Task Scheduling in Cloud Computing Based on FPA Metaheuristic Algorithm

Arslan Nedhir Malti*, Badr Benmammar*, and Mourad Hakem†

*LTT Laboratory of Telecommunication Tlemcen, UABT, Tlemcen, Algeria
{arslannedhir.malti, badr.benmammar}@univ-tlemcen.dz
†DISC Laboratory, Femto-ST Institute, UMR CNRS, Université de Franche-Comté, Besançon, France
mourad.hakem@univ-fcomte.fr

Abstract—**In this paper, we propose to address the multi-objective task scheduling issue in cloud computing system in order to find the best assignment of the user-submitted tasks to the cloud resources provided by the cloud service providers. To achieve a high overall performance for computational cloud resources, a good scheduler algorithm is required to meet customer requirements and ensure efficient QoS. To this end, the Flower Pollination Algorithm metaheuristic has been used to orchestrate the trade-offs relationship between three end-user QoS-based criteria, namely reliability, time makespan and execution cost. The results obtained from the experiment conducted with the CloudSim simulator clearly show the interest of our approach based on FPA.**

Index Terms—**Cloud computing, Multi-objective optimization , QoS, FPA, CloudSim.**

I. Introduction

Cloud computing (CC) is a large-scale distributed computing system that offers the distribution of different types of services and resources to customers. In other words, cloud service provides flexible, dynamic, scalable and on-demand computing services to cloud users according to the flexible pay-as-you-go payment plan. These network services have enabled many companies to benefit from global connectivity, as it allows their employees and customers to access the nearest endpoint.

Although the concept of CC is widely used today, due to the various services offered to users according to their specific requirements, there are still many problems that should be investigated in order to tackle new constraints and limitations. In recent years, an efficient scheduling strategies widely regarded as as one of these significant challenges that has attracted the interest of researchers. This problem is among the multi-objective optimization issues [1], as it has several complex and often conflicting targets. Thus, it is regarded as NP-hard problem in its general form. Deterministic algorithms are ineffective in solving such optimization problems they may get stuck in local minima and are unable to reach the global optimum. However, a major challenge is to find a good trade-off that satisfy all the objectives through the use of advanced optimization methods such as heuristics and meta-heuristics [2].

The purpose of task scheduling in this context, aims to orchestrate the optimal assignment of submitted tasks to the computational resources made available by the Cloud Service Providers (CSPs) on the basis of the varied end-users functional requirements. Despite the fact that several studies have been proposed in the literature, the majority of them have mainly concentrated on the optimization of a single Quality of Service (QoS), which is the makespan time believed to be the only primary concern [3].

To overcome these limitations, the multi-objective task scheduling in CC systems has been tackled in this study. To accomplish this, the Flower Pollination Algorithm (FPA) metaheuristic has been chosen and adaptes in a multi-objective setting to acheive an efficient assignment of submitted tasks to the set of VMs within a reasonable time, based on three conflicting QoS criteria namely execution cost, time makespan, and reliability.

In this work, the effectiveness of the FPA's solutions is evaluated using two variants of the compromise function: the Pareto optimality concept [4] based on the TOPSIS algorithm [5], and on the other hand, the weighted sum method, the most popular and well-known in optimization processes due to its simplicity. It merges the different quality of service parameters into a single-objective.

The rest of this paper is organized as follows. Section 2 offers a review of the relevant approaches in the literature. Section 3 describes our approach which is made upon the pollination behavior of the flowers. We report in Section 4, series of experimental results to assess the performances of the proposed algorithm according to criteria that we have defined for this purpose. Finally, some concluding remarks are made in Section 5.

II. Related works

Chakravarthi and Shyamala [6] have proposed a new approach to schedule dynamic concurrent workflows in cloud computing environments. The proposed technique which is called TOPSIS inspired Budget and Deadline Aware Multi-Workflow Scheduling (T-BDMWS) seeks to minimize execution cost, time makespan and improves VM's resource utilization while guaranteeing the deadline and the budget constraints specified by the user. A weighted sum of cost, makespan and data transfer time is used to determine the best resource among the available resources based on the task requirements. The effectiveness of the T-BDMWS was compared with four well-known existing algorithms such as Cloud-based Workflow Scheduling Algorithm (CWSA), Budget and Deadline Constraint Heterogeneous Earliest Finish Time (BDHEFT) and Budget-Heterogeneous Earliest Finish Time (BHEFT). The simulation results performed with CloudSim toolkit reveal that the author's proposal provides better results in terms of makespan and cost-effective schedules.

