ [4], where the vertices T define the list of ordered tasks and the edges E indicate the interdependence between the various tasks. Every edge e_{ij} connecting two tasks has a weight w, which represents the amount of data that must be transmitted from the parent task T_i to the child task T_j.

All the notations used in the mathematical formulation of the problem are summarized in Table 3.

We consider T the set of dependent tasks that constitute the Workflow G, formulated by Equation 1:

$$T = \{T_1, T_2, T_3, ..., T_n\} \tag{1}$$

Where each task T_i has several properties, such as the computational workload is indicated by the number of instructions for each task denoted by $CW\ (T_i)$, the memory size required for the task denoted by $SM(T_i)$.

The set $E \subseteq T \times T$ defines the precedence relationships between the different tasks in the G workflow, each element

Table 2: Summary of the mathematical notations used in the problem formulation

Notation	Description
T_i	Task i
N_j	Node j
$CW(T_i)$	Computational workload of task T_i
$SM(T_i)$	Size of memory required by task T_i
$SO(T_i)$	Output data size of task T_i
$SI(T_i)$	Input data size of task T_i
CPU_F(N_j)	CPU operating frequency of node N_j
$MS(N_j)$	Memory size of node N_j
$SC(N_j)$	Storage capacity of node N_j
$TR(L_k)$	Data transfer rate of link L_k
$ExT(T_i, N_j)$	Execution time of task T_i on node N_j
$SD(e_{ij})$	Size of data transmitted through edge e_{ij}
$OS(T_i)$	Output file size generated by the source task of edge e_{ij}
e_{ij}	Communication edge from task T_i to T_j
τ_i	Indicates task T_i is assigned to node N_i
$TrT(e_{ij})$	Data transmission time across edge e_{ij}
$Avail(N_j)$	Availability time of node N_j
$pred(T_i)$	Predecessors of task T_i
$ST(T_i)$	Start time of task T_i
$FT(T_i)$	Finish time of task T_i
U_{cp}	Processing cost per unit of time
$C_{cp}(T_i, N_j)$	Monetary cost of computing task T_i on node N_j
U_{mm}	Memory cost per data unit
$MemSize(T_i, N_j)$	Size of memory used by task T_i on node N_j
$C_{mm}(T_i, N_j)$	Monetary cost of memory use by task T_i on node N_j
$StorSize(T_i, N_j)$	Size of data storage used by task T_i on node N_j
U_{st}	Storage cost per data unit
$C_{st}(T_i, N_j)$	Monetary cost of using the data storage for task T_i on node N_j
U_{cm}	Data transfer cost per unit
$C_{cm}(T_i, N_j)$	Monetary cost of data transfer for task T_i
$Cost(T_i, N_j)$	Total cost of executing task T_i on node N_j
F	Objective function

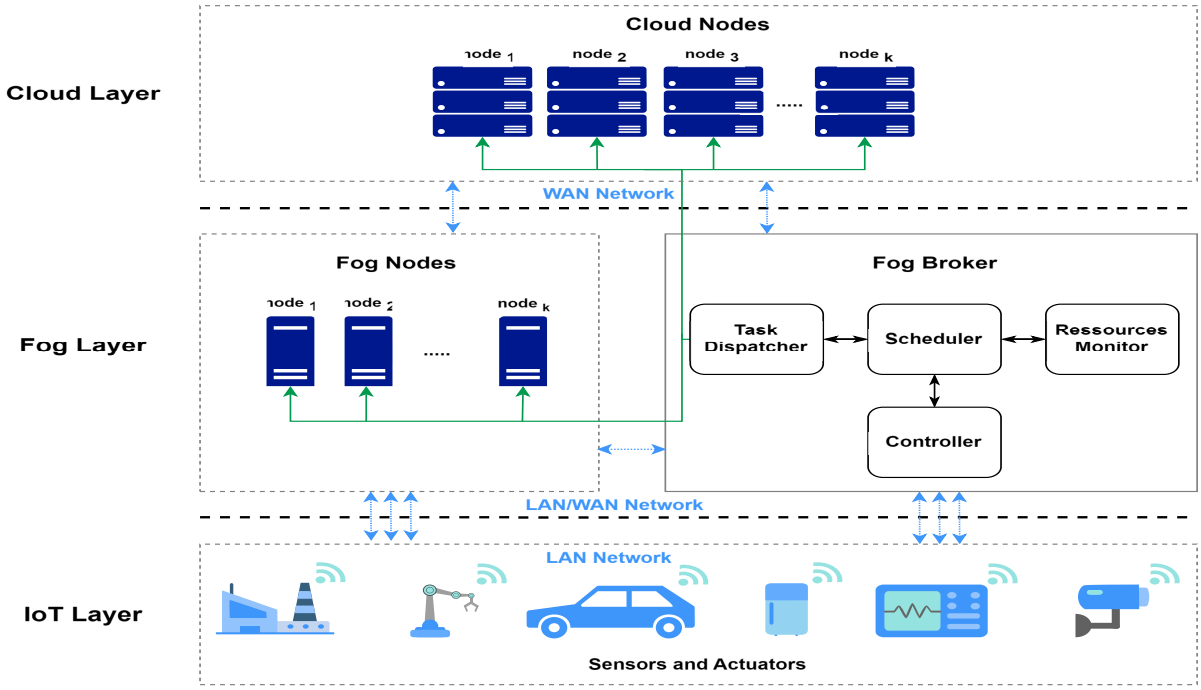


Figure 1: Schematic of our system architecture and scheduling approach

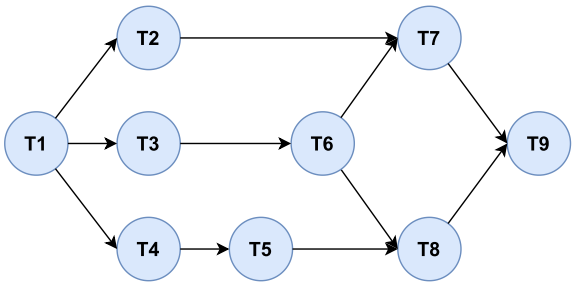


Figure 2: Structure of a synthetic workflow model

$e_{ij} = (t_i, t_j) \in E$ represents a dependency between two tasks, meaning that task t_j cannot start until task t_i is fully completed, this dependency may also involve data transfer from source task t_i to target task t_j.

Each task is assigned to a node for execution, and once completed, the node generates output data, including task results and associated application data. This data is subsequently transmitted to the next workflow tasks, with the output data size being quantified by $SO(T_i)$. Upon receipt by the destination node, the transmitted data is allocated to the relevant task, serving as input data for the current task, and quantified by $SI(T_i)$.

The broker's Task Dispatcher sub-component distributes the workflow-dependent tasks to the various nodes positioned within the Fog-Cloud infrastructure. The list of infrastructure nodes is denoted by the set N, where $N = \{N_{cloud} \cup N_{fog}\}$, and N formulated as follows:

$$N = \{N_1, N_2, N_3, ..., N_m\} \tag{2}$$

Each infrastructure node N_j characterized by: a CPU Operating Frequency CPU_F (N_j), the memory size $MS\,(N_j)$, the storage capacity $SC\,(N_j)$.

The nodes are connected by a network, the connection can be a local connection or a WAN connection, and each connection link is characterized by a data transfer rate $TR(L_k)$.

3.1 Makespan model

The makespan is the workflow's overall execution time. It contains two parts: the total time taken for all workflow tasks to complete and the total time taken for data to transfer between tasks [49]. When a task T_i of the workflow is allocated to a node N_j (part of the Fog-Cloud infrastructure), it will be executed by N_j. The execution time, $ExT\,(T_i, N_j)$, is computed using Equation 3.

$$ExT\,(T_i, N_j) = \frac{CW\,(T_i)}{CPU\_F\,(N_j)} \tag{3}$$

It is necessary to communicate data across the various tasks for the infrastructure's nodes to fully execute the workflow, which consequently results in a data transmission time denoted by $TrT\,(e_{ij})$. It is computed using Equation 4.

$$TrT\,(e_{ij}) = \begin{cases} \sum_k \frac{SD(e_{ij})}{TR(L_k)} & \text{if} \quad \tau_i \neq \tau_j \\ 0 & \text{if} \quad \tau_i = \tau_j, \end{cases} \tag{4}$$

$TrT(e_{ij})$ represents the sum of the transmission times between the different links L_k which connect the nodes allocating the source task T_i and the destination task T_j of the workflow. Where e_{ij} indicates the edge between two successive tasks T_i and T_j, $SD(e_{ij})$ represents the size of data transmitted through an edge e_{ij}, and $SD(e_{ij})$ equal to the size of the output file $OS(T_i)$ generated by the source task of the edge e_{ij}, $TR(L_k)$ represents data transfer rate of an k-link L_k that related between node source and node destination, τ_i indicates that task T_i has been assigned to node N_i, same for τ_j. In case two related tasks are assigned to the same node ($\tau_i = \tau_j$), the data transmission time $TrT\ (e_{ij})$ is negligible.

