Indonesian Journal of Electrical Engineering and Computer Science
Vol. 40, No. 2, November 2025, pp. 871∼882
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v40.i2.pp871-882

Fuzzy multi-objective energy optimization of workflow scheduling

Ayoub Chehlafi, Mohammed Gabli
Department of Computer Science, Faculty of science (FSO), University Mohammed First (UMF), Oujda, Morocco

Article Info

Article history:

Received Feb 2, 2025
Revised Jul 18, 2025
Accepted Oct 14, 2025

Keywords:

Dynamic multi-objective
Energy efficiency
Fuzzy logic
Metaheuristic algorithms
Optimization
Workflow scheduling

ABSTRACT

Task scheduling is a key and challenging problem in cloud computing systems, requiring decisions regarding resource allocation to tasks to optimize a performance criterion. This problem has required researchers and developers to overcome significant challenges. Our goal in this study aims to minimize both the makespan and energy consumption in cloud computing systems by efficiently scheduling workflows. To achieve this, we first proposed a dynamic multi-objective model, which was then simplified into a single-objective problem using dynamic weights. Then, we proposed a dynamic genetic algorithm (DGA) and a dynamic particle swarm optimization algorithm (DPSO) to address the problem. To deal with the situation where the makespan is uncertain and not exact, we present a fuzzy model, treating each value as a fuzzy number and we utilize both possibility and necessity metrics. The results are contrasted with the Heterogeneous earliest finish time (HEFT) algorithm and Considerably lowered the total energy consumption, especially for DGA.

CC BY SA

Corresponding Author:

Chehlafi Ayoub
Department of Computer Science, Faculty of science (FSO), University Mohammed First (UMF)
Oujda, Morocco
Email: ayoub.chehlafi@ump.ac.ma

1. INTRODUCTION

Task scheduling in cloud computing settings has become a subject of significant academic interest owing to the exponential growth in data volumes and the increasing demand for reduced execution times. This explosive growth of data traffic has forced researchers and developers to propose numerous approaches aimed at handling the complexities of task scheduling. These methods aim to optimize objectives including makespan, energy consumption, or cost. Nonetheless, the essential complexity and fluctuating dynamic nature of cloud environments present substantial obstacles to traditional scheduling algorithms. In cloud computing, one of the primary challenges is scheduling workflows efficiently while preserving the task dependencies within the workflow structure. Task scheduling in cloud refers to the process of assigning tasks to available resources and managing their execution to achieve optimal performance. This process guarantees the allocation of tasks to the most suitable virtual machines (VMs) or servers, depending on factors like resource availability, computational capabilities, and task requirements. The second challenge is minimizing metrics such as makespan and energy consumption. Figure 1 concisely illustrates the generation and processing steps involved in scientific workflow applications. Indeed, multiple applications, originating from users, submit complex requests requiring multi-tasking. These requests are modeled as workflows. Once generated, the workflow is sent to the cloud, where it is processed in a distributed manner. Processing is handled by different service centers (SCs), each hosting multiple VMs.

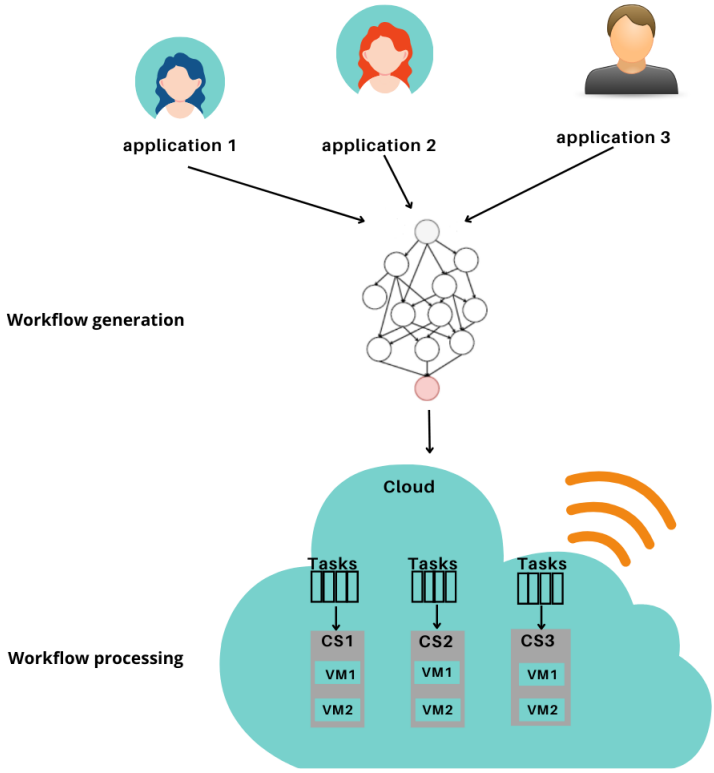


Figure 1. The scheduling of workflows in cloud computing

Cloud computing denotes the use of computing resources distributed across cloud servers. Each cloud server (CS) hosts a set of m VMs, then, $CS = \{V_1, V_2, \ldots, V_m\}$. The VMs operating on a server can be heterogeneous, varying in processing power, memory capacity, and storage space. Additionally, each cloud server includes a resource manager responsible for storing information about the available VMs, and creating, allocating, and deleting VMs as needed.