In the work of Medara and Singh [7], an energy-efficient and reliability aware workflow scheduling in a cloud environment (EERS) algorithm is presented, which optimize the reliability of task workflows while saving energy consumption. The performance of the proposed technique was evaluated using two real-world scientific workloads Montage and CyberShake, and the numerical results show that this approach surpasses the related existing approaches, namely HEFT [8], EES [9], and REEWS [10].

By modeling the IaaS cloud and task workflows, Han et al. [11] have developed an efficient heuristic named CMSWC (Cost and Makespan Scheduling of Workflows in the Cloud) that aims to minimize execution cost and makespan of the workflows simultaneously. The CMSCW algorithm follows a two-phase scheduling: ranking and mapping. The latter is designed to avoid exploring unnecessary resources for tasks, which significantly reduces the search space. An effective resource selection policy and optimized solution selection strategy are designed by combining two approaches: the quick non-dominated sorting approach and the Shift-Based Density Estimation (SDE) based crowding distance in order to make the solutions close to Pareto front. Extensive experiments on real-life workflows demonstrate that this strategy has better performances in terms of makespan-cost tradeoff compared to the concurrent approaches: FDHEFT [12] , NSGA-II [13] and MODE [14] for all the tested scenarios.

The authors in [15] have designed a new framework as whale optimizer algorithm (WOA) which mimics the social behaviour of humpback whales. The main idea of the presented work is to improve the workflow scheduling constraints and balance the load among the used resources. The performances of proposed WOA was evaluated as a multi-objective optimization problem, measured in terms of makespan, deadline hit and resource utilization. The authors' proposal performed well compared to other existing techniques such as Gray Wolf Optimizer (GWO), PSO, ACO, GA.

A novel optimization techniques which is an improvement of the basic SFLA algorithm is introduced by Kaur et al. [16]. The proposed approach which is called Augmented Shuffled Frog Leaping Algorithm (ASFLA) determines a resource provisioning and workflow scheduling strategy in IaaS cloud environment. The objective is to optimize the running cost of the application while meeting the deadline constraint. It was showen that this technique is able to reduce the overall execution cost compared to those of Shuffled Frog Leaping Algorithm (SFLA) and PSO algorithm. However, this reduction leads to an increase in execution time. As a result, it can be conjectured that ASFLA is more appropriate when minimizing the execution cost is the main concern.

Mansouri et al. [17] have made a proposal of a new independent task scheduling technique named FMPSO. It is based on a fuzzy theory and a modified particle swarm optimization to improve both load balancing and throughput in cloud environments. The experiment results proved that the proposed technique has the ability to reduce the total execution time, makespan and increase the resource utilization efficiently. It is noteworthy that the FPA algorithm was previously applied for this problem [18]–[20]. However, they differ from our study in the way they adapt and evaluate the generated solutions, as well as data set and different parameters were used.

III. Adaptation of the FPA metaheuristic

To achieve an efficient multi-objective optimization for the task scheduling problem in a CC environment, we consider three conflicting QoS criteria listed below.

- **Makespan time:** it reflects the interval between the last task's completion time and the first task's submission time.
- **Cost:** is the sum of the resource running costs for all submitted users' tasks. The unit cost of the resource that was chosen in accordance with the scheduling algorithm and the task's execution time on that resource are both factors that affect the task execution cost.
- **Reliability:** it reflects the likelihood that a given task will be accomplished successfully. Our reliability measurement model is based on a failure rate according to the following formula [21].

$$Reliability = \exp^{-\sum_{i=1}^{n} TE(T_i)*\lambda_j} \quad (1)$$

Where, $TE(T_i)$ is the task's execution time T_i and λ_j is the failure rate of the selected virtual machine which performs the task.

The aforementioned Qos criteria are divided into two groups; the former aims to maximizes the benefit criteria (reliability), whereas the later seek to minimize the cost and makespan time criteria. In this context, a comparison was made with the weighted sum method that combines these three objectives considered into a single objective for the minimization problem, as shown in equation (eq.2).

$$F = w_1 * Makespan + w_2 * Cost + w_3 * \left(\frac{1}{Reliability}\right) \quad (2)$$

where $w = \{w_1, w_2, w_3\}$ is the weight vector that depicts the user's importance or requirement for each criterion.