In the next step, we determined the start time $ST\ (T_i)$ and the finish time $FT\ (T_i)$ values for each task in the workflow to calculate the makespan of the workflow.

Firstly, to calculate the start time $ST\ (T_i)$ we apply Equation 5.

$$ST\ (T_i) = max\left\{Avail\ (N_j)\ , max\left\{FT\ (T_p) + TrT\ (e_{ij})\right\}\right\} \quad ,T_p \in pred(T_i) \tag{5}$$

Where $Avail\ (N_j)$ denotes the time of availability of the node N_j, $FT\ (T_p)$ denotes the finish time of a predecessor task T_p, where task T_i includes T_p on its list of predecessors denoted by $pred(T_i)$.

Secondly, we calculate the finish time $FT\ (T_i)$ using Equation 6.

$$FT\ (T_i) = ST\ (T_i) + Ext\ (T_i, N_j) \tag{6}$$

The makespan value of the workflow is computed using the following equation.

$$Makespan = max\left\{FT(T_i)\right\} \tag{7}$$

3.2 Cost model

During the execution of workflows by the Fog-Cloud infrastructure's nodes, monetary costs are generated by the use of the infrastructure resources, such as computation, memory, storage, bandwidth, etc.

To calculate the cost of the resources used by a task T_i, which is processed by a node N_j, we use Equation 8.

$$Cost(T_i, N_j) = \begin{cases} C_{cp}(T_i, N_j) + C_{mm}(T_i, N_j) \\ +C_{st}(T_i, N_j) + C_{cm}(T_i, N_j) \\ if N_j \in N_{cloud} \\ C_{cm}(T_i, N_j)\ if N_j \in N_{fog} \end{cases} \tag{8}$$

$C_{cp}(T_i, N_j)$ represents the monetary cost of computing the task T_i by node N_j, is calculated using Equation 9.

$$C_{cp}(T_i, N_j) = U_{cp} \times ExT(T_i, N_j) \tag{9}$$

Where U_{cp} represents the processing cost per unit of time. $C_{mm}(T_i, N_j)$ represents the monetary cost of using memory of size $MemSize(T_i, N_j)$ by a task T_i on node N_j, and calculated using Equation 10.

$$C_{mm}(T_i, N_j) = U_{mm} \times MemSize(T_i, N_j) \tag{10}$$

Where U_{mm} represents the memory cost per data unit. $C_{st}(T_i, N_j)$ represents the monetary cost of using the data storage of size $StorSize(T_i, N_j)$ by a task T_i on the node N_j, and calculated by Equation 11.

$$C_{st}(T_i, N_j) = U_{st} \times StorSize(T_i, N_j) \tag{11}$$

Where U_{st} represents the storage cost per unit of data. $C_{cm}(T_i, N_j)$ represents the monetary cost of communication for the amount of data transferred between nodes and is calculated by Equation 12.

$$C_{cm}(T_i, N_j) = \begin{cases} \sum_{T_p}(U_{cm} \times SD(e_{pi})) \\ \text{if} \quad \tau_i \neq \tau_j \\ 0 \text{ if} \quad \tau_i = \tau_j \end{cases} \quad ,T_p \in pred(T_i) \tag{12}$$

Where U_{cm} represents data transfer cost per data unit, and the cost is considered if two consecutive tasks are allocated to different nodes ($\tau_i \neq \tau_j$), otherwise, the communication cost is null.

The workflow's total monetary cost is determined using the Equation 13.

$$TotalCost = \sum Cost(T_i, N_j) \tag{13}$$

3.3 The objective function

To minimize the objective function, and optimize the trade-off between makespan and the total cost of the workflow, we provide Equation 14.

$$F = \alpha \times Makespan + \beta \times TotalCost \tag{14}$$

Subject to:

$$1 \leq \tau_i \leq m \quad ,\forall i \in \{1, \ldots, n\} \tag{15}$$

$$\sum_{j=1}^{m} \text{Active}_j \leq M_{\max} \quad , \quad \text{Active}_j \in \{0, 1\} \tag{16}$$

$$\sum_{i:\, \tau_i = k} R_i \leq C_k, \quad \forall k \in \{1, \ldots, m\} \tag{17}$$

Where $\alpha \in [0, 1]$ and $\beta = 1 - \alpha$, represent balance coefficients between the Makespan and the Total Cost.

Constraint 15 ensures that each T_i task is assigned to a single node among the available m nodes. Constraint 16, defined by the binary variable $Active_j \in \{0, 1\}$, specifies the activation state of the node j. The constraint M_{max} imposes an upper bound on the total number of nodes that

can be activated, limiting their selection to a maximum size subset of the m nodes available in the infrastructure. Constraint 17, ensures that the total resource requirements (RAM, storage, etc.) for the tasks T_i allocated to node K do not exceed the available capacity of that node, denoted C_K.

Numerous scheduling approaches based on heuristics and meta-heuristics algorithms such as HEFT, PSO and GA have been proposed to solve this type of problem, in an environment combining cloud and fog computing. In this context, this study aims to answer the following question. How effective is the proposed algorithm SGWO in optimizing workflow scheduling within a fog-cloud computing environment, by minimizing both makespan and resource utilization cost, compared to PSO, GA, SQGA, MHEFT and FCFS algorithms?

4 Scheduling approach

We present a scheduling algorithm that uses Grey Wolf optimization GWO [31]. This approach is based on swarm intelligence to imitate the natural hierarchy and hunting behaviors of gray wolves in nature, they typically live in a pack of 5 to 12 individuals, in a rigid social hierarchy of dominance, as shown in Figure 3.

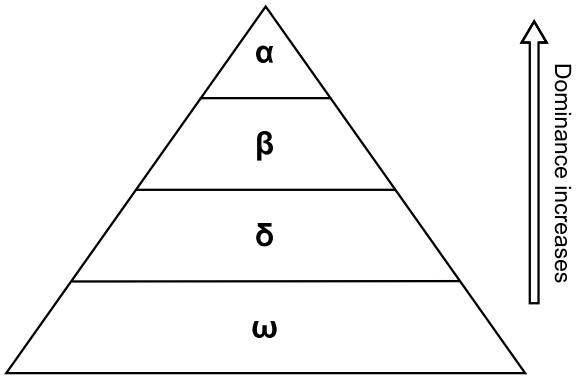


Figure 3: Diagram illustrating the dominance hierarchy structure in gray wolf packs

Figure 3 illustrates the hierarchical organization and levels of social dominance within gray wolf packs. This representation clearly shows the pyramidal structure of the social relationships within these groups.

The fittest solution called the alpha (α), is considered as the pack's leader, which presents the most dominant member. The second best solution called beta (β), is the inferior Wolf which supports the alpha in making decisions. The delta (δ) is ranked third best members, this category of wolves has to submit to α and β members, and they control the omega wolves. The remaining members are categorized as omega (ω), ranked as the lowest in the pack, and they must always yield to any other dominant wolf.

Group hunting is another significant social behavior included in the social hierarchy [33]. It is composed of 3 main parts:

- Tracking, pursuing, and approaching towards the prey.
- Encircling and harassing the prey until it stops moving.
- Attacking the prey.

4.1 GWO algorithm mathematical model

The following mathematical models 18 and 19, describe the social behavior of gray wolves as they circle their prey during the hunting phase.

$$\vec{D} = \left| \vec{C} \cdot \vec{X_p}(t) - \vec{X}(t) \right| \tag{18}$$

$$\vec{X}(t+1) = \vec{X_p}(t) - \vec{A} \cdot \vec{D} \tag{19}$$

Where the distance vector of the prey's location is represented by $\vec{D}$, t denotes the current iteration, $\vec{X}$ indicates the gray wolf's position vector, and $\vec{X_p}$ indicates the position vector of the prey.

Equations 20-21, calculate the coefficient vectors $\vec{A}$ and $\vec{C}$.

$$\vec{A} = 2\vec{a} \cdot \vec{r_1} - \vec{a} \tag{20}$$

$$\vec{C} = 2\vec{r_2} \tag{21}$$

Where the random vectors r_1 and r_2 value between $[0, 1]$, and the elements of $\vec{a}$ in the standard algorithm GWO [31] throughout iterations, is linearly decreased from 2 to 0. In this work, we use a modified grey wolf optimizer named mGWO [32], which represents an improved version of the GWO algorithm. This algorithm uses an exponential formula for balancing between exploration and exploitation and for the decay of $\vec{a}$ across iterations, where $\vec{a}$ is calculated using Equation 22.

$$\vec{a} = 2\left(1 - \frac{t^2}{T^2}\right) \tag{22}$$

Where the parameter t denotes the current iteration and the parameter T denotes the maximum number of iterations.

