Multi-objective Task Scheduling in Cloud Computing

Arslan Nedhir Malti[a,*], Mourad Hakem[b], Badr Benmammar[a]

[a] *LTT Laboratory of Telecommunication Tlemcen, UABT, Tlemcen, Algeria*
[b] *DISC Laboratory, Femto-ST Institute, UMR CNRS, Université de Franche-Comté, Besançon, France*

Abstract

Cloud computing services are used to fulfill user requests, often expressed in the form of tasks and their execution in such environments requires efficient scheduling strategies that take into account both algorithmic and architectural characteristics. Unfortunately, this problem is known to be NP-hard in its general form. Despite the fact that several studies have been published in the literature, there are still interesting and relevant questions to be addressed. Indeed, most of the previous studies focus on a single objective and in the case where they deal with a set of objectives, they use a simple compromise function and do not consider how each of the parameters might influence the others. To this end, we propose an efficient task scheduling algorithm which is based on the pollination behavior of flowers and makes use of both Pareto optimality principle and TOPSIS technique to improve the quality of the obtained solutions. Both single and multiobjective optimization variants are investigated. In the latter case, three optimization criteria are considered namely, minimizing the time makespan or schedule length, the execution cost, and maximizing the overall reliability of the task mapping. Different test-bed scenarios and QoS metrics were considered and the obtained results corroborate the merits of the proposed algorithm.

Keywords: Cloud computing, Task scheduling, Multiobjective optimization, FPA, Pareto optimality, TOPSIS technique

1. Introduction

A new technology which has emerged in recent years and has garnered significant interest for scientific applications is the cloud computing platform. It is a simple consumer-provider service model which permits flexible and scalable computer resources and permits their leasing on an elastic pay-per-use model without any geographical restrictions. A cloud user can access to these computing resources, which provide some services as on-demand utilities, with minimal management effort and interaction with the service provider. Most individuals, businesses and even government agencies are turning to cloud technology due to the benefits this new paradigm offers, including no upfront investment, reduced operating costs, rapid scalability, unlimited storage and ubiquitous accessibility.

To effectively harness the potential of the cloud system, cloud service providers (CSPs) such as Amazon AWS, Google, Microsoft, Adobe, Accenture, IBM, and Cisco strive to meet the diverse requirements of their clients by offering various services that generally fall into three main categories, namely Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Saas offers complete applications as an on-demand service, completely abstracted from the used hardware and software. It is widely employed for conventional cloud applications such as online e-mail, documents editing services, etc. PaaS provides a framework for developers in order to design or customize their own applications. The third type of service model IaaS is the lowest paradigm of cloud service. We mainly focus on the latter which consists of outsourcing the physical infrastructure of the IT service (networks, storage and servers) that can be rented by users according to their needs.

Although the concept of cloud computing is a widespread today, due to the various benefits and the different services offered to users according to their needs and contracts, there are many issues that need to be addressed by developing new technical solutions to handle new challenges and constraints. One of these main challenges, which reflects how well the tasks are managed, and serves as a basis for cloud's QoS, is task scheduling. This is the process of assigning appropriately user-submitted requests to the set of resources under specific constraints in IaaS environment. Further, due to the virtualization technology, the cloud gives the illusion of an infinite virtual machines (VMs) with different configurations. These VMs should be acquired or released from an IaaS provider based on the applications needs. Therefore, careful attention is needed for choosing the appropriate resources from a tremendous collection of computing resources in order to execute users' applications. Consequently, to achieve better performances and efficient utilization of these resources, novel and innovative scheduling policies should be designed and implemented.