In this work and in the field of metaheuristics, we takes a new look on the flower pollination algorithm. It was proposed in 2012 by Xin-She Yang [22], taking its metaphor with the characteristics of flowering plants pollination.

The pollination process is necessary to reproduce flower plants. Therefore, FPA algorithm has two main key processes namely, cross-pollination and self-pollination. For further details, cross-pollination or global pollination accomplished when pollen is transferred to a flower from another plant by intervention of pollinators according to Levy's flight behavior [23]. In contrast, the self-pollination also called local pollination, takes place when pollen from one flower is used to pollinate another flowers or the same flower within the same plant with the aid of environmental factors [22].

Now we can idealize the characteristics of the pollination process, flower constancy and pollinator behavior based on four main rules listed as follows:

- **Rule 1:** Biotic, cross-pollination acting as a global pollination process via the Levy flight.
- **Rule 2:** Abiotic and self-pollination are considered local pollination.
- **Rule 3:** Consistency of flowers may be involved due to the similarity of two flowers involved.
- **Rule 4:** Global as well as the process of local pollination are controlled by a switching probability $p \in [0, 1]$.

Based on the above four rules, the basic steps of the adapted FPA based approach for task scheduling in cloud

Algorithm 1 Adapted FPA

```
1: Set the switch probability.
2: for each task $T_i$ in the cloudlets list do
3:    Generate the initial population with random VMs
4:    Determine the best Machine $V_{best}$ for the task $T_i$
5:    while stop criterion do
6:       for each machine $V_i$ in the population do
7:          if $rand() < p$ then
8:             Perform Levy flight and draw $L$ (step vector)
9:             Perform global pollination via
               $V_i^{'} = V_i + L(V_i - V_{best})$
10:         else
11:            Draw $\varepsilon$ from a uniform distribution in the range $[0, 1]$
12:            Choose randomly two machines $V_j$ and $V_k$ within all solutions.
13:            Perform local pollination as
14:         end if
15:         Calculate the fitness of the new virtual machine $V_i^{'}$
16:         if new generated machine is better then
17:            replace $V_i$ with $V_i^{'}$
18:         end if
19:      end for
20:      Update $V_{best}$ (the best solution found so far)
21:   end while
22: end for
```

computing environments are outlined in Algorithm 1. The notations used are shown in TABLE I with their description.

TABLE I
NOTATION USED

Notations	Descriptions
p	Switching probability of FPA
T_i	The i^{th} task
V_i	The i^{th} virtual machine
$V_i^{'}$	The i^{th} new generated virtual machine
V_{best}	Best virtual machine in the population
L	Step size based on Levy flight law

IV. Simulation results and analysis

The CloudSim simulator was chosen for its modeling qualities to simulate our work [24]. The latter is a generalized toolkit. It is made up of a set of Java class libraries to provide reliable, secure, fault-tolerant and scalable computing services. Moreover, it also covers the behavior of CC components such as data centers, computing resources, virtual machines, tasks (cloudlets) and brokers in order to assess the effectiveness of new algorithms and applications.

For the proposed FPA algorithm, two specific parameter settings must be defined: the population size and the switching probability p which control respectively the proportions of *exploitation* and *exploration* during the search process. In simulations performed. we took a switching probability equal to $p = 0.8$ as given in [22]. we used 20% of created virtual machines as the population size since this percentage provided us better performances.

The simulations were conducted in heterogeneous cloud computing environment. All parameters of the simulations are shown in TABLE II.

Since metaheuristic algorithms are characterized by randomization, the designed algorithm is not deterministic, i.e. a single run is not sufficient to draw conclusions. Therefore, we propose to compute an average of 10 simulations for each type of scheduling in order to make the results more representative.