To represent the workflow model we adopt a directed acyclic graph (DAG) denoted as $G =< T; E >$, where $T = \{T_1, \ldots, T_n\}$ signifies the collection of n tasks in the workflow. The set $E = \{c_{i,j} | 1 \leq i \leq n, 1 \leq j \leq n\}$ describes the communication requirements between task T_i and Tj, and if task T_i precedes task T_j, then, task T_j cannot start execution until task T_i has completed its execution. A communication cost is associated with data transfer between tasks. However if two tasks T_i and T_j are affected to the identical machine, the communication cost becomes zero because the data remains on the same machine.

In the considered architecture, the time execution of a task primarily relies on runtime and communication time (CT). The runtime (RT) containing the execution times of different tasks on various VMs, and depends on the size of a specific task T_i. The RT between task T_i and VM V_j can be expressed by (1). Accordingly, the CT is determined by the data size and the bandwidth of the communication channel, as represented by (1).

$$RT_{(i,j)} = \frac{t_{\text{size}}}{V_j} \tag{1}$$

$$CT_{(i,j)} = \begin{cases} 0 & \text{if } V_i = V_j \\ \frac{D_{i,j}}{B} & \text{if } V_i \neq V_j, \end{cases} \tag{2}$$

where $D_{(i,j)}$ specifies the volume of data exchanged from task T_i to T_j, and B indicates the communication channel bandwidth between two cloud servers.

Let $EST(T_i, V_j)$ be the starting time and $EFT(T_i, V_j)$ be finishing time where T_i is task assigned to V_j. We defined $EST(T_i, V_j)$ as follows [1]:

$$EST(T_i, V_j) = \max\left\{\text{avail}[j], \max_{T_m \in \text{prd}(T_i)} (EFT(T_m, V_j) + CT_{m,i})\right\} \quad (3)$$

where `avail[j]` indicates the moment when VM V_j become available. If another task is still running on this machine, T_i will need to wait until V_j is free before starting. This factor ensures that T_i does not begin until the machine is ready. And, `prd(T_i)` represents the collection of predecessor tasks (Tasks that must be completed before launching T_i). If T_i has multiple dependencies, the EFT of the predecessor task that finishes last is taken.

On the other hand, $EFT(T_i, V_j)$ can be defined by [1]:

$$EFT\,(T_i, V_j) = w_{i,j} + \text{EST}\,(T_i, V_j) \quad (4)$$

where $w_{i,j}$ Corresponds to the workload (execution time) for task T_i on VM V_j.

MAKESPAN. In workflow the makespan is defined as the finish time of the last task (T_{exit}). It is indicated by (5).

$$M = \max\{EFT(T_{\text{exit}}, V)\} \quad (5)$$

ENERGY. The system saves energy via dynamic voltage and frequency scaling. Complementary metal oxide semiconductor (CMOS) use dynamic and static energy. The latter is ignored since dynamic power dissipation is the most expensive and time-consuming [1]. The total energy consumption is defined as follows (6).

$$E_{\text{total}} = K \times (\sum_{j=1}^{m} V_j^2 \times f \times t_j + V_{lowest_j}^2 \times f_{lowest_j} \times t_{idle}) \quad (6)$$

where K is a dynamic power constant. V_j is the voltage provided for the j_{th} VM and f is the frequency at the V_j of same VM. t_j is the time for which task is running on VM v_j. V_{lowest_j} and f_{lowest_j} are, respectively, the lowest voltage and lowest frequency of the VM v_j. Finally, t_{idle} represents the idle time of v_j.

Several studies have addressed this issue, and researchers have developed various techniques to optimize task execution by parallelizing sub-tasks while maintaining their dependencies. A review of the literature on workflow scheduling highlights two main categories of approaches based on the number of objectives they address: mono-objective and multi-objective workflow scheduling.

Mono-objective workflow scheduling focuses on optimizing one metric, such as makespan, energy consumption, or cost. For instance, Ding *et al.* [2] the authors optimize energy consumption in cloud computing using dynamic task scheduling based on Q-learning. Wang and Zuo [3] to reduce the total time needed for completing tasks in workflow scheduling, integrated PSO with idle time slot-aware rules. Another notable contribution, in [4] Faragardi *et al.* proposes an algorithm called Greedy Resource Provisioning and modified HEFT, which is designed to reduce the makespan of a specific workflow subject to a budget constraint. Wu *et al.* [5], minimized makespan by accounting for both computational and communication resources. Additionally, Lu *et al.* [6] utilized a method multi-hierarchy PSO to reduce total monetary cost by applying an on-demand pricing structure for heterogeneous cloud resources.

Moving to multi-objective workflow scheduling, researchers have focused on optimizing two or three metrics simultaneously, like cost and makespan. For instance, Zhu *et al.* [7] a multi-objective evolutionary optimization method was developed aimed at reducing both makespan and cost. Tang [8] introduced a new method known as fault-tolerant, cost-efficient workflow scheduling, which reduces the costs of executing applications and makespan while maintaining reliability. Chen *et al.* [9] introduced a new method called multiobjective ant colony system approach , which focuses on minimizing the workflow execution time and cost by utilizing co-evolutionary multiple populations for multiple objectives. Furthermore, Iranmanesh and Naji [10] presents a hybrid GA to reduce both the cost and makespan, integrating enhanced genetic operators, adaptive fitness functions, and a load balancing routine. Additional research, such as Jena [11], applied nested PSO to decrease two main objectives like the makespan and energy consumption, though it overlooked resource utilization. Similarly, Kumar *et al.* [12] the authors proposed a new approach to minimize the makespan and energy consumption systems using PSO. Verma and Kaushal [13] presented a hybrid multi-objective PSO approach for