In the hunting stage, the process is guided by α, β and δ, where α presents the best candidate solution than other wolves, and these three kind of agents have better awareness of the potential location prey, the rest of the wolves include ω, updated their positions by the best research agent's position, this process is calculated using the formulas 23, 24 and 25 below.

$$\begin{aligned} \vec{D}_\alpha &= \left| C_1 \cdot X_\alpha - \vec{X} \right|, \\ \vec{D}_\beta &= \left| C_2 \cdot X_\beta - \vec{X} \right|, \\ \vec{D}_\delta &= \left| C_3 \cdot X_\delta - \vec{X} \right|, \end{aligned} \tag{23}$$

$$\begin{aligned}\vec{X_1} &= \vec{X_\alpha} - \vec{A_1} \cdot \vec{D_\alpha},\\ \vec{X_2} &= \vec{X_\beta} - \vec{A_2} \cdot \vec{D_\beta},\\ \vec{X_3} &= \vec{X_\delta} - \vec{A_3} \cdot \vec{D_\delta},\end{aligned} \tag{24}$$

$$\vec{X}(t+1) = \frac{\vec{X_1}(t) + \vec{X_2}(t) + \vec{X_3}(t)}{3} \tag{25}$$

Gray wolves finish their hunting stage by attacking their prey, according to the previous mathematical model, when the value of $\vec{a}$ is reduced to approach the prey. The fluctuation range of $\vec{A}$ is also reduced by $\vec{a}$, when the value of $\left|\vec{A}\right| < 1$ the wolves are closer to the prey, at this stage the wolves are forced to attack the prey.

4.2 The proposed scheduling algorithm SGWO

To enhance the scheduling in fog and cloud computing infrastructure, we proposed a new scheduling algorithm based on GWO called SGWO algorithm. This algorithm was adapted to improve the scheduling with the aim of reducing the makespan and the cost.

The proposed scheduling algorithm SGWO, is an adaptation of the standard GWO algorithm to address the scheduling problem in the Fog/Cloud environment. An encoding phase has been introduced to represent scheduling solutions in the form of task allocations to various infrastructure nodes. Furthermore, the orginal formula used in the standard GWO for calculating the control vector $\vec{a}$ has been modified, and we adopt a dynamic strategy based on formula 22 proposed in [32], with the aim of improving the balance between exploration and exploitation throughout the optimization process.

4.2.1 Encoding of the solution

Before using the GWO algorithm in scheduling, we must first go through the encoding phase, which allows us to present the GWO algorithm in a discrete representation as shown in Figure 4.

Wolf Position	T_1	T_2	T_3	·····	T_n
	N_3	N_8	N_2	·····	N_j

Figure 4: Illustrative diagram of the encoding of gray wolf positions in scheduling

Figure 4 shows the encoding of a gray wolf's position in the search space. This encoding is represented as a two-line table: the first line indicates the workflow tasks, denoted T_i, while the second line corresponds to the nodes, denoted N_j, to which these tasks have been assigned.

This algorithm is based on the wolf's position in relation to the prey's position. In this case, wolf's position represents the status of the workflow tasks allocation to the various Fog-Cloud infrastructure nodes. Each time a wolf moves in the search space, we obtain a new state of allocation, where this wolf movement is guided by the proposed scheduling algorithm.

4.2.2 SGWO algorithm

We present the proposed scheduling algorithm SGWO on Fog-Cloud computing infrastructure, as shown in Algorithm 1.

Algorithm 1 SGWO for Fog-Cloud Computing.

```
t ← 0
Initialize population()
Initialize parameters()
Allocate()
Evaluate fitness()
Get best agents()
while t < max_iterations do
    for each agent do
        Update position()
    end for
    Update parameters()
    Allocate()
    Evaluate fitness()
    Get best agents()
    t ← t + 1
end while
return best agent X_α
```

- In the first function "$Initialize\ population()$", we initialize the population of gray wolves named by search agents X_i where $i = 1, 2, ..., n$, by defining their positions, we apply the encoding phase mentioned above, see Figure 4, to adapt the GWO based algorithm to the nature of the scheduling problem.
- The second initialization function is "$Initialize\ parameters()$", which allows the calculation of the initial parameters of the algorithm, such as the parameter $\vec{a}$, the coefficient vectors $\vec{A}$ and $\vec{C}$ mentioned by Equations 20 and 22.
- The "$Allocate()$" function, is introduced as a key component in SGWO algorithm. It is responsible for allocating the different workflow tasks to the specific nodes of the infrastructure, according to the state of the positions of the wolves or search agents, see Figure 4.
- The "$Evaluate\ fitness()$" function, after allocating the different tasks to the specific nodes according to the state of the agent, we calculate for each search agent the fitness function mentioned by the objective function Equation 14.

  In the new proposed SGWO version, the "$Evaluate\ fitness()$" function has been adapted to the scheduling context in order to evaluate the

allocation of tasks to nodes. This evaluation takes into account several goal related criteria, including makespan, execution cost, and infrastructure constraints.

- The "$Get\ best\ agents()$" function, after calculating the objective function for each agent, this function sorts the results obtained and extracts and stores the best results. The first best search agent is named X_α, then the second is X_β and the third best search agent is X_δ.

- In the while-loop block, for each iteration, we repeat the same treatment. We start by updating the position of each agent using the function "$Update\ position()$", which calculates the new position of the agent using Equation 25. When we have the new positions, we should apply the "$Update\ parameters()$" function, which uses Equations 20 and 21 to update the parameters $\vec{A}$, $\vec{C}$, and $\vec{a}$. In the next step (line 12), the "$Allocate()$" function allows the different tasks of the workflow to be reallocated to different nodes of the infrastructure. The function "$Evaluate\ fitness()$" reevaluates the fitness functions for each gray wolf according to the new values. In the last step of this block, the new best search agents X_α ,X_β and X_δ are selected by "$Get\ best\ agents()$" function. Then we increase the number of iterations.

- Finlay, our algorithm returns the value of X_α, the best search agent, which presents the best solution found by the SGWO scheduling algorithm.

Regarding the complexity of SGWO algorithm compared to the standard GWO algorithm, the standard GWO algorithm has a complexity of $O(N \cdot D \cdot T)$ [46], where N represents the number of agents, D the problem dimension and T the number of iterations. SGWO being based on the standard GWO, is adapted to solve the scheduling problem, each agent represents the complete state of the schedule, and the $Allocate()$ function directly assigns tasks to their corresponding nodes, this operation is performed at each iteration for each agent and remains $O(N \cdot M)$. Therefore, the total complexity of SGWO remains $O(N \cdot D \cdot T)$, with a slight additional constant factor related to scheduling operations, allowing the performance of the standard GWO algorithm to be maintained without changing its computational complexity.

5 Experimentation and evaluation

In the following part, we want to evaluate and examine the proposed scheduling algorithm SGWO and compare its performance with other existing scheduling algorithms based on: the PSO Algorithm [24], GA Algorithm [41], SQGA Algorithm [8], MHEFT Algorithm and the classic First-Come First-Served algorithm FCFS.

Given that the MHEFT algorithm is based on EHEFT-R [48], and both are derived from the standard HEFT algorithm, it uses the weighted sum function of EHEFT-R, equivalent to the fitness function, to rank tasks in the first phase.

5.1 Experiment settings

The experimentation is based on the simulation of the scheduling of dependent tasks or workflow, deployed on the various heterogeneous nodes of the fog and cloud computing infrastructure. A static simulation environment is considered, in which all available infrastructure resources such as fog nodes, cloud nodes, and network connections, remain unchanged and fully available throughout the simulation scenarios.

In this simulation, the performance of the proposed scheduling algorithm SGWO, is evaluated against five other algorithms: PSO, GA, SQGA, MHEFT, and FCFS. Their performance is compared in terms of Makespan, Total Cost, and Energy Consumption. The experiments were carried out on a machine equipped with an Intel Core I3-12100 processor and 16 GB RAM. All scheduling algorithms (SGWO, PSO, GA, SQGA, MHEFT, FCFS) and associated simulations were developed in Java.

Initially, the parameters α and β of the fitness function are defined. Then, the performance of the six algorithms (SGWO, PSO, GA, SQGA, MHEFT, and FCFS) is evaluated, the utilization rate of cloud nodes in scheduling is studied, the impact of topology on scheduling is analyzed, and finally the energy consumption of the scheduling algorithms is examined.

In these simulation scenarios, the infrastructure nodes are heterogeneous. These nodes serve the purpose of handling user requests, which have been segmented into individual tasks. Furthermore, every task has the following characteristics: the number of instructions measured in a million (MI), and the size of the output and input data generated during the task execution and transfer via the network measure by (MB). According to the experimentation model, the use of cloud layer resources generates a paid cost, and the cost is expressed in cloud cost symbol (C$), knowing that the cost of processing per unit of time (CPU) measured by C$/h, the cost of memory per unit of data (RAM), the cost of storage per unit of data (Disk) and the data transfer cost per data unit measured by C$/G, as well as in the fog layer only external data transfer is paid [38].