*Corresponding author
Email address: arslannedhir.malti@univ-tlemcen.dz (Arslan Nedhir Malti)

In recent years, task scheduling problem has become a compelling process in cloud environments and has attracted many scientists. For instance, in [1, 2, 3], the authors have presented an overview of cloud computing and focused on the state-of-the-art research challenges in which task scheduling is a major concern. Unfortunately, this problem has been shown to be NP-Complete in its general form and heuristics and metaheuristices are needed to achieve near optimal solutions but in polynomial time complexity [4]. Moreover, several variants of this optimization problem have also been studied such as multi-objective optimization [5, 6, 7, 8] where various conflicting criteria need to be considered at the same time, not only to meet the different user demands, but also to improve the Quality of Service (QoS) and the overall system performance according to the Service Level Agreement (SLA). Due the fact that there are two primary entities in the cloud: cloud consumers and cloud service providers, cloud customers transmit their tasks to be processed and care more about the performance of their application, while cloud providers rent out their resources to cloud consumers and are more concerned with the efficient use of their resources in order to maximize revenues. Thus, QoS constraints can be divided into two main categories: service provider wishes and consumer desires.

Despite the fact that several works have been proposed by different researchers to address the scheduling problem, most of them have mainly focused on a single optimization objective such as time makespan [9], and when some existing works consider other quality of service parameters, these studies rely on a simple compromise function between the targeted objectives which do not ensure efficient trade-off solutions that satisfy all user objectives.

In this paper, the focus is on dependent tasks scheduling in IaaS cloud environment with heterogeneous resources and various requirements of end-users. To this end, an efficient task scheduling algorithm based on a Flower Pollination (FPA) metaheuristic is presented. This approach generates a variety of possible planning solutions for the set of submitted tasks, and its combination with both the Pareto based approach and the TOPSIS technique, improves the quality of the obtained solutions in contrast to what have been proposed in the literature. This allows the user to select the preeminent solution according to his preferences while taking into account the cloud characteristics. We formulate two tasks scheduling scenarios, depending on the number of parameters involved to measure the cloud consumers' QoS. The first is a single objective scheduling that aims to minimize the time makespan, while the second is a multiobjective scheduling optimization that attempts to simultaneously optimize three conflicting criteria, namely cost, makespan and reliability.

The main contributions and novelties of the presented study are summarized as follows:

1. We use a bio-inspired pollination behavior of flowers to design an efficient task scheduling algorithm in heterogeneous cloud computing environments.
2. A new multi-objective evaluation of the fitness function is proposed. It consists of first generating the set of all Pareto-optimal solutions and then choosing the most interesting solution using TOPSIS (Technique for Order Preference by Similarity to Ideal Solution). As far as we know, this study is the first to investigate the combination of the Pareto based approach as well as TOPSIS method with the pollination flower scheme to achieve high performances of the task scheduling optimization problem addressed in this work.
3. Both single and multi-objective task scheduling variants are investigated. In the former case, a single objective optimization is formulated in which the time makespan is considered as the only criteria. The obtained results were compared to the traditional methods, namely Round Robin, Max-Min and Min-Min. The second variant, which refers to multi-criteria optimization, aims to orchestrate the trade-offs relationship between cost, reliability and time makespan optimization and their impact on the global performances of the proposed algorithm.
4. Based on CloudSim framework, series of test-bed scenarios and QoS metrics were considered and the obtained results show that our proposal achieves good performances compared to its direct competitor, namely the aggregate weighted sum technique that has been used in [10].

The rest of this paper is organized as follows: Section 2 reviews the relevant techniques that have been proposed in the literature to deal with task scheduling in cloud computing environments. The detail descriptions of the task scheduling model and the problem statement are given in section 3. In Section 4, we present in details the proposed algorithm which is made upon the pollination behaviour of the flowers and the use of both Pareto optimality principle and TOPSIS technique. Section 5 outlines the key features of the mutli-criteria based WSM approach [10], which is, to our knowledge, the closest work to the one presented in this work. We report in Section 6, series of experimental results to asses the performances of our proposals. This paper concludes with a summary of the contributions and some future work directions in Section 7.

2. Related works

Several techniques have been proposed in the literature to deal with the problem of task scheduling in parallel and distributed network systems. The proposed methods are designed under various and non similar assumptions and have different metrics for end-users QoS requirements. Furthermore, most published multiobjective optimization algorithms transform the multiobjective problem into a single-objective problem by using a simple weighted sum approach for the considered objectives to assess the algorithm's performances.