the efficient scheduling of scientific workflows, maximizing resource utilization while minimizing execution time and costs. Zhou *et al.* [14] used fuzzy dominance sort with HEFT algorithm to explore the collaborative optimization of makespan and cost. Durillo and Prodan [15], researchers combined HEFT with other meta-heuristic techniques like PSO to improve results. Hao *et al.* [16] proposed a three-objective DAG problem to minimize makespan, energy cost, and maximize revenue for cloud scheduling systems. Similarly, Yuan *et al.* [17] introduced the IMEAD algorithm to balance revenue and energy costs effectively. Another study, detailed in [18], presented a multi-objective GA (MOGA) that considers conflicting stakeholder interests while optimizing makespan, budget, and energy efficiency. To optimize resource utilization, energy consumption, and cost while enhancing security, thus benefiting both users and service providers [19]. Furthermore in [20], Adhikari *et al.* introduced a strategy based on the Firefly Algorithm aimed at tackling several competing goals, such as workload allocation, total completion time, resource usage, and dependability. Behera and Sobhanayak [21] presented a novel hybrid algorithm that merges GA and grey wolf optimization to reduce three key performance indicators like makespan, energy consumption, and computational cost. Wu *et al.* [22] authors propose a multi-objective optimization model for collaborative task scheduling across cloud, edge, and end devices. It focuses on reducing task delay and improving load balancing by optimizing server allocation, service deployment, caching, and resource allocation. Abualigah *et al.* [23], introduced an improved synergistic swarm optimization algorithm, enhanced with the Jaya approach to minimize makespan and improve load balancing. Although in [24], Zade *et al.* introduced an modified beluga whale optimization algorithm that incorporates a ring topology to improve task scheduling in cloud computing. This approach aims to reduce both makespan and cost by boosting solution diversity and preventing early convergence.

Our goal in this paper is to identify the best solution to the task scheduling problem to decrease two main objectives: the makespan and energy consumption in the cloud. Our problem is then multi-objective. In order to contribute to the improvement of existing literature, we addressed two challenges: (i) The first is to find a way to fairly optimize each criterion of the objective function, i.e., not to minimize one criterion (makespan or energy) at the expense of the other. (ii) The second is to consider the real situation where the makespan varies dynamically due to several factors and remains not exact but uncertain, see for instance [25]-[27].

To meet the first challenge, we introduced a dynamic multi-objective modeling of the problem, by using dynamic and automating the weight selection process, see the next section. Then we developed two metaheuristics to solve it. Concerning the second challenge, and to make our model more realistic compared to other studies, we improved it by introducing a fuzzy model which in each makespan value is defined as a fuzzy number. To achieve this, we used both possibility and necessity metrics. Finally, we compared our results to those of other algorithms, such as the HEFT algorithm used in [1]. The results found are promising and show the robustness of our approach. The primary contributions of this paper are summarized below.

- A dynamic multi-objective model was proposed of the problem focusing on reducing both makespan and energy consumption.
- We improved our model by using dynamic weights to ensure that all objectives are treated equitably and to automate the choice of weights.
- We introduced a fuzzy model considering the uncertain aspect of the makespan. This makes our model more realistic.
- We proposed two dynamic meta-heuristic algorithms to identify a solution to the proposed problem and we compared its by HEFT algorithm.

The paper is organized into the following sections. In section 2, we introduced a new model by reformulating the existing one and considering the dynamic and uncertain aspect of our problem. In section 3, we presented our method and described the two algorithms developed to address this challenge. In section 4, we presented and discussed the obtained numerical results and compared them with the HEFT algorithm. Section 5 concludes this paper.

2. THE PROPOSED MODEL

Based on the above analyses, this research aims to identify a compromise solution that simultaneously minimizes both the makespan and energy consumption. So, we introduce a binary decision variable $X_{i,j}$ between the task T_i and the VM V_j as follows.

$$X_{i,j} = \begin{cases} 1 & \text{if } T_i \text{ is assigned to } V_j \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

In result, we have two objectives to achieve, and hence we obtain a multi-objective problem as follows.

$$\begin{cases} \text{Minimize } f_1(X_{ij}) = \min M \\ \text{Minimize } f_2(X_{ij}) = \min E_{\text{total}} \end{cases} \tag{8}$$

It is unreasonable to view CT as precise but uncertain because it varies due to several factors, see for instance [25]-[27]. To address this issue, we present a fuzzy model,which in each communication time CT_{ij} is defined as fuzzy numbers $C\tilde{T}_{ij}$ with the subsequent membership function:

$$\mu_{C\tilde{T}_{ij}}(t) = \begin{cases} max(0, 1 - \frac{CT_{ij}-t}{\alpha_{ij}}) & \text{if } t <= CT_{ij} \\ max(0, 1 - \frac{t-CT_{ij}}{\beta_{ij}}) & \text{if } t > CT_{ij} \end{cases} \tag{9}$$

where the constants α_{ij} and β_{ij} represent the left-side and right-side spreads of fuzzy numbers, with both being positive values. The corresponding membership function $\mu_{C\tilde{T}_{ij}}$ presented in Figure 2.

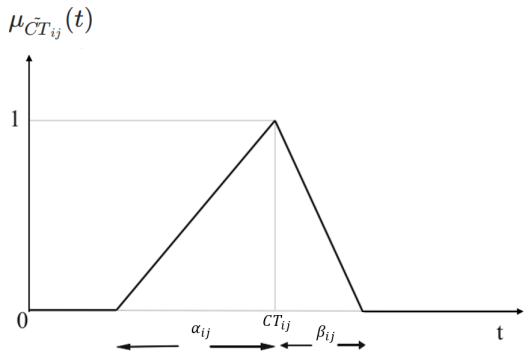


Figure 2. Membership function of $C\tilde{T}_{ij}$

In this study, we use both possibility and necessity metrics. To determine the makespan using the possibility measure, we set $\beta_{ij} = 0$ for each task i and j. To accomplish the necessity measure, we set $\alpha_{ij} = 0$ for each task i and j. For the process of defuzzification, we apply the center of gravity method which involves identifying of taking the abscissa corresponding to the center of gravity of the membership function.