In this simulation, we used synthetic workflows generated for the experiment, with the number of tasks varying between 100 and 1000 tasks per workflow .In a simulation scenario, the workflow tasks are deployed on the proposed infrastructure, which is made up of 10 nodes of the fog layer and 20 nodes of the cloud layer. Then the simulation process is executed 100 times, in each scenario, to calculate the mean value of the results. Table 5.1 summarizes the different parameters used in the experiment.

Table 3: Summary of experiment parameters

Entity	Parameter	Values
Cloud nodes	CPU rate	[3000-4000] MIPS
	RAM size	[4000-16000] MB
	Data Transfer Rate	1000 Mbps
	Processing Cost	0.4 C$/h
	Memory Cost	0.1 C$/GB
	Storage Cost	0.05 C$/GB
	Data Transfer Cost	0.2 C$/GB
	Number Nodes	20
	Active Power	[400-500] W
	Idle Power	100 W
Fog nodes	CPU rate	[1000-2000] MIPS
	RAM size	[500-2000] MB
	Data Transfer Rate	[100-1000] Mbps
	Data Transfer Cost	0.2 C$/GB
	Number Nodes	10
	Active Power	[10-30] W
	Idle Power	3 W
Tasks	Size	[100-1000] MI
	I/O file size	[10-1000] MB
Workflow	Number of Tasks/Workflow	100, 250, 500, 1000
Algorithms	SGWO number of wolves	5
	GA number of chromosomes	20
	SQGA Rotation angle δ	0.001π
	PSO number of particles	10

5.2 Sensitivity analysis and parameter calibration of the SGWO algorithm

In this section, we conduct a sensitivity study of the main parameters of the SGWO algorithm to analyze their impact on the performance of the scheduling algorithm. The aim is to calibrate these parameters and as much as possible to adapt them to the specificity of the scheduling problem.

- Defining the number of wolves: To determine the best number of wolves, we set the workflow size to the maximum value used in these experiments, which is equal to 1000 tasks. Then the number of wolves per pack is incrementally varied from 5 to 50 wolves. For each configuration, the value of the objective function is measured. The results are shown in Figure 5. According to Figure 5, the proposed SGWO algorithm achieves the best value of the fitness function when the number of wolves per pack is equal to 5.
- Balance between exploration and exploitation: the balance between exploration and exploitation is influenced by the parameter $\vec{a}$, represented by a vector in the SGWO algorithm> This parameter plays a key role in the convergence of the algorithm. In the proposed SGWO algorithm, Equation 22 is used to calculate the value of $\vec{a}$. This equation was proposed in the modified version of the GWO algorithm, called mGWO [32]. The experimental results show that Equation 22 performs better than the standard equation used to determine this parameter.
- Defining the number of iterations: Based on the results presented in the Section 6.6 on studying the convergence of different scheduling algorithms (see Figure 11), we observe that SGWO algorithm converges from the 50th iteration.

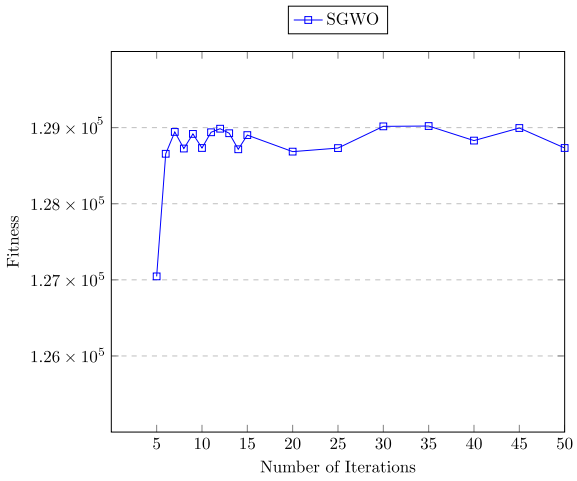


Figure 5: The impact of population size on the performance of the SGWO algorithm

5.3 Determination of Alpha and Beta parameter values

Before evaluating the proposed scheduling method and comparing it to other existing methods, we must first study and define α and β parameters' values of the proposed objective function defined in Equation 14. For this, we performed a series of simulations on the proposed SGWO scheduling algorithm by varying α and β parameters and setting up the workflow tasks' number at 1000. The results obtained are shown in Figure 6.

The points in Figure 6 represent the distribution of solutions where the fitness value is generated by the variation of the parameters α and β of the objective function. The variation in the color of the points represents the α value used in the experiments.

From the results, we observe that, in the case where α is close to 0, we observe that the Makespan value is close to its

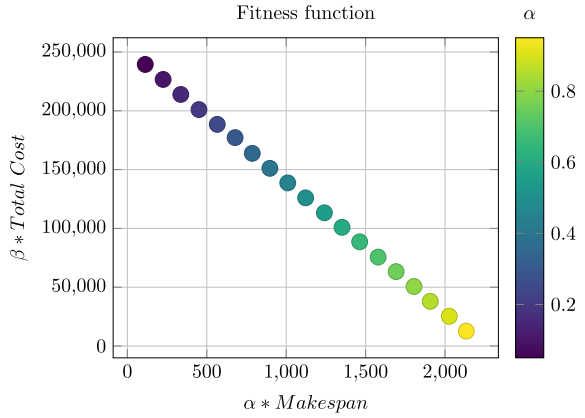


Figure 6: The effect of α and β parameters on the objective function of the schedule

minimum value and the Total Cost is close to its maximum value, is presented in Figure 6 by the solutions where the fitness value is in purple. When we increase the value of α the Makespan also increases, and the value of Total Cost decreases, which is presented in the chart by the variation of the fitness color from blue to green. Until the value of α is close to 1, we observe that the value of Makespan is close to its maximum value and the value of Total Cost is close to its minimum value, presented in the chart by the yellow fitness values.

This means that the values of Makespan and Total Cost are in the trade-off, and are affected by the following equation: $\beta = 1 - \alpha$, and we notice that the parameters α and β play a balancing role between Makespan and Total Cost in the proposed scheduling method. In the case where we need a scheduling that reduces the Makespan, we choose a reduced α value less than 0.5, and in the case where we need a scheduling that generates reduced Total Cost, we choose an α value greater than 0.5. For solutions balanced between Total Cost and Makespan, the α value should be equal to 0.5, presented in Figure 6 as the midpoint solution between the blue and green points.

After defining the roles of the parameters α and β to evaluate the proposed scheduling algorithm and to compare it with other algorithms (Figure 6), we observe that when α and β are both equal to 0.5, the objective function achieves a balance between Total Cost and Makespan. This result demonstrates that the proposed SGWO scheduling algorithm is efficient in terms of both metrics.

6 Results and discussions

In order to evaluate the effectiveness of the proposed SGWO scheduling algorithm, we performed a number of evaluations based on performance criteria. These evaluations determine the effectiveness of the proposed solution.

6.1 Evaluation of the makespan

In order to evaluate SGWO, and to compare its performance in terms of Makespan to PSO, GA, SQGA, MHEFT and FCFS, a series of simulations are carried out and the results obtained are presented in Figure 7.

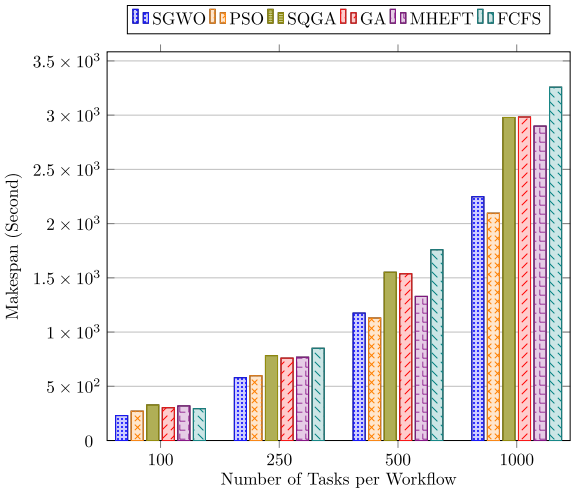


Figure 7: Comparison of the SGWO algorithm Makespan with those of PSO, GA, SQGA, MHEFT and FCFS

Based on the results shown in Figure 7 for various simulation cases, and after computing the average result values across test cases, we observe that SGWO achieves a minimum Makespan value, view Equation 7. It outperforms SQGA, GA, MHEFT, FCFS by 26.2%, 24.0%, 21.7%, 29.5%, respectively. However, PSO performs better than SGWO in high-load scenarios, specifically when the number of tasks per workflow ranges from 500 to 1000.

The results obtained demonstrate that the SGWO scheduling algorithm effectively reduces the Makespan of the workflow, compared to SQGA, GA, MHEFT, FCFS scheduling algorithms. It means that SGWO has selected, properly, the distributed nodes in the fog and cloud infrastructure, so that it simultaneously ensures the reduction of workflow task execution time and the data transmission time.