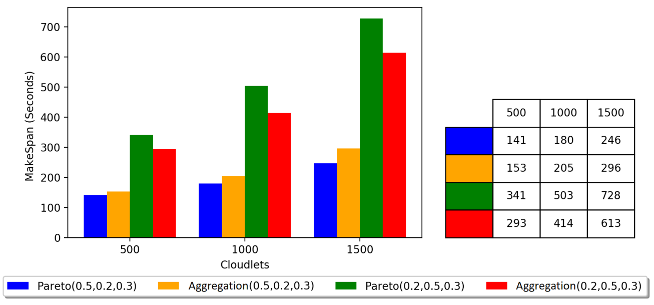
To determine the best compromise solutions, a comparison was made using two versions for the evaluation of the FPA algorithm solutions, namely the weighted sum method by aggregating the three considered objectives in accordance with the previously stated objective function

TABLE II
AN EXPERIMENTAL SETTINGS

Data-center	No. of Data-centers	4
Host	No. of Hosts	8
	PES	4
	MIPS	6000
	RAM	20 GB
	Bandwidth	10 GB
	Storage	1 TB
Virtual Machine	No. of VMs	30 - 50
	MIPS	1000 to 5000
	Cost	1.0 to 30.0
	Failure rate	10^{-5} to 10^{-7}
	RAM	1 to 5 GB
	Storage	10 GB
	Bandwidth	100 to 500 MB
Cloudlets	No. of cloudlets	500 - 1000 - 1500
	Length	3000 to 10000
	Type	Heterogeneous
	Submission time	Poisson distribution of parameter λ

(eq.2) and the Pareto principle based on the TOPSIS multi-criteria technique.

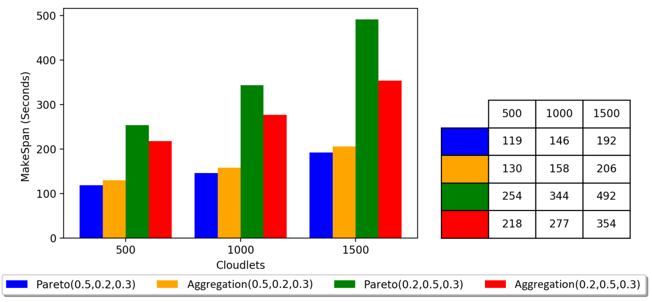
In order to better serve users' demands and provide a better QoS according to their needs, we have selected two vectors for each version, with different weights $\{0.5, 0.2, 0.3\}$ and $\{0.2, 0.5, 0.3\}$, corresponding respectively to the targeted objectives {Makespan, Cost, Reliability}. The first vector favors the makespan time while the second vector favors the execution cost objective.



	500	1000	1500
🟦	141	180	246
🟧	153	205	296
🟩	341	503	728
🟥	293	414	613

Fig. 1. Comparison of makespan with 30 VMs.

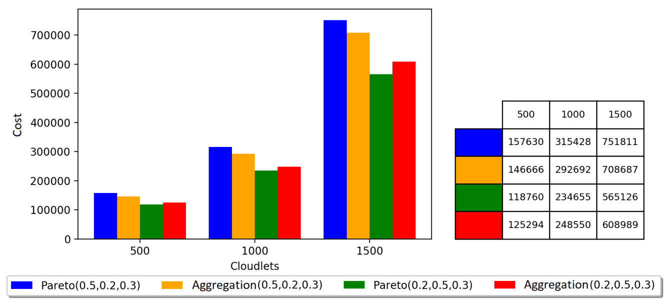
First, we investigate the makespan values obtained by the compared approaches. We can observe from Figs. 1 and 2 that, the makespan aim increases as the tasks number increases. Given that the resources' ability to execute the tasks gradually declines as the



	500	1000	1500
🟦	119	146	192
🟧	130	158	206
🟩	254	344	492
🟥	218	277	354

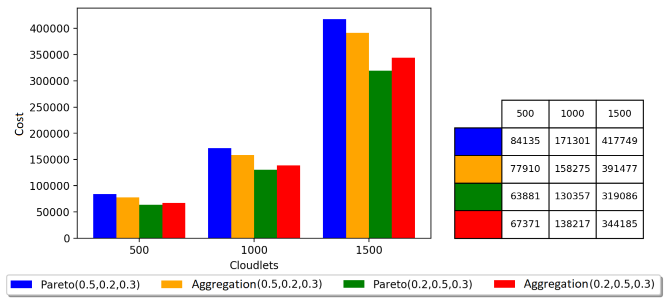
Fig. 2. Comparison of makespan with 50 VMs.

load grows. In fact, the Pareto-TOPSIS based technique performs best in terms of makespan with a weight vector $\{0.5, 0.2, 0.3\}$ which favors the makespan objective. However, the aggregation minimizes well with weight vector $\{0.2, 0.5, 0.3\}$ which favors the cost objective. This result shows that Pareto-TOPSIS based technique and Pareto dominance concept alleviates this problem by finding better trade-off solutions between all considered objectives.