After this new modeling, the multiobjective problem in 2. becomes:

$$\begin{cases} \text{Minimize } \tilde{f}_1(X_{ij}) = \min \tilde{M} \\ \text{Minimize } f_2(X_{ij}) = \min E_{\text{total}} \end{cases} \tag{10}$$

we convert this multi-objective problem into a single-objective formulation as follows.

$$\text{Minimize } f(X_{ij}) = \text{Minimize } (\omega_1 \times \tilde{f}_1(X_{ij}) + \omega_2 \times f_2(X_{ij})) \tag{11}$$

Subject to:

$$\sum_{j=1}^{m} X_{ij} = 1, \quad \forall i, \quad 1 \leq i \leq n \tag{12}$$

which means that a task T_i should not be assigned to only one VM V_j. ω_1 and ω_2 are positive weights satisfying $0 \leq \omega_k \leq 1$, $k = 1, 2$, and $\omega_1 + \omega_2 = 1$. The choice of ω_1 and ω_2 for the decision maker is not an easy. Moreover, these values should be chosen in such a way that the optimization of both criteria ($\tilde{f}_1$ and f_2) is fair. To do this, ω_1 and ω_2 should not be considered constant, but change dynamically in each iteration (t) of the algorithm as in [28].

$$w_1(t+1) = \frac{f_2(X_t)}{\tilde{f}_1(X_t) + f_2(X_t)} \quad \text{and} \quad w_2(t+1) = \frac{\tilde{f}_1(X_t)}{\tilde{f}_1(X_t) + f_2(X_t)} \tag{13}$$

where t is an iteration of the used algorithm. Consequently, our problem becomes.

$$\text{Minimize } f(X_{ij}) = \text{Minimize } (\omega_1(t) \times \tilde{f}_1(X_{ij}) + \omega_2(t) \times f_2(X_{ij})) \tag{14}$$

3. METHOD

Let n and m correspond to the total number of tasks and VMs, respectively. To model our problem in DGA and DPSO, we use an integer encoding scheme. We represented each solution like an vector of n integers, where each element's value ranges from 0 to m. For instance, in Figure 3, the encoding '2021212011' indicates that task T_0 is assigned to VM_2, T_1 is assigned to VM_0, and we have 3 machines ($VM_0, VM_1,$ and VM_2).

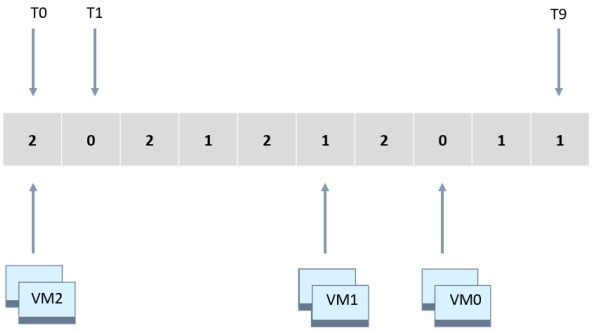


Figure 3. Example of solution representation

3.1. Dynamic particle swarm optimization (DPSO) algorithm

The DPSO algorithm comprises S particles, where each particle's position corresponds to a candidate solution within the search space. The particles update their states based on the equations defined in (15) and (16), respectively.

$$V_i^{t+1} = (\omega \otimes V_i^t) \oplus (q1 \otimes (bestP_i \ominus P_i^t)) \oplus (q2 \otimes (bestG \ominus P_i^t)). \tag{15}$$

$$P_i^{t+1} = P_i^t \oplus V_i^{t+1} \tag{16}$$

where ω represents the inertia weight, q_1 and q_2 are coefficient numbers selected randomly from the interval $[0, 1]$, during each iteration, $bestP_i$ denotes the best solution identified by the particle x_i and $bestG$ corresponds to the best solution identified by all particles.

Define $f(x) = \omega_1 \tilde{f}_1(x) + \omega_2 f_2(x)$ as the fitness function to be optimized and S corresponds to the swarm size. The main procedures of the proposed DPSO algorithm are presented in Algorithm 1.

To apply the DPSO algorithm to our problem, we proceed as follows.

- Step 1: (Initialize particles) Consider n tasks and m VMs. each particle in the population is encoded as a vector of n integers, where each element in the vector as randomly selected from de set $\{0, \cdots, m\}$. A population refers to a collection of solutions.
- Step 2: (Fitness evaluation) The objective function value assigned to each particle is computed using (14). If the particle's current objective function value is more effective than its previous personal best (bestP), the actual value is assigned as the new bestP. Then, the global best solution (bestG) is determined as the particle with the highest fitness across the entire population.
- Step 3: (Update velocity and position) We calculate the velocity of each particle using in (15). Followed by a position update using in (16).
- Step 4: Dynamically update ω_1 and ω_2 according to (13).
- Step 5: In this step, we find bestG.