Bezdan et al. [11] developed an independent task scheduling algorithm which is called BA-ABC. It is based on two meta-heuristic approaches Bat and Bee colony algorithms. The objective is to assign the submitted tasks to the virtual instance types by Bat's hybrid optimization algorithm, where the exploration phase of the Bat Algorithm (BA) is enhanced by the observer bee search of Artificial Bee Colony (ABC) algorithm. BA-ABC uses aggregation approach to optimize during the scheduling process the makespan and execution cost. The effectiveness of BA-ABC was compared to four other optimization algorithms and the obtained results showed that the authors' proposal has a relative advantage over the compared algorithms.

In the work of Sardaraz et al. [12], a hybrid algorithm based on Particle Swarm Optimization (PSO) for scheduling scientific workflows is proposed. The main idea of this algorithm is to reduce in a first step, the execution time in order to give higher priority to the tasks in queu list and reduce both makespan and execution cost in the second step. The algorithm also monitors load balancing for efficient use of cloud resources. The performance of the proposed algorithm is validated through numerical simulations with standard PSO, Genetic Algorithm (GA) and specialized scheduling approaches based on PSO technique such as PSO-DS [13] and GA-PSO [14]. A novel dynamic two-phase strategy based on Firefly Algorithm (FA) is proposed by Adhikari et al. [15] for workflows scheduling. Based on the aggregated approach, several conflicting objectives in IaaS cloud were considered such as cloud server workload, makespan, resource utilization and reliability. First, this proposed strategy seeks to find a best server for each workflow that can fulfill its requirements while balancing the loads and resource utilization. Then a policy-based job assignment is used to allocate the jobs to the appropriate set of VMs, which minimize the makespan of the workflow while increasing the reliability of the cloud servers.

In a recent paper, Pirozmand et al. [4] designed a new hybrid algorithm, called GAECS, combining GA and the Energy-Conscious Scheduling (ECS) model, which is a time- and energy-aware technique for multi-objective task scheduling in cloud computing systems. The authors' goal is to solve the above problem by assigning tasks to processors that allow better compromise between time makespan and energy consumption. The performance of the proposed GAECS algorithm is analyzed using MATLAB and shows that it outperforms other compared algorithms in terms of accuracy. In [16], the authors developed a multi-objective evolutionary algorithm called Many-Objective Genetic Algorithm Scheduler (MOGAS) to solve the container scheduling problem based on a Non-dominated Sorting Genetic Algorithm III (NSGA-III) [17]. The proposed scheduler distributes a batch of different tasks to a heterogeneous group of nodes. Various parameters such as availability, load balancing, resource utilization, energy, and maximum number of assigned tasks were considered and the experimental results corroborate the merits of the introduced algorithm compared to Ant Colony Optimization (ACO) scheduler.

Chakravarthi and Shyamala [18] 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 [19], 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 [20], EES [21], and REEWS [22].

By modeling the IaaS cloud and task workflows, Han et al. [23] 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 [24], NSGA-II [25] and MODE [26] for all the tested scenarios.

The authors in [27] 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. Abualigah and Diabat [28] have developed a new multi-objective task scheduling which makes use of AntLion Optimization (MALO) method. It uses local search

technique and Differential Evolution (DE) strategy in order to improve the exploitability of ALO and avoid being trapped in local optima. Practical results show that the authors' work presents a significant improvement not only in terms of resource utilization and makespan but also in terms of convergence speed of the optimization process.

A bi-objective model involving QoS and energy consumption in Cloud Manufacturing (CMfg) is proposed by Yang et al. [29]. It presents some improvements of original Multi-Objective Grey Wolf Optimizer (MOGWO) in order to increase the efficiency of finding solutions for the Multi-Objective Service Composition and Optimal Selection (MO-SCOS) problem. The authors use the aggregate weighted sum approach between a maximum QoS value and a low energy consumption during the service composition optimization process. The simulation results illustrated that the improved strategies have good effects on the performance compared to other multi-objective benchmark algorithms. To deal with heterogeneity and dynamicity in cloud systems, a new approach which relies on an existing technique called Shortest Job First (SJF) is presented in [30]. Tasks are scheduled to minimize the energy consumption, CPU usage and time makespan. The experimental study showed that the proposed approach (DHSFJ) Following the comparison performed between the authors' work and other concurrent techniques such as First Come First Serve (FCFS), SJF and its variant heterogeneous HSJF showed that the introduced method offers better performance in terms of makespan and energy consumption due to the heterogeneity of both resources and workload.