	500	1000	1500
🟦	157630	315428	751811
🟧	146666	292692	708687
🟩	118760	234655	565126
🟥	125294	248550	608989

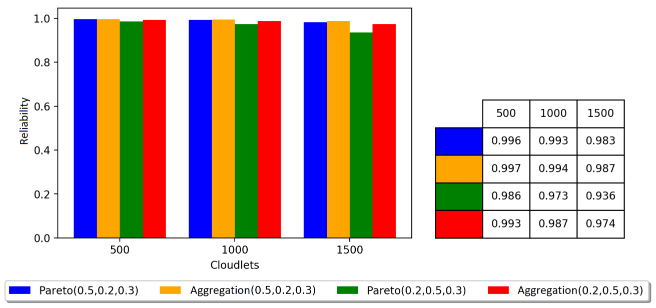
Fig. 3. Comparison of the execution cost with 30 VMs.



	500	1000	1500
🟦	84135	171301	417749
🟧	77910	158275	391477
🟩	63881	130357	319086
🟥	67371	138217	344185

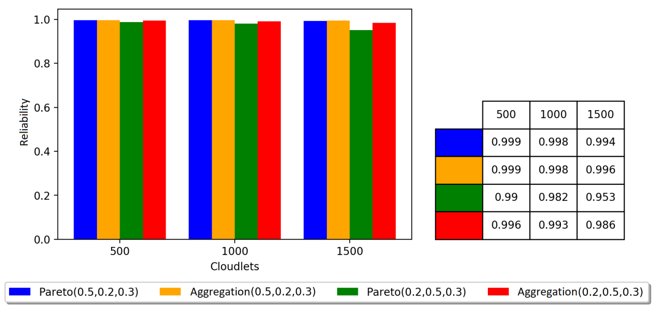
Fig. 4. Comparison of the execution cost with 50 VMs.

Comparing the results in terms of execution cost, it can be seen in Figs. 3 and 4 that the Pareto-TOPSIS approach has higher performances with a weighting vector that favors execution cost. This is due to the fact that, in contrast to the aggregation method, the latter seeks to ensure the three objectives considered in light of their importance. However, in terms of reliability optimization, Figs. 5 and 6 show that regardless of the number of tasks and the number of VMs, there is a relatively small difference in the performance obtained of the compared approaches.



	500	1000	1500
🟦	0.996	0.993	0.983
🟧	0.997	0.994	0.987
🟩	0.986	0.973	0.936
🟥	0.993	0.987	0.974

Fig. 5. Comparison of reliability with 30 VMs.



	500	1000	1500
🟦	0.999	0.998	0.994
🟧	0.999	0.998	0.996
🟩	0.99	0.982	0.953
🟥	0.996	0.993	0.986

Fig. 6. Comparison of reliability with 50 VMs.

Taken as a whole, it can be concluded that the findings of our research highlight that a purely multi-objective approach such as Pareto-TOPSIS with the adapted weight vector is more recommended. Indeed, the weighted sum method is a simple projection from the multi-objective to the mono-objective. Concerning reliability, this is a rather special target since existing CSPs offer services with too low failure rates that can go down to 10^{-7}. For this reason, the used method does not have a great effect on this objective.

V. Conclusion

In this paper, we proposed a multi-objective optimization approach for task scheduling in cloud computing. We have implemented the FPA metaheuristic with the Pareto principle based on TOPSIS to select the most compromising solution from the Pareto front. Comparing the results with the weighted sum method, particularly for the privileged criterion, the results obtained are more intriguing for the Pareto-TOPSIS-based strategy, which aims to give users a better QoS in accordance with their needs defined in the SLA contract.

References

[1] E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, "Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends," *Swarm and Evolutionary Computation*, p. 100841, 2021.

[2] P. Pirozmand, A. A. R. Hosseinabadi, M. Farrokhzad, M. Sadeghilalimi, S. Mirkamali, and A. Slowik, "Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing," *Neural Computing and Applications*, pp. 1–14, 2021.