Algorithm 1 DPSO algorithm

Initialize Swarm parameters and S.
Initialize Positions and velocities of all particles.
for each solution i **do**
  Set $bestP_i \leftarrow x_i$
  Evaluate $f(x_i)$ using (14)
end for
while the termination criterion is not met **do**
  for $1 < i < S$ **do**
    Compute velocity of the particle using Eq.(16),
    Compute position of the particle using Eq. (15),
    Evaluate $f(x_i)$ of each solution i
    if $f(x_i)$ is more effective $bestPi$ **then**
      $bestP_i = x_i$
    end if
    if $f(bestP_i)$ is more effective than $f(bestG)$ **then**
      $bestG = bestP_i$
    end if
  end for
  Dynamically update ω_1 and ω_2 according to 2..
end while
Return the best solution bestG.

3.2. Dynamic genetic algorithm (DGA)

The DGA is a metaheuristic that mimics natural evolution. It works with populations composed of several solutions. The population size represents the total number of chromosomes. Each chromosomes is referred to as a solution (individual). Each chromosome has a set of genes. In this paper, we developed a DGA as illustrated in Algorithm 2.

Algorithm 2 DGA Algorithm

Step 1: Initialize the population with random individuals.
Step 2: Repeat for a fixed number of generations
  a. Evaluate each chromosome's fitness in the population using (14).
  b. Select parents for reproduction based on their fitness.
  c. Create a new generation applying crossover and mutation operators to parents.
  d. dynamically update ω_1 and ω_2 according to (13)
Step 3: Select the best individual from the current population as the final solution.

To apply the DGA algorithm to our problem, we proceed as follows.

- Codage: we used the representation defined in section 3.
- Initial population: Consider n tasks and m VMs. Each chromosome in the population is initialized with n randomly generated genes, in which each gene is an integer drawn from the collection $\{0, ..., m\}$. A population is a collection of solutions.
- Selection: We use in this operation the roulette wheel method.
- Crossover: A single-point crossover is applied where we determine randomly the crossover position. All genes located after this point are swapped between the two parent chromosomes. The crossover process is illustrated in Figure 4.
- Mutation: Once the gene (digit) selected for mutation, its value is exchanged with a number randomly selected from the set $\{1, 2, ..., m\}$. The process of mutation operation is demonstrated in Figure 5.
- Weights: We dynamically update ω_1 and ω_2 according to (13).

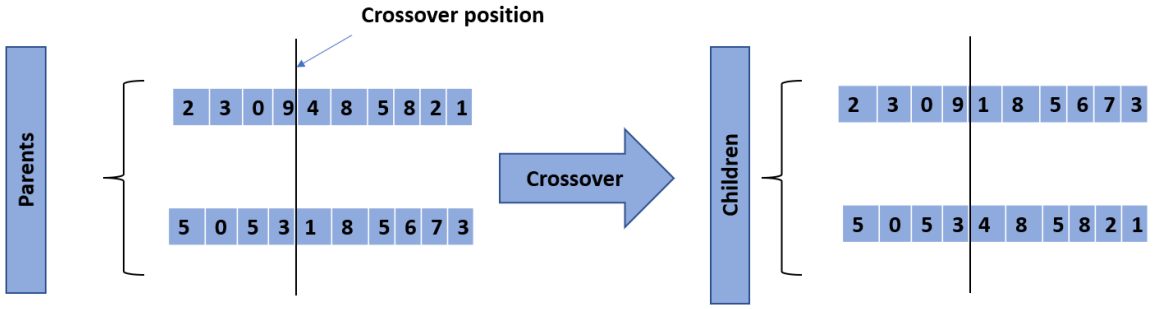


Figure 4. The crossover mechanism

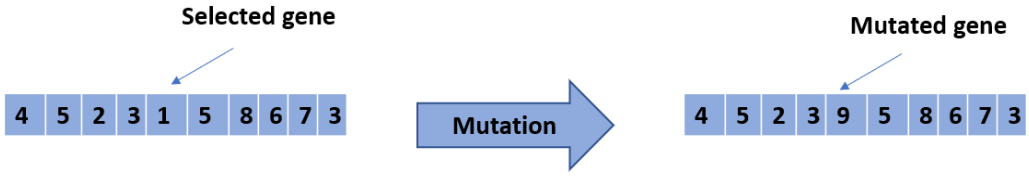
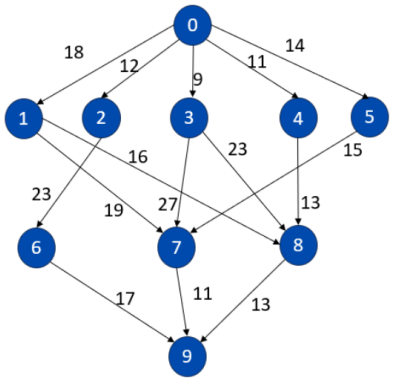


Figure 5. The mutation mechanism

4. RESULTS AND DISCUSSION

4.1. Problem data

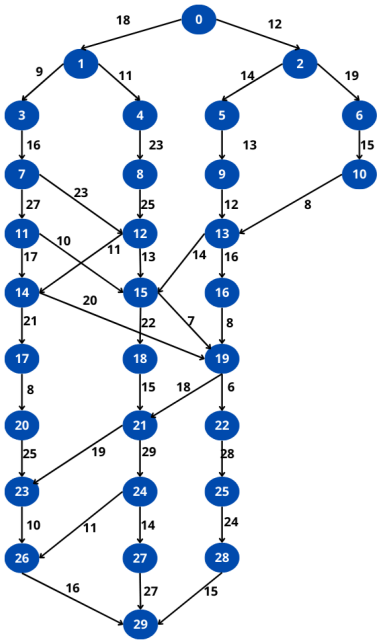
This part measures the performance of our algorithms by testing them utilizing two distinct problem instances. The first one contains 10 tasks and 3 VMs. The second one contains 30 tasks and 3 VMs. The two DAG with communication costs between the nodes distributed across three VMs are presented in Figure 6 and Figure 7, respectively. Thus, Figure 6 on the left presents the communication requirements between tasks T_i and T_j. For example, the value 18 between tasks T_0 and T_1 represents the quantity of data to be sent from task T_0 to task T_1. Figure 6 on the right also presents the computation time matrix between tasks and VM_s. For example, task T_0 costs 14 ms if executed on VM_1, 16 ms if executed on VM_2, and 9 ms if executed on VM_3. The same is true for Figure 7 on the left and right, but this time for 30 tasks and 3 VMs.