6.2 Evaluation of the total cost

The workflow tasks execution time in the fog and cloud infrastructure, generates a monetary cost by the use of the resources of the infrastructure, see Equation 13. To evaluate the performance of SGWO in terms of Total Cost, and compare it to PSO, SQGA, GA, MHEFT and FCFS, we conducted a series of simulations. The results are illustrated in Figure 8.

According to Figure 8, SGWO achieves the minimum total cost. It outperformed PSO, SQGA, GA, MHEFT and FCFS by 43.5%, 50.9%, 55.5%, 54.6%, 57.9%, respectively.

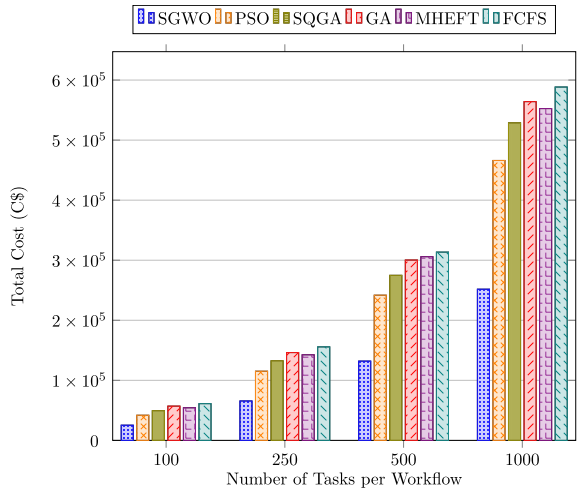


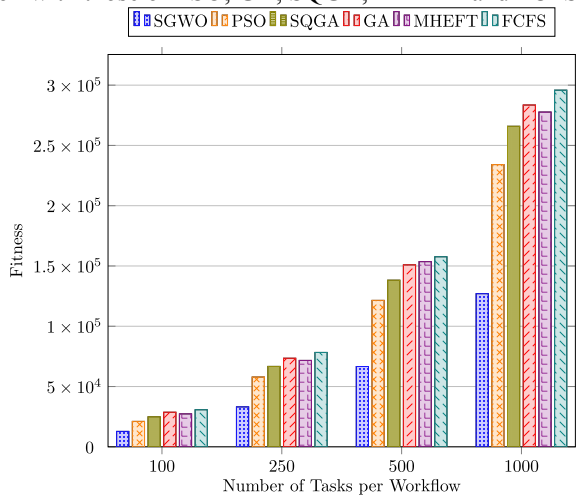
Figure 8: Comparison of the SGWO algorithm Total Cost with those of PSO, GA, SQGA, MHEFT and FCFS

According to the results obtained, SGWO reduced the Total Cost of using the resources of the infrastructure because it has been able to select the adequate nodes to execute the workflow tasks.

6.3 Evaluation of the fitness function

In this paper, we want to optimize the proposed fitness function formulated by Equation 14, therefore, we must minimize at the same time the workflow Makespan and the Total Cost generated by the exploitation of the resources of the fog and cloud layers. For this purpose, we run a series of simulations of SGWO and the other algorithms mentioned above, to compare their fitness functions. The results are shown in Figure 6.3.

Figure 9: Comparison of the SGWO algorithm fitness function with those of PSO, GA, SQGA, MHEFT and FCFS



According to the results in Figure 6.3 we observe that in different simulation cases, the SGWO surpasses PSO, SQGA, GA, MHEFT and FCFS by 43.3%, 50.8%, 55.4%, 54.5% and 57.7%, respectively.

The results demonstrate that SGWO has successfully optimized the fitness function, by minimizing the Total Cost and Makespan simultaneously and considering the coefficients α and β of the objective function. SGWO selects the fog or cloud infrastructure node, to solve the trade-off between Total Cost and Makespan. This selection allows the allocation of workflow tasks to the different infrastructure nodes in a way to utilizes fewer paid resources of fog and cloud infrastructure, which reduces the total execution time.

The proposed SGWO scheduling algorithm is appropriate for IoT applications that require a reduced response time with a minimum resource exploitation cost.

6.4 Analysis of the statistical significance of SGWO's performance

To validate the effectiveness of the proposed scheduling algorithm SGWO compared to the other algorithms PSO, SQGA, MHEFT, FCFS, we performed a series of paired t-tests on the fitness function results obtained by SGWO and the other algorithms compared, through 100 independent runs per algorithm, executing a workflow of 1000 tasks and an infrastructure of 20 cloud nodes and 10 fog nodes. The results obtained are summarized in Table 6.4.

Table 4: Results of paired t-tests between SGWO and other algorithms

Comparison vs SGWO	t-value	p-value	Significant difference
FCFS	-475.24	4.61×10^{-168}	Yes
MHEFT	-424.17	3.54×10^{-163}	Yes
GA	-282.11	1.17×10^{-145}	Yes
SQGA	-358.33	6.26×10^{-156}	Yes
PSO	-167.82	2.25×10^{-123}	Yes

According to the statistical results shown in Table 6.4, SGWO significantly outperforms all other algorithms, with p-values below 0.05, highlighting a high statistical significance that rejects the null hypothesis of equal performance. These test results demonstrate that the performance improvement introduced by the SGWO algorithm is not only observed empirically but is also statistically significant.

6.5 Evaluation of the cloud layer nodes usage

Cloud layer nodes are considered as powerful resources compared to fog layer nodes [5], which helps applications to perform their tasks in a reduced time, this high performance provided, implies a high cost due to the use of paid

cloud nodes [35]. To these effects, we have carried out a series of simulations on SGWO and the other scheduling algorithms PSO, SQGA, GA, MHEFT and FCFS, to compare the rate of cloud nodes usage in their scheduling processes, the results obtained are presented in Figure 10.

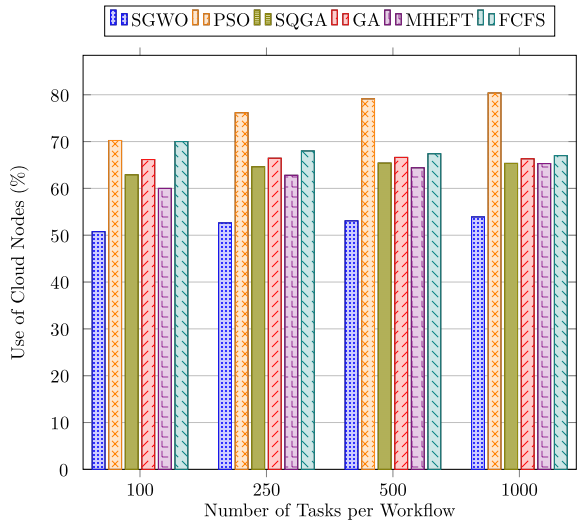


Figure 10: Comparison of the SGWO algorithm cloud node usage with PSO, GA, SQGA, MHEFT and FCFS algorithms

Based on the results shown in Figure 10, we observe that in different simulation cases, SGWO uses fewer cloud nodes in its scheduling process. It only used an average rate of 52.6% of cloud nodes from all existing nodes in the infrastructure, see Equation 2, and outperformed SQGA, GA, MHEFT, FCFS and PSO by 64.6%, 66.4%, 63.1%, 68.1%, 76.5%, respectively. Finally, we also observe that PSO uses a high average rate of cloud layer nodes in its scheduling process.

The obtained results demonstrate that the SGWO scheduling algorithm is more efficient than PSO, SQGA, GA, MHEFT and FCFS algorithms, because it uses fewer cloud layer resources that are paid, which affects the Total Cost, and implies that SGWO gave a minimum value of Total Cost during scheduling (see Figure 8). For this purpose, the reduced use of cloud nodes by SGWO does not affect the Makespan during executing workflow tasks, which gave a minimum value of Makespan compared to the SQGA, GA, MHEFT, FCFS algorithms (see Figure 7), this justifies that SGWO gave a better result in terms of fitness function (see Figure 6.3).

6.6 Evaluation of the convergence of scheduling algorithms

The convergence of an algorithm presents an essential factor for evaluating its performance; for this reason, we need to conduct a simulation case study to examine the effectiveness of the proposed SGWO with PSO, SQGA and GA algorithms. In the simulation scenario, we have fixed the size of the workflow to 1000 tasks for each evaluation, and we increased the number of iterations executed by the scheduling algorithms, to calculate the solutions given by the fitness function 14, The results obtained are presented in the form of graphs for each algorithm in Figure 11, where the average of the measurements is displayed, accompanied by error bars representing the standard error of the mean.