[3] A. Semmoud, M. Hakem, B. Benmammar, and J.-C. Charr, "Load balancing in cloud computing environments based on adaptive starvation threshold," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 11, p. e5652, 2020.

[4] B. Benmammar, Y. Benmouna, and F. Krief, "A pareto optimal multi-objective optimisation for parallel dynamic programming algorithm applied in cognitive radio ad hoc networks," *International Journal of Computer Applications in Technology*, vol. 59, no. 2, pp. 152–164, 2019.

[5] C.-L. Hwang and K. Yoon, "Methods for multiple attribute decision making," in *Multiple attribute decision making*. Springer, 1981, pp. 58–191.

[6] K. K. Chakravarthi and L. Shyamala, "Topsis inspired budget and deadline aware multi-workflow scheduling for cloud computing," *Journal of Systems Architecture*, vol. 114, p. 101916, 2021.

[7] R. Medara and R. S. Singh, "Energy efficient and reliability aware workflow task scheduling in cloud environment," *Wireless Personal Communications*, pp. 1–20, 2021.

[8] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE transactions on parallel and distributed systems*, vol. 13, no. 3, pp. 260–274, 2002.

[9] Q. Huang, S. Su, J. Li, P. Xu, K. Shuang, and X. Huang, "Enhanced energy-efficient scheduling for parallel applications in cloud," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*. IEEE, 2012, pp. 781–786.

[10] R. Garg, M. Mittal *et al.*, "Reliability and energy efficient workflow scheduling in cloud environment," *Cluster Computing*, vol. 22, no. 4, pp. 1283–1297, 2019.

[11] P. Han, C. Du, J. Chen, F. Ling, and X. Du, "Cost and makespan scheduling of workflows in clouds using list multiobjective optimization technique," *Journal of Systems Architecture*, vol. 112, p. 101837, 2021.

[12] 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, 2019.

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[14] A. K. A. Talukder, M. Kirley, and R. Buyya, "Multiobjective differential evolution for scheduling workflow applications on global grids," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 13, pp. 1742–1756, 2009.

[15] S. R. Thennarasu, M. Selvam, and K. Srihari, "A new whale optimizer for workflow scheduling in cloud computing environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 3, pp. 3807–3814, 2021.

[16] P. Kaur and S. Mehta, "Resource provisioning and work flow scheduling in clouds using augmented shuffled frog leaping algorithm," *Journal of Parallel and Distributed Computing*, vol. 101, pp. 41–50, 2017.

[17] N. Mansouri, B. M. H. Zade, and M. M. Javidi, "Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory," *Computers & Industrial Engineering*, vol. 130, pp. 597–633, 2019.

[18] I. Gupta, A. Kaswan, and P. K. Jana, "A flower pollination algorithm based task scheduling in cloud computing," in *International Conference on Computational Intelligence, Communications, and Business Analytics*. Springer, 2017, pp. 97–107.

[19] T. Bezdan, M. Zivkovic, M. Antonijevic, T. Zivkovic, and N. Bacanin, "Enhanced flower pollination algorithm for task scheduling in cloud computing environment," in *Machine Learning for Predictive Analysis*. Springer, 2021, pp. 163–171.

[20] N. K. Walia, N. Kaur, M. Alowaidi, K. S. Bhatia, S. Mishra, N. K. Sharma, S. K. Sharma, and H. Kaur, "An energy-efficient hybrid scheduling algorithm for task scheduling in the cloud computing environments," *IEEE Access*, 2021.

[21] S. Yassa, "Allocation optimale multicontraintes des workflows aux ressources d'un environnement cloud computing," Ph.D. dissertation, Cergy-Pontoise, 2014.

[22] X.-S. Yang, "Flower pollination algorithm for global optimization," in *International conference on unconventional computing and natural computation*. Springer, 2012, pp. 240–249.

[23] I. Pavlyukevich, "Lévy flights, non-local search and simulated annealing," *Journal of Computational Physics*, vol. 226, no. 2, pp. 1830–1844, 2007.

[24] Cloudsim: A framework for modeling and simulation of cloud computing infrastructures and services. Www.cloudbus.org/cloudsim/, Accessed: 2021-09-01.