Tâche	VM1	VM2	VM3
0	14	16	9
1	13	19	18
2	11	13	19
3	13	18	7
4	12	13	10
5	13	16	9
6	7	15	11
7	5	11	14
8	18	12	20
9	21	7	16

Task execution times on VMs

Figure 6. First problem instance: DAG with 10 tasks, 3 VMs and a matrix detailing the computation times for each task on the different VMs



Taches	VM1	VM2	VM3
0	14	16	9
1	13	19	18
2	11	13	19
3	13	18	7
4	12	13	10
5	13	16	9
6	7	15	11
7	5	11	14
8	18	12	20
9	21	7	14
10	21	21	16
11	15	7	17
12	16	17	21
13	22	7	15
14	23	15	13
15	20	7	19
16	7	19	20
17	15	7	12
18	14	12	6
19	21	12	7
20	15	13	14
21	10	15	22
22	11	14	20
23	21	16	9
24	10	15	8
25	21	8	17
26	12	9	13
27	23	7	19
28	15	12	17
29	21	20	18

Task execution times on VMs

Figure 7. Second problem instance: DAG with 30 tasks, 3 VMs and a matrix detailing the computation times for each task on the different VMs

4.2. Computational results

As described before, we developed two dynamic meta-heuristic algorithms DPSO and DGA to solve the problem. The parameter values of DPSO and DGA have been set as shown in Tables 1 and 2, respectively. The number of iterations for each method is $itermax = 500$ for the first instance and $itermax = 1,000$ for the second one. The population size is $psize = 30$ for the first instance and $psize = 50$ for the second one. Table 3 provides the voltage values and corresponding frequencies for each VM under consideration. For fuzzy logic we decided to take $\alpha_{ij} = 0.5$ and $\beta_{ij} = 0.2$ for $i, j \in \{0, 1, \cdots, n\}$.

Table 1. DPSO parameters

Parameters	Value
w	$(\omega_{max} - ((\omega_{max} - \omega_{min}) * it))/it_{max}$
$q1, q2$	Generated at random

Where $iter_{max}$ refers to the total number of iterations, it indicates the current iteration, ω_{max} represents the maximal value of of the parameter ω ($\omega_{max} = 0.9$) and ω_{min} refers to the minimal value of the parameter ω ($\omega_{min} = 0.4$).

Table 2. DGA parameters

Parameters	Value	Description
P_{cr}	0.5	Crossover operation probability
P_{mut}	0.01	Mutation operation probability

Table 3. VM frequency and voltage

	VM1	VM2	VM3
Voltage (V)	1.1	1.3	1.5
Frequency (GHz)	2.0	2.5	3.0
Lowest voltage (V)		0.7	
Lowest frequency (GHz)		0.1	

We measured the efficiency of our approach by evaluating it with HEFT algorithm, which is widely recognized in the literature (see [1]). The implementation of all algorithms was carried out using the Java programming language. The algorithms were implemented in Java and executed on a system featuring an Intel Core i5-5200U processor running at 2.20 GHz and 8 GB of RAM.

4.3. Discussion

4.3.1. Evaluation of our first challenge

The first challenge focuses on reducing the makespan and minimizing energy consumption in the cloud, simultaneously and equitably. In Table 4 we presented the comparison of the three algorithms HEFT, DPSO an DGA for each instance of the problem without using the fuzzy logic approach. The time in the fourth column denotes the execution time (in ms) of each algorithm on our machine.

The results show that our two algorithms, DPSO and DGA, significantly outperform HEFT in terms of energy consumption. In particular, DGA stands out as the most efficient, consistently reducing energy consumption across all tested instances. Although HEFT achieves a slightly lower makespan, the difference is minimal, confirming that our approach offers a suitable balance between execution performance and energy efficiency.

4.3.2. Evaluation of the second challenge

To assess the usefulness of the fuzzy approach, we presented the results (i) considering the fuzzy approach and (ii) without considering the fuzzy approach. Thus, Table 5 presented the comparison of DGA performance for the two instances of problem with and without using the fuzzy logic approach. In this table, we denoted by Poss_DGA the result by DGA when we used possibility measure and by Nec_DGA the result by DGA when we used necessity measure.

The results showed that this approach is suitable for making a more realistic decision. Indeed, in the second case for instance, unlike the approach without fuzzy logic (i.e. DGA), which sets the makespan at 321 and the energy consumption at 2043.54, the fuzzy logic approach introduced a certain flexibility. It provided a range of values: the makespan varies between 316 and 322.99, and the energy consumption between 2043.54 and 2049.249, depending on whether an optimistic or pessimistic perspective is adopted in the decision-making process. The same remarks are observed for the first instance. As a result, we observe that our DGA algorithm performed better in addressing this scheduling problem. It achieved the main objective of this paper, namely, optimizing energy efficiency in cloud computing, with only a slight degradation of the secondary objective, which is makespan. Moreover, the results obtained through the fuzzy logic approach assisted decision-makers in reaching more relevant and realistic decisions. In summary, our work offered two major contributions:

- A substantial reduction in energy consumption compared to traditional approaches such as HEFT.
- The integration of fuzzy logic, enabling more realistic and uncertainty-aware decision-making.