Menouer and Darmon [31] have designed a new scheduling method in Docker Swarm, by combining the Spread and the Bin Packing principles with TOPSIS technique. The aim of this strategy is to select among a set of nodes that form a cloud infrastructure, the most suitable node to execute each container of the submitted users' tasks, while achieving a good compromise between three criteria: the number of containers executed, the number of available CPUs and the size of the available memory. A hybrid bi-objective scheduling algorithm which combines Cuckoo Search (CS) and PSO is proposed in [32]. It seeks to reduce both cost and maksepan value to generate task-VM mapping in a heterogeneous cloud environment. It was shown through simulations that the proposed approach improves not only the time makespan and cost values but also the deadline violation rate. Samriya and Kumar [33] focused on energy, migration costs, and eminent resource utilization in cloud systems. They introduced a hybridization of PSO and Fuzzy TOPSIS for productive job scheduling. First, the available task and the number of VMs are optimized by the PSO algorithm. Then Fuzzy TOPSIS solves the multi-objective job scheduling problem using the weighted sum method for energy, cost and execution time as the objective function. The experimental results confirmed that the authors' scheduling approach provides higher QoS compared to other algorithms PSO and GA.

In recent years, flower pollination algorithm has attracted the interest of researchers in various types of complex problems such as in the field of engineering [34], geology [35], industries for process control [36], medical sector for EEG signal denoising [37], and so on. For instance, Gupta et al. [10] presented a bi-objective optimization for independent task scheduling on virtual resources in the cloud. It is based on FPA optimization algorithm, and compared to three other metaheuristic approaches PSO, GA and Gravitational Search Algorithm (GSA). It uses an efficient pollen representation scheme and a dedicated process to determine the task-VM mapping from a given pollen so that the makespan and the average cloud resource utilization are optimized. The simulation results illustrate that the task scheduling based on FPA is better than the compared concurrent metaheuristics approaches. However, the proposed work lacks in terms of dynamicity as it deals only with static independent task scheduling and virtual machines.

The authors, in [38], have used a flower pollination algorithm to deal with independent task scheduling in cloud systems. The proposed technique, which is called Exploration-Enhanced FPA (EEFPA), sustains QoS by considering only time makespan objective. This study reveal that the FPA approach is not able to discover the right section of the search space in the beginning, due to lack of exploration power. To alleviate this, the authors propose that in the first 30% of iterations, the worst individuals in the population are removed and replaced with a new random solution. Compared to other methods, it was shown that this technique is able to reduce the value of makespan and give a better convergence speed. Recently, Walia et al. [39] presented an energy-efficient scheduling algorithm (HS) which relies on both FPA and GA algorithms. The aim is to distribute the resources among tasks with less energy. The authors' study considers resource utilization, completion time, energy consumption, and cost of computation as performance metrics for homogeneous and heterogeneous cloud environments. The conducted simulation performed using ASP.NET tool showed that the proposed algorithm is able to produce efficient task scheduling and better resource management than existing algorithms like GA and FPA. However, the HS fairness index value is less than the prescribed threshold, which indicates a lack of fairness in resources allocation. Table 1 shows a comparative analysis of the existing state-of-art scheduling strategies recently published, which are designed with various optimization techniques.

To the best of our knowledge, none of the aforementioned research works have considered three-dimensional (makespan, cost, reliability) optimization by employing the flower pollination scheme combined with the use of both of Pareto optimality and TOPSIS multi-criteria technique as in this paper. The objective is to take advantage of each approach to improve the overall performances of the task scheduling process in cloud computing environments in terms of the achieved trade-offs between cost, time makespan and reliability.