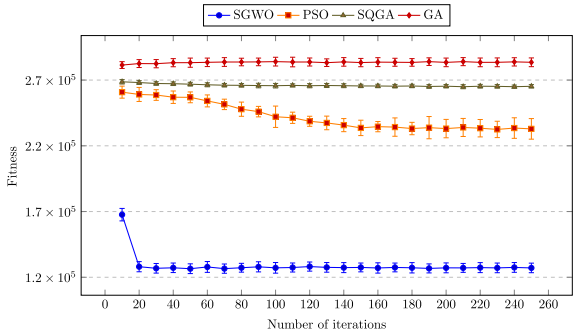


Figure 11: Convergence comparison of SGWO algorithm with PSO, SQGA and GA algorithms

According to the results shown in Figure 11, we observe that SGWO converged from iteration 30 with a better fitness value, then PSO converged starting from iteration 150, and finally, the SQGA and GA gave less efficient fitness values. Notably, SQGA started to converge from iteration 150, while GA showed convergence from iteration 70.

The error bars show the variability of the results. For SGWO, they are normal and relatively stable across all iterations. PSO has the widest error bars, with significant variation depending on the iteration. SQGA has the smallest error bars, indicating good stability. GA has normal error bars, similar to those of SGWO.

Convergence simulations were not performed for the FCFS and MHEFT algorithms, as these generate identical results at each iteration due to their deterministic nature.

The obtained results demonstrate that SGWO is more efficient than PSO, SQGA and GA. It achieves a better fitness value compared to the other algorithms in a reduced number of iterations , while maintaining stable results. This performance is due to the nature of the SGWO algorithm, which is designed to properly balance between exploration and exploitation in its process [31], to discover a higher-quality solution with a reduced time.

6.7 Study the impact of topology on scheduling algorithms

As part of this study, we analyzed the impact of infrastructure topology variation on the performance of different scheduling algorithms, SGWO, PSO, MHEFT, SQGA, and FCFS. The study involved a series of simulations in three distinct topological configurations: an infrastructure consisting only of 10 fog nodes, an infrastructure consisting exclusively of 20 cloud nodes, and finally a hybrid infras-

tructure integrating all nodes, consisting of 10 fog nodes and 20 cloud nodes.

It is important to note that fog nodes generally offer lower performance than cloud nodes, in addition, a hybrid topology involves resource heterogeneity and increased communication between different types of nodes. It should also be noted that the cost of cloud resources is generally higher than that of fog resources.

For each configuration, we measured the average value of the fitness function obtained by the algorithms. The results of these measurements are presented in the graph in Figure 12.

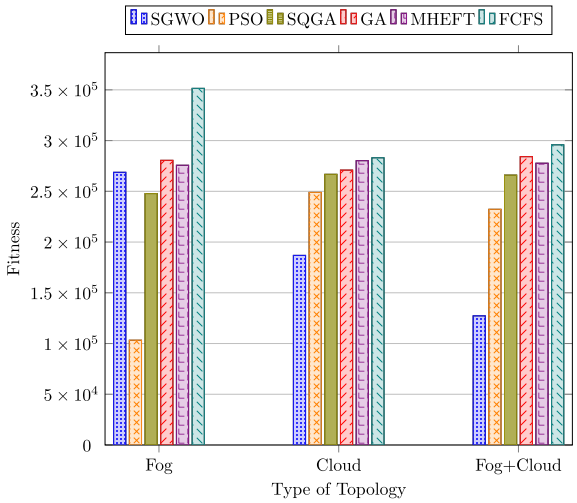


Figure 12: Impact of topology type on the performance of algorithms SGWO, PSO, SQGA, GA, MHEFT, FCFS

The results obtained show that SGWO algorithm offers the best performance in a heterogeneous infrastructure combining fog and cloud nodes, this improvement is reflected in the fitness function, where SGWO outperforms PSO, SQGA, GA, MHEFT and FCFS by 45.2%, 52.2%, 55.2%, 54.2% and 57.0% respectively. Even under cloud topology, the performance of SGWO remains significant in terms of fitness, although it is lower compared to the heterogeneous fog-cloud environment. It still achieves respective improvements of 25.0%, 30.0%, 31.0%, 33.3%, and 34.0% over PSO, SQGA, GA, MHEFT and FCFS. Finally, in the fog-only topology, SGWO reaches its lowest performance level compared to the other topologies. Nevertheless, it still outperforms GA, MHEFT, and FCFS by 4,2%, 2,5%, and 23,5%, respectively, in most scenarios. An exception is observed with the PSO and SQGA algorithms, which display higher fitness values than the proposed SGWO in this case, surpassing it by 61,6% and 7,8%, respectively.

MHEFT performs consistently across different topologies, demonstrating increased robustness to topology changes. Furthermore, PSO stands out for its ability to provide the best fitness value in a topological configuration comprising only fog nodes.

Based on the results obtained, it appears that SGWO algorithm offers better performance in different topologies studied. It seems particularly well suited to the heterogeneous nature of infrastructures, especially in hybrid environments combining fog and cloud. This diversity of resources and communication characteristics helps improve the efficiency of the SGWO algorithm by exploiting the heterogeneity present in the topology.

6.8 Analysis of energy consumption as a function of topology

The following section presents an evaluation of energy consumption resulting from various scheduling algorithms studied under different topologies. A simulation scenario was setup in which the workflow size was set at 1000 tasks per workflow. The infrastructure topology was then varied according to three configurations: fog nodes only, coud nodes only, and a hybrid configuration that combined both types of nodes.

For each configuration, we applied scheduling algorithms: SGWO, PSO, SQGA, GA, MHEFT, FCFS, and measured the average value of the fitness function obtained, in relation to average energy consumption. The results of these simulations are presented in Figure 13. The energy consumption values (in Joules) are represented on a logarithmic scale to better visualize the differences between algorithms across several orders of magnitude.

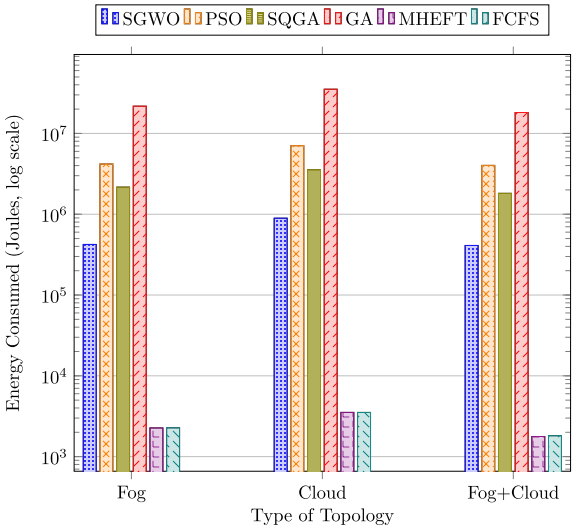


Figure 13: Comparison of energy consumption of algorithms SGWO, PSO, SQGA, GA, MHEFT, FCFS, according to topology (logarithmic Y-axis)

The experimental results indicate that the deterministic algorithms MHEFT and FCFS have lower energy consumption, regardless of the topology used, and metaheuristic algorithms PSO, SQGA and GA are outperformed by SGWO algorithm.

The SGWO algorithm outperforms the PSO, SQGA, and GA algorithms in all of the tested topologies. More specifi-

cally, in the fog topology, it achieves gains of 89.9%, 80.5% and 98.1%, respectively. In the cloud topology, the algorithms PSO, SQGA, and GA are outperformed by 87.3%, 74.9%, 97.5%. Finally, in the hybrid topology, the differences reach 89.8%, 77.4% and 97.7% respectively. This reduction is related to its better fitness value compared to other metaheuristic algorithms, which has a positive impact on several aspects of performance.

Furthermore, the reduced energy consumption of SGWO compared to the studied metaheuristic algorithms is explained by the analysis of the number of cloud nodes used during scheduling in a hybrid topology (see Figure 10) , which demonstrates that SGWO utilizes fewer cloud nodes, thus avoiding the higher energy costs associated with these resources.

7 Discussion and comparative analysis

This section presents a comparative discussion between the proposed algorithm SGWO and the reference algorithms studied in this work, notably PSO, SQGA, GA, MHEFT and FCFS. These algorithms fall into two families: heuristics (SGWO, PSO, SQGA, GA) and metaheuristics (MHEFT, FCFS). The analysis is based on key performance metrics as well as results obtained through different simulation scenarios, with the aim of evaluating the performance of the proposed algorithm SGWO relative to existing approaches.

When evaluating the fitness function, makespan, and total cost, it is observed that SGWO outperforms all other algorithms studied, with the exception of one overload scenario in which PSO achieves a better makespan. However, this improvement does not impact the overall fitness function, which remains better optimized by the SGWO algorithm. The results of the scalability tests (see Figure 6.3) confirm that SGWO remains generally more efficient.

This performance is due, on the one hand, to the metaheuristic nature of SGWO, which allows it to avoid local optima and converge towards a global solution, unlike heuristic approaches, which show inferior performance. On the other hand, SGWO outperforms other algorithms in the same family by achieving the best balance between exploration and exploitation, as shown in Figure 6.3. Furthermore, SGWO demonstrates more efficient resource management than PSO. As shown in Figure 10, PSO tends to use more cloud nodes for scheduling , which are powerful but also more costly than fog nodes. In contrast, SGWO achieves a better balance between cost and makespan using fewer cloud nodes, resulting in a more balanced performance, and lower fitness values than the other algorithms.