Table 4. Comparison of HEFT, DPSO, and DGA performance for the two instances of problem

Instance	Algorithm	Makespan (ms)	Energy (J)	Time (ms)
First instance	HEFT	**75**	685	0.151
	DPSO	88.0	**657.48**	4.43
	DGA	88.0	**657.48**	5.832
Second instance	HEFT	**304**	2293.36	0.291
	DPSO	346.0	2161.20	4.187
	DGA	321	**2043.54**	6.983

Table 5. Comparison of DGA performance for the two instances of problem with and without using the fuzzy logic approach

Instance	Algorithm	Makespan (ms)	Energy (J)	Time (ms)
First instance	DGA	88.0	657.48	5.832
	Poss_DGA	86.5	657.48	5.9
	Nec_DGA	88.6	657.48	5.95
Second instance	DGA	321	2043.54	6.983
	Poss_DGA	316.0	2049.249	7.39
	Nec_DGA	322.99	2043.54	7.983

5. CONCLUSION

In this study, we have focused on the workflow scheduling optimization problem, which is essential to the performance of cloud computing. To do this, we have introduced a dynamic multi-objective modeling of the problem aiming at increasing both the makespan and the energy consumption. To handle this multi-objective problem, we use dynamic weights to transform it into a single-objective one, ensuring an equitable treatment of all objectives and automating the choice of weights. To address the situation where the CT is uncertain and not exact, we introduced a fuzzy model in which each CT is considered as a fuzzy number, and we used both possibility and necessity metrics. We have proposed two dynamic meta-heuristic algorithms (DPSO and DGA) to solve this problem and compared them with the HEFT algorithm. The results found demonstrated that the proposed algorithms are more efficient in minimizing the energy consumption compared to the HEFT algorithm, in particular DGA which minimized the energy consumption with a slight deterioration of makespan. As research perspectives, we can, on the one hand, improve the current model by integrating additional objectives such as cost or reliability, key criteria in cloud environments. On the other hand, we can explore hybrid approaches exploiting the complementary strengths of different metaheuristics, or integrate machine learning techniques to more efficiently guide the search process towards optimal solutions.

FUNDING INFORMATION

Authors state no funding involved.

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

REFERENCES

[1] A. Kumar, S. Ghosh, B. B. Naik, and P. Kuila, "Energy efficient workflow scheduling in cloud computing systems using particle swarm optimization," in *Proceedings of the International Conference on Signal Processing and Computer Vision (SIPCOV 2023)*, 2024, pp. 266–278, doi: 10.2991/978-94-6463-529-4_24.

[2] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng, "Q-learning based dynamic task scheduling for energy-efficient cloud computing," *Future Generation Computer Systems*, vol. 108, pp. 361–371, Jul. 2020, doi: 10.1016/j.future.2020.02.018.

[3] Y. Wang and X. Zuo, "An effective cloud workflow scheduling approach combining PSO and Idle time slot-aware rules," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 5, pp. 1079–1094, May 2021, doi: 10.1109/JAS.2021.1003982.

[4] H. R. Faragardi, M. R. S. Sedghpour, S. Fazliahmadi, T. Fahringer, and N. Rasouli, "GRP-HEFT: a budget-constrained resource provisioning scheme for workflow scheduling in IaaS clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1239–1254, Jun. 2020, doi: 10.1109/TPDS.2019.2961098.

[5] Q. Wu, M. Zhou, and J. Wen, "Endpoint communication contention-aware cloud workflow scheduling," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 2, pp. 1137–1150, Apr. 2022, doi: 10.1109/TASE.2020.3046673.

[6] C. Lu, J. Zhu, H. Huang, and Y. Sun, "A multi-hierarchy particle swarm optimization-based algorithm for cloud workflow scheduling," *Future Generation Computer Systems*, vol. 153, pp. 125–138, 2024, doi: 10.1016/j.future.2023.11.030.

[7] Z. Zhu, G. Zhang, M. Li, and X. Liu, "Evolutionary multi-objective workflow scheduling in cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1344–1357, May 2016, doi: 10.1109/TPDS.2015.2446459.

[8] X. Tang, "Reliability-aware cost-efficient scientific workflows scheduling strategy on multi-cloud systems," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2909–2919, Oct. 2022, doi: 10.1109/TCC.2021.3057422.

[9] Z.-G. Chen *et al.*, "Multiobjective cloud workflow scheduling: a multiple populations ant colony system approach," *IEEE Transactions on Cybernetics*, vol. 49, no. 8, pp. 2912–2926, Aug. 2019, doi: 10.1109/TCYB.2018.2832640.

[10] A. Iranmanesh and H. R. Naji, "DCHG-TS: a deadline-constrained and cost-effective hybrid genetic algorithm for scientific workflow scheduling in cloud computing," *Cluster Computing*, vol. 24, no. 2, pp. 667–681, Jun. 2021, doi: 10.1007/s10586-020-03145-8.

[11] R. K. Jena, "Multi objective task scheduling in cloud environment using nested PSO framework," *Procedia Computer Science*, vol. 57, pp. 1219–1227, 2015, doi: 10.1016/j.procs.2015.07.419.

[12] A. Kumar, S. Ghosh, B. B. Naik, and P. Kuila, "Energy efficient workflow scheduling in cloud computing systems using particle swarm optimization," in *International Conference on Signal Processing and Computer Vision (SIPCOV-2023)*, Atlantis Press, 2024, pp. 266–278.