Contributions	Scheduling Type	Parameters	Simulation tool	Year	Ref
A method based on BA and diversification of the ABC algorithm	Task scheduling	Makespan Execution cost	CloudSim	2020	[11]
A hybrid algorithm based on the PSO algorithm for scientific workflow scheduling	Workflow scheduling	Execution time Cost Load balancing	CloudSim	2019	[12]
A new dynamic strategy based on the Firefly algorithm	Workflow scheduling	Cloud server workload Makespan Resource utilization Reliability	N/A	2020	[15]
A new meta-heuristic method based on the GA and the ECS model	Task scheduling	Makespan Energy	MATLAB	2021	[4]
A multi-objective evolutionary algorithm implemented based on NSGA-III	Container scheduling	Load balancing Tasks assigned value Resource utilization Energy	Swarmkit	2020	[16]
A new approach to schedule dynamic concurrent workflows in cloud environments	Workflow scheduling	Makespan Cost Deadline Resource utilization	CloudSim	2021	[18]
An energy-efficient and reliability-aware scheduling algorithm	Workflow scheduling	Energy Reiliability	WorkflowSim	2021	[19]
A novel approach for optimizing cost and makespan simultaneously	Workflow scheduling	Makespan Cost	N/A	2021	[23]
New framework are introduced as whale optimizer algorithm	Workflow scheduling	Makespan Deadline Resource utilization	CloudSim	2021	[27]
The hybrid Antlion optimization algorithm with DE strategy	Task scheduling	Makespan Resource utilization	CloudSim	2021	[28]
Improvements of the original MOGWO	Manufacturing task scheduling	QoS value Energy	MATLAB	2020	[29]
An improved SJF algorithm to handle dynamic load in a heterogeneous cloud environment	Task scheduling	Makespan Energy CPU utilization	CloudSim	2019	[30]
Hybridization of the Spread and Bin Packing principles using the TOPSIS technique	Containers scheduling	Containers executed Available CPUs Available memory	Grid5000	2019	[31]
Hybridization of Cuckoo Search and Particle Swarm Optimization	Task scheduling	Makespan Cost Deadline	CloudSim	2019	[32]
Hybrid Fuzzy TOPSIS-PSO based scheduling approach	Task scheduling	Energy Cost Execution time	CloudSim	2020	[33]

Contributions	Scheduling Type	Parameters	Simulation tool	Year	Ref
An efficient pollen representation scheme to determine the task-VM mapping	Task scheduling	Makespan Resource utilization	MATLAB	2017	[10]
Enhanced FPA for Task Scheduling in Cloud	Task scheduling	Makespan	CloudSim	2021	[38]
An energy-efficient hybrid algorithm that relies on both FPA and GA algorithms	Task scheduling	Completion Time Ressource utilization Energy Cost	ASP.NET	2021	[39]

Table 1: An overview of the related works.

3. Problem statement

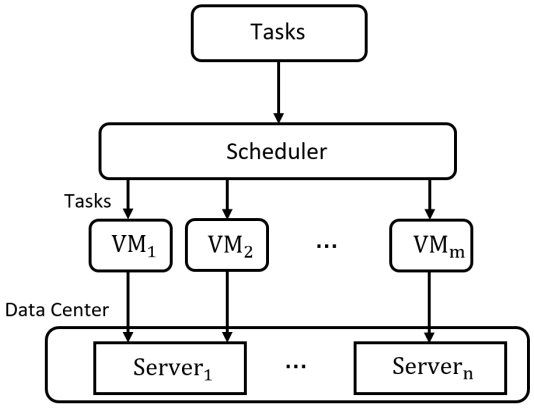


Figure 1: Task scheduler in cloud environment [1].