According to the convergence graph (see Figure 11), the SGWO algorithm converges faster than the others, with a low standard error, indicating good solution stability. This rapid convergence reduces execution time, making SGWO well suited for IoT applications requiring low latency, such as video surveillance in smart cities, where tasks (pre-processing, motion detection, face detection, archiving) must be efficiently scheduled on fog and cloud nodes. SGWO thus enables fast execution at low cost. The same advantage applies to smart healthcare applications, where responsiveness is also essential.

Evaluation across different topologies (see Figure 12) shows that SGWO adapts well to available resources, offering better fitness values due to its ability to exploit the diversity of Fog-Cloud nodes. Although PSO performs slightly better in a Fog-only topology, SGWO consistently proves more efficient when a Cloud layer is required, especially to run heavy tasks.

An analysis of energy consumption (see Figure 13) shows, on the one hand, that heuristic algorithms consume less energy than metaheuristics. On the other hand, SGWO shows greater energy efficiency than other algorithms in its category. This performance is due to its adaptive selection of nodes according to their computational capacity, which directly influences energy consumption. In fact, the most powerful nodes often consume the most energy.

Nonetheless, SGWO presents a limitation regarding energy efficiency, as energy consumption is not explicitly integrated into its optimization process. This limitation should be addressed in future work, especially for battery-powered IoT applications.

8 Conclusion

In this work, a new Grey Wolf Optimization-based scheduling algorithm (SGWO) has been proposed to efficiently schedule workflow tasks in fog and cloud environments. This scheduling approach enables the simultaneous minimization of Makespan and overall Cost of infrastructure resource utilization.

Experiment results for various simulation cases indicate that the proposed SGWO considerably outperforms PSO, SQGA, GA, MHEFT and FCFS algorithms by 43.3%, 50.8%, 55.4%, 54.5% and 57.7% in terms of fitness function, respectively. This improvement is mainly due to the nature-inspired algorithm GWO, which allows SGWO to quickly converge to the optimal schedule while efficiently allocating heterogeneous resources across both layers.

This means that SGWO efficiently allocates workflow tasks to different nodes in the fog and cloud infrastructure, ensuring a better balance in the trade-off between Makespan and Total Cost. Consequently, SGWO reduces the usage of paid-cloud layer resources without compromising the Makespan, resulting in improved Makespan with a reduced Total Cost.

The experimental results also show that the SGWO algorithm converges quickly (around iteration 30) in the experimental context. This performance can be attributed to the effective balance between the exploration and exploitation phases achieved by proposed algorithm, which allows it to obtain a better fitness value in a heterogeneous and hybrid

topology. SGWO outperforms PSO, SQGA, GA, MHEFT and FCFS by 45.2%, 52.2%, 55.2%, 54.2% and 57.0%, respectively. In addition, SGWO reduces energy consumption compared with PSO, SQGA, and GA by 89.8%, 77.4% and 97.7%, respectively. This performance is mainly due to the lower use of cloud nodes (52.6%), which are known for their high energy consumption. Although SGWO shows lower energy consumption than competing algorithms , but it is still far from optimal, highlighting the need to explicitly integrate energy consumption into the optimization process, particularly for energy-constrained IoT environments.

These performances obtained by SGWO enhance QoS-aware IoT applications by reducing task execution time while minimizing resource utilization costs.

For future work, the objective is to optimize additional metrics such as energy, resource usage and delay, to further improve the performance of IoT applications, meet the QoS requirements of end users, and align the proposed scheduling approach with other comparable infrastructure types. In addition, we plan to compare the proposed solution with machine learning-based scheduling methods, as well as to extend the approach to multi-objective optimization, incorporating key metrics such as energy consumption and latency, which are critical for IoT applications.

References

[1] Maher Abdelshkour. *IoT, from Cloud to Fog Computing*. https://blogs.cisco.com/perspectives/iot-from-cloud-to-fog-computing Accessed Nov. 05, 2024. 2015.

[2] Deafallah Alsadie. "A Comprehensive Review of AI Techniques for Resource Management in Fog Computing: Trends, Challenges, and Future Directions". In: *IEEE Access* 12 (2024), pp. 118007–118059. DOI: https://doi.org/10.1109/ACCESS.2024.3447097.

[3] Muhammad Rizwan Anawar et al. "Fog Computing: An Overview of Big IoT Data Analytics". In: *Wireless Communications and Mobile Computing* 2018 (2018), pp. 1–22. ISSN: 1530-8669. DOI: https://doi.org/10.1155/2018/7157192.

[4] Lokesh Kumar Arya and Amandeep Verma. "Workflow scheduling algorithms in cloud environment - A survey". In: *2014 Recent Advances in Engineering and Computational Sciences (RAECS)* (Mar. 2014), pp. 1–4. DOI: https://doi.org/10.1109/RAECS.2014.6799514.

[5] Hany Atlam, Robert Walters, and Gary Wills. "Fog Computing and the Internet of Things: A Review". In: *Big Data and Cognitive Computing* 2 (Apr. 2018). DOI: https://doi.org/10.3390/bdcc2020010.

[6] Hany F. Atlam et al. "Integration of Cloud Computing with Internet of Things: Challenges and Open Issues". In: *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. 2017, pp. 670–675. DOI: https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2017.105.

[7] Raouf Belmahdi, Djamila Mechta, and Saad Harous. "A Survey on Various Methods and Algorithms of Scheduling in Fog Computing." In: *Ingénierie des systèmes d'information* 26.2 (2021). DOI: https://doi.org/10.18280/isi.260208.

[8] Raouf Belmahdi et al. "SQGA: Quantum Genetic Algorithm-based Workflow Scheduling in Fog-Cloud Computing". In: *2022 International Wireless Communications and Mobile Computing (IWCMC)*. 2022, pp. 131–136. DOI: https://doi.org/10.1109/IWCMC55113.2022.9825324.

[9] Alessio Botta et al. "Integration of Cloud computing and Internet of Things: A survey". In: *Future Generation Computer Systems* 56 (2016), pp. 684–700. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2015.09.021.

[10] Gustavo Caiza et al. "Fog computing at industrial level, architecture, latency, energy, and security: A review". In: *Heliyon* 6 (4 Apr. 2020), e03706. ISSN: 24058440. DOI: https://doi.org/10.1016/j.heliyon.2020.e03706.

[11] Chii Chang, Satish Narayana Srirama, and Rajkumar Buyya. "Indie Fog: An Efficient Fog-Computing Infrastructure for the Internet of Things". In: *Computer* 50.9 (2017), pp. 92–98. DOI: https://doi.org/10.1109/MC.2017.3571049.

[12] OpenFog Consortium. *OpenFog Reference Architecture for Fog Computing*. https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf Accessed Nov. 05, 2024. 2017.

[13] Statista Research Department. *IoT devices installed base worldwide 2015-2025*. https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide Accessed Nov. 05, 2024. 2016.

[14] José Fernando Gonçalves, Jorge José de Magalhães Mendes, and Maurício G.C. Resende. "A hybrid genetic algorithm for the job shop scheduling problem". In: *European Journal of Operational Research* 167 (1 Nov. 2005), pp. 77–95. ISSN: 03772217. DOI: https://doi.org/10.1016/j.ejor.2004.03.012.

[15] Ali Salah Hashim, Wid Akeel Awadh, and Mohammed S Hashim. “Non-dominated sorting genetic optimization-based fog cloudlet computing for wireless metropolitan area networks”. In: *Informatica* 47.10 (2023). DOI: https://doi.org/10.31449/inf.v47i10.5118.

[16] Doan Hoang and Thanh Dat Dang. “FBRC: Optimization of task Scheduling in Fog-Based Region and Cloud”. In: *2017 IEEE Trustcom/BigDataSE/ICESS*. 2017, pp. 1109–1114. DOI: https://doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.360.

[17] Farooq Hoseiny et al. “PGA: A Priority-aware Genetic Algorithm for Task Scheduling in Heterogeneous Fog-Cloud Computing”. In: *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2021, pp. 1–6. DOI: https://doi.org/10.1109/INFOCOMWKSHPS51825.2021.9484436.

[18] Pejman Hosseinioun et al. “aTask scheduling approaches in fog computing: A survey”. In: *Transactions on Emerging Telecommunications Technologies* 33 (3 Mar. 2022). ISSN: 2161-3915. DOI: https://doi.org/10.1002/ett.3792.

[19] Mehdi Hosseinzadeh et al. “Task Scheduling Mechanisms for Fog Computing: A Systematic Survey”. In: *IEEE Access* 11 (2023), pp. 50994–51017. DOI: https://doi.org/10.1109/ACCESS.2023.3277826.