[13] A. Verma and S. Kaushal, "A hybrid multi-objective particle swarm optimization for scientific workflow scheduling," *Parallel Computing*, vol. 62, pp. 1–19, Feb. 2017, doi: 10.1016/j.parco.2017.01.002.

[14] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, and S. Hu, "Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT," *Future Generation Computer Systems*, vol. 93, pp. 278–289, Apr. 2019, doi: 10.1016/j.future.2018.10.046.

[15] J. J. Durillo and R. Prodan, "Multi-objective workflow scheduling in Amazon EC2," *Cluster Computing*, vol. 17, no. 2, pp. 169–189, Jun. 2014, doi: 10.1007/s10586-013-0325-0.
[16] Y. Hao, C. Zhao, Z. Li, B. Si, and H. Unger, "A learning and evolution-based intelligence algorithm for multi-objective heterogeneous cloud scheduling optimization," *Knowledge-Based Systems*, vol. 286, 2024, doi: 10.1016/j.knosys.2024.111366.
[17] H. Yuan, H. Liu, J. Bi, and M. Zhou, "Revenue and energy cost-optimized biobjective task scheduling for green cloud data centers," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 2, pp. 817–830, Apr. 2021, doi: 10.1109/TASE.2020.2971512.
[18] A. Rehman, S. S. Hussain, Z. ur Rehman, S. Zia, and S. Shamshirband, "Multi-objective approach of energy efficient workflow scheduling in cloud environments," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 8, Apr. 2019, doi: 10.1002/cpe.4949.
[19] P. V. Reddy and K. G. Reddy, "An energy efficient RL based workflow scheduling in cloud computing," *Expert Systems with Applications*, vol. 234, p. 121038, Dec. 2023, doi: 10.1016/j.eswa.2023.121038.
[20] M. Adhikari, T. Amgoth, and S. N. Srirama, "Multi-objective scheduling strategy for scientific workflows in cloud environment: a Firefly-based approach," *Applied Soft Computing*, vol. 93, p. 106411, Aug. 2020, doi: 10.1016/j.asoc.2020.106411.
[21] I. Behera and S. Sobhanayak, "Task scheduling optimization in heterogeneous cloud computing environments: a hybrid GA-GWO approach," *Journal of Parallel and Distributed Computing*, vol. 183, p. 104766, Jan. 2024, doi: 10.1016/j.jpdc.2023.104766.
[22] D. Wu, Z. Li, H. Shi, P. Luo, Y. Ma, and K. Liu, "Multi-dimensional optimization for collaborative task scheduling in cloud-edge-end system," *Simulation Modelling Practice and Theory*, vol. 141, p. 103099, May 2025, doi: 10.1016/j.simpat.2025.103099.
[23] L. Abualigah *et al.*, "Improved synergistic swarm optimization algorithm to optimize task scheduling problems in cloud computing," *Sustainable Computing: Informatics and Systems*, vol. 43, p. 101012, Sep. 2024, doi: 10.1016/j.suscom.2024.101012.
[24] B. M. H. Zade, N. Mansouri, and M. M. Javidi, "An improved beluga whale optimization using ring topology for solving multi-objective task scheduling in cloud," *Computers & Industrial Engineering*, vol. 200, p. 110836, Feb. 2025, doi: 10.1016/j.cie.2024.110836.
[25] H. Abd and E.-W. Khalifa, "Developing a new fuzzy approach for solving two-machine flow shop scheduling problems under fuzziness," *Computational Algorithms and Numerical Dimensions*, vol. 2, no. 4, pp. 195–204, 2023, [Online]. Available: https://www.journal-cand.com/article_194229.html.
[26] C.-C. Chyu and W.-S. Chang, "Optimizing fuzzy makespan and tardiness for unrelated parallel machine scheduling with archived metaheuristics," *The International Journal of Advanced Manufacturing Technology*, vol. 57, no. 5–8, pp. 763–776, Nov. 2011, doi: 10.1007/s00170-011-3317-3.
[27] Junqing Li, Shengxian Xie, Tao Sun, Yuting Wang, and Huaqing Yang, "Solving fuzzy job-shop scheduling problem by genetic algorithm," in *2012 24th Chinese Control and Decision Conference (CCDC)*, May 2012, pp. 3243–3247, doi: 10.1109/CCDC.2012.6244513.
[28] M. Gabli, E. M. Jaara, and E. B. Mermri, "A genetic algorithm approach for an equitable treatment of objective functions in multi-objective optimization problems," *IAENG International Journal of Computer Science*, vol. 41, no. 2, pp. 102–111, 2014.

BIOGRAPHIES OF AUTHORS

Ayoub Chehlafi is a Ph.D. student in the Department of Computer Science at the FSO, UMF, Oujda, Morocco. His research focuses on artificial intelligence, machine learning, and optimization techniques. He has knowledge in applied AI, metaheuristic algorithms, optimization network, cloud computing, and energy consumption. His work aims to contribute to advancements in intelligent computing and efficient resource management. He can be contacted at email: ayoub.chehlafi@ump.ac.ma.

Mohammed Gabli is an associate professor of Computer Science in the Department of Computer Science at the FSO, UMF, Oujda, Morocco. He is a former guidance counselor at the Moroccan Ministry of National Education. Currently, he is coordinator of the master's degree in Artificial Intelligence and Data Sciences. His research interests include artificial intelligence, metaheuristics, optimization, and education. He has several publications in the field of artificial intelligence. He can be contacted at email: medgabli@yahoo.fr or medgabli@ump.ac.ma.