A typical cloud computing model is outlined in Figure 1. As illustrated, cloud service providers supply different types of virtual machine instances, distributed in servers or in physical machines depending on the availability of resources such as processing capacity, storage, memory, bandwidth and other necessary resources. In this model, users can lease VMs on demand according to their needs. To this purpose, cloud users initiate various service requests to run their applications, as a collection of tasks submitted to the scheduler through the CSP, that acts as a mediator between the cloud user and the cloud scheduler. The CSP provides all requested services to the users according to SLA contract. In simple terms, the task scheduling process is made upon the following three steps [40]:

- **Resource discovery:** the broker discovers all available resources in the system and collects the relevant information such as capacity, processing cost, etc.
- **Resource selection:** the most appropriate resource is selected based on the task requirements and the resource specifications.
- **Task submission:** the task is scheduled on the resource that was selected according to the scheduling algorithm decisions.

A task scheduler must also take into account the dependencies between tasks. A task can only be executed if its preceding tasks have completed their execution. In the presented study, a cluster scheduling is represented as follows: a set of clusters defines the tasks as an application to be executed. It consists of a variable number of tasks that are executed either sequentially or in parallel. The following assumptions are made:

(1) At any given time, each task must be processed on a single selected resource, and each resource can execute several tasks.

(2) Each cluster has different number of tasks. We note that there is no processing dependency between tasks in a cluster. In other words, tasks within a cluster can be executed in any order.

(3) The parent cluster must be executed before any other cluster begins execution. In other terms, the cluster designed by c_1 have to be completed before the one denoted by c_2 can start its execution (see Figure 2).

(4) Some parameters must be pre-computed for each task on each assigned resource.

(5) Once the task has been processed, any interruption is ignored.

An application is represented as a set of n clusters submitted by an end user $C = \{c_1, c_2, ..., c_n\}$, where each c_i contains a batch of tasks $T = \{T_{i1}, T_{i2}, ..., T_{ik}\}$. Each T_{ik} is defined by a pair (a_{ik}, l_{ik}), where a_{ik} denotes the arrival time of the user task T_{ik} and l_{ik} the length of the user task T_{ik}. Consider a set of m virtual machines $S = \{VM_1, VM_2, ..., VM_m\}$. Each virtual machine VM_j is associated with a tuple (s_j, u_j, r_j), where s_j is the processing speed of the j-th virtual machine, expressed in

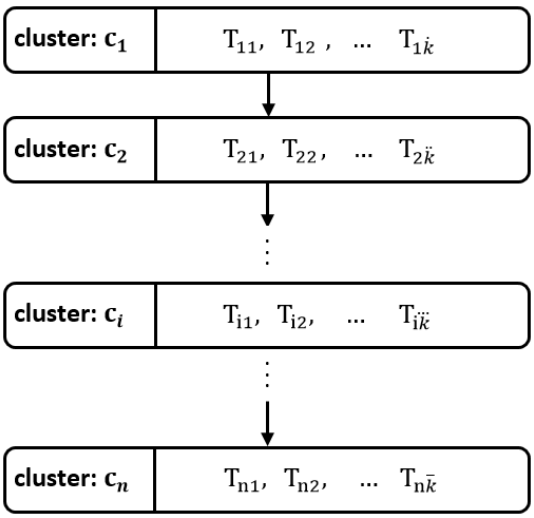


Figure 2: A simple application example.

Millions of Instructions Per Second (MIPS), u_j is the unit price of using the resource and r_j is the reliability rate of the resource that will be defined in this Section.

When designing an optimization algorithm for a particular problem, we seek to optimize certain criteria. These criteria depend on the nature of the problem to be addressed. In this work, the selection of VMs during the scheduling process is evaluated based on three end-user QoS-based criteria, namely makespan, execution cost, and reliability.

Makespan: it is the overall completion time required to complete the execution of all tasks. It is calculated as the difference between the time of submission of the first task and the time of receiving the results (the time when the last task is completed). It is defined as follows:

$$Makespan = max_{j \in TSK}\left(FinishTime_j\right) - min_{i \in TSK}\left(SubmissionTime_i\right) \quad (1)$$

where TSK is the set of tasks submitted to the scheduler for execution.