[20] Lu Hou et al. “Internet of Things Cloud: Architecture and Implementation”. In: *IEEE Communications Magazine* 54.12 (2016), pp. 32–39. DOI: https://doi.org/10.1109/MCOM.2016.1600398CM.

[21] Pengfei Hu et al. “Survey on fog computing: architecture, key technologies, applications and open issues”. In: *Journal of Network and Computer Applications* 98 (Nov. 2017), pp. 27–42. ISSN: 10848045. DOI: https://doi.org/10.1016/j.jnca.2017.09.002.

[22] Omer K. Jasim Mohammad and Bassim M. Salih. “Improving Task Scheduling in Cloud Datacenters by Implementation of an Intelligent Scheduling Algorithm”. In: *Informatica (Slovenia)* 48.10 (2024), pp. 77–88. DOI: https://doi.org/10.31449/inf.v48i10.5843.

[23] Amir Karamoozian, Abdelhakim Hafid, and El Mostapha Aboulhamid. “On the Fog-Cloud Cooperation: How Fog Computing can address latency concerns of IoT applications”. In: *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*. 2019, pp. 166–172. DOI: https://doi.org/10.1109/FMEC.2019.8795320.

[24] J. Kennedy and R. Eberhart. “Particle swarm optimization”. In: *Proceedings of ICNN’95 - International Conference on Neural Networks*. Vol. 4. 1995, 1942–1948 vol.4. DOI: https://doi.org/10.1109/ICNN.1995.488968.

[25] Zulfiqar Ali Khan et al. “A Review on Task Scheduling Techniques in Cloud and Fog Computing: Taxonomy, Tools, Open Issues, Challenges, and Future Directions”. In: *IEEE Access* 11 (2023), pp. 143417–143445. DOI: https://doi.org/10.1109/ACCESS.2023.3343877.

[26] P. Kuppusamy et al. “Job scheduling problem in fog-cloud-based environment using reinforced social spider optimization”. In: *Journal of Cloud Computing* 11.1 (2022). ISSN: 2192-113X. DOI: https://doi.org/10.1186/s13677-022-00380-9.

[27] Jian Ma et al. “Reliable Task Scheduling in Cloud Computing Using Optimization Techniques for Fault Tolerance”. In: *Informatica* 48.23 (2024), pp. 159–170. DOI: https://doi.org/10.31449/inf.v48i23.6901.

[28] Khaled Matrouk and Kholoud Alatoun. “Scheduling Algorithms in Fog Computing: A Survey”. In: *International Journal of Networked and Distributed Computing* 9 (1 Jan. 2021), p. 59. ISSN: 2211-7946. DOI: https://doi.org/10.2991/ijndc.k.210111.001.

[29] Yamina Mehor, Mohammed Rebbah, and Omar Smail. “Energy-aware Scheduling of Tasks in Cloud Computing.” In: *Informatica* 48.16 (2024). DOI: https://doi.org/10.31449/inf.v48i16.5741.

[30] Eva Mindasari, Sawaluddin Sawaluddin, and Parapat Gultom. “Comparison of Genetic Algorithm and Particle Swarm Optimization in Determining the Solution of Nonlinear System of Equations”. In: *sinkron* 8 (3 July 2024), pp. 1600–1607. ISSN: 2541-2019. DOI: https://doi.org/10.33395/sinkron.v8i3.13785.

[31] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. “Grey Wolf Optimizer”. In: *Advances in Engineering Software* 69 (2014), pp. 46–61. ISSN: 0965-9978. DOI: https://doi.org/10.1016/j.advengsoft.2013.12.007.

[32] Nitin Mittal, Urvinder Singh, and Balwinder Singh Sohi. “Modified grey wolf optimizer for global engineering optimization”. In: *Applied Computational Intelligence and Soft Computing* 2016 (2016). DOI: https://doi.org/10.1155/2016/7950348.

[33] C. Muro et al. “Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations”. In: *Behavioural Processes* 88.3 (2011), pp. 192–197. ISSN: 0376-6357. DOI: https://doi.org/10.1016/j.beproc.2011.09.006.

[34] Tina Samizadeh Nikoui et al. “Cost-Aware Task Scheduling in Fog-Cloud Environment”. In: *2020 CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*. 2020, pp. 1–8. DOI: https://doi.org/10.1109/RTEST49666.2020.9140118.

[35] Xuan-Qui Pham et al. “A cost- and performance-effective approach for task scheduling based on collaboration between cloud and fog computing”. In: *International Journal of Distributed Sensor Networks* 13.11 (2017). DOI: https://doi.org/10.1177/1550147717742073.

[36] Narayana Potu, Chandrashekar Jatoth, and Premchand Parvataneni. “Optimizing resource scheduling based on extended particle swarm optimization in fog computing environments”. In: *Concurrency and Computation: Practice and Experience* 33 (23 Dec. 2021). ISSN: 1532-0626. DOI: https://doi.org/10.1002/cpe.6163.

[37] Esmat Rashedi, Hossein Nezamabadi-Pour, and Saeid Saryazdi. “GSA: a gravitational search algorithm”. In: *Information sciences* 179.13 (2009), pp. 2232–2248. DOI: https://doi.org/10.1016/j.ins.2009.03.004.

[38] Yichen Ruan et al. “Pricing Tradeoffs for Data Analytics in Fog–Cloud Scenarios”. In: *Fog and Fogonomics* (Mar. 2020), pp. 83–106. DOI: https://doi.org/10.1002/9781119501121.ch4.

[39] Muhammad Saad, Rabia Noor Enam, and Rehan Qureshi. “Optimizing multi-objective task scheduling in fog computing with GA-PSO algorithm for big data application”. In: *Frontiers in Big Data* 7 (Feb. 2024). ISSN: 2624-909X. DOI: https://doi.org/10.3389/fdata.2024.1358486.

[40] Ling Shen et al. “A hybrid GA-PSO algorithm for seru scheduling problem with dynamic resource allocation”. In: *International Journal of Manufacturing Research* 18.1 (2023), pp. 100–124. DOI: https://doi.org/10.1504/IJMR.2023.129301.

[41] Lovejit Singh and Sarbjeet Singh. “A Genetic Algorithm for Scheduling Workflow Applications in Unreliable Cloud Environment”. In: *Recent Trends in Computer Networks and Distributed Systems Security: Second International Conference, SNDS 2014, Trivandrum, India, March 13-14, 2014, Proceedings 2*. Vol. 420. 2014, pp. 139–150. DOI: https://doi.org/10.1007/978-3-642-54525-2_12.

[42] Rainer Storn and Kenneth Price. “Differential Evolution-A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces”. In: *Journal of Global Optimization* 11 (1997), pp. 341–359. DOI: https://doi.org/10.1023/A:1008202821328.

[43] Jin Xie et al. “Review on flexible job shop scheduling”. In: *IET Collaborative Intelligent Manufacturing* 1 (3 Sept. 2019), pp. 67–77. ISSN: 2516-8398. DOI: https://doi.org/10.1049/iet-cim.2018.0009.

[44] Vinita Yadav, B V Natesha, and Ram Mohana Reddy Guddeti. “GA-PSO: Service Allocation in Fog Computing Environment Using Hybrid Bio-Inspired Algorithm”. In: *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*. 2019, pp. 1280–1285. DOI: https://doi.org/10.1109/TENCON.2019.8929234.

[45] Hyunsik Yang and Younghan Kim. “Design and Implementation of High-Availability Architecture for IoT-Cloud Services”. In: *Sensors* 19.15 (2019). ISSN: 1424-8220. DOI: https://doi.org/10.3390/s19153276.

[46] Mingyang Yu et al. “Improved multi-strategy adaptive Grey Wolf Optimization for practical engineering applications and high-dimensional problem solving”. In: *Artificial Intelligence Review* 57 (10 Sept. 2024). The time complexity of theoriginal GWO algorithm is O(T × N × D) .page 15, p. 277. ISSN: 1573-7462. DOI: 10.1007/s10462-024-10821-3. URL: https://link.springer.com/10.1007/s10462-024-10821-3.

[47] Arkady Zaslavsky, Charith Perera, and Dimitrios Georgakopoulos. “Sensing as a Service and Big Data”. In: *ArXiv* (Jan. 2013). DOI: https://doi.org/10.48550/arXiv.1301.0159.

[48] Honglin Zhang, Yaohua Wu, and Zaixing Sun. “EHEFT-R: multi-objective task scheduling scheme in cloud computing”. In: *Complex & Intelligent Systems* 8 (6 Dec. 2022). ajouter EHEFT-R, pp. 4475–4482. ISSN: 2199-4536. DOI: 10.1007/s40747-021-00479-7. URL: https://link.springer.com/10.1007/s40747-021-00479-7.

[49] Junlong Zhou et al. “Cost and makespan-aware workflow scheduling in hybrid clouds”. In: *Journal of Systems Architecture* 100 (2019), p. 101631. ISSN: 1383-7621. DOI: https://doi.org/10.1016/j.sysarc.2019.08.004.