Cost: most cloud providers set prices for their services. In our case, the total runtime cost is the sum of the individual costs associated with the used VMs for all tasks. The usage cost for a given instance is calculated by the bill function, which takes two inputs: the unit price of the resource, and the execution time required to complete the task. It is formulated as follows:

$$Cost = \sum_{i=1}^{n} vm_i^{time} * vm_i^{Uprice} \quad (2)$$

where n is the number of tasks in the application, vm^{time} refers to the time a VM has executed a given task and vm^{Uprice} is the unit price of VM for executing a task.

Reliability: it represents the probability that a task will be performed successfully, without any resource failure. Our model for measuring reliability is inspired from the one provided in [41]. It is based on a failure rate λ which is an intrinsic property of the resource taking values ranging from 10^{-5} to 10^{-7} as in [22]. The reliability is calculated by the following formula.

$$Reliability = \exp^{-\sum_{i=1}^{n} TE(T_i)*\lambda_j} \quad (3)$$

where, $TE(T_i)$ is the execution time of the task T_i and λ_j is the failure rate of the machine executing the task.

4. The proposed approach

In this section, we first give the definition of the studied problem and then present our proposal. Let's remember that our main aim is to assign tasks to VMs in order to satisfy one or more QoS optimization objectives. For this purpose, two different optimization problems are considered. First, we deal with a single-objective where we aim at minimizing the time maskespan. To assess the performances of our algorithm, we compare it to three known task scheduling algorithms which are Round Robin [42], Max-Min [43] and Min-Min [44]. In the second case, we address the multiobjective scheduling optimization in cloud environments. To this end, we use a bio-inspired pollination behavior of flowers to design an efficient task scheduling algorithm. This approach was chosen due to the fact that it is one of the best performing meta-heuristic approaches, with a high convergence rate compared to other meta-heuristic approaches such as GA and PSO, which are widely recognized as the most significant references in the optimization field [45]. As reported above, we focus on three criteria: makespan, cost and reliability. The fitness function is calculated using Pareto-optimal front approach and TOPSIS method due to their ability to obtain ideal solutions for local optima.

A comparison was made against the aggregate weighted sum technique used in [10] which is, as far as we know, the closest work to the one presented in this paper. It consists of aggregating the different QoS criteria mentioned above into a single-objective as defined by equation (eq.4).

$$F = w_1 * Makespan + w_2 * Cost + w_3 * \left(\frac{1}{Reliability}\right) \quad (4)$$

where $w = \{w_1, w_2, w_3\}$ refers to the weight vector that reflects the importance or user's requirements with respect to each criterion.

In order to optimize simultaneously diverse objectives which are often conflicting and provide greater flexibility to the end-user based on his prescribed objective needs, a suitable compromise function that takes into account the considered criteria parameters is needed to achieve efficient trade-offs between

the targeted objectives in order to meets the user's QoS requirements. For this reason, aggregation solutions and multi-criteria decision methods such as TOPSIS are investigated with different weights assigned to each objective. In this study, we used two vectors, with different weights $\{0.5, 0.2, 0.3\}$ and $\{0.2, 0.5, 0.3\}$, corresponding respectively to the considered objectives {makespan, cost, reliability}. This means that we give more importance to the makespan objective in the first vector, and to the cost in the second weight vector.

The detailed presentation of FPA, Pareto optimilaty based approach and TOPSIS technique are given in this section.

4.1. Flower pollination algorithm

A bio-inspired flower pollination algorithm is originally proposed in 2012 by Xin-She Yang [45]. It is inspired by the pollination characteristics of flowering plants. The pollination process can take two forms, biotic and abiotic, depending on the mechanisms of pollen transfer. About 90% of flowering plants belong to biotic pollination, in which pollen is transferred through a specific pollinator such as insects, bats, birds or other animals [45]. The second form has a limited occurrence because it does not require any organism. Wind, gravity or diffusion in water contribute to transfer pollen and grass is an example for this type of pollination. In addition, pollination can be done by self or cross pollination. The former, which is also called local pollination, occurs when pollen from one flower pollinates the itself or other flowers of the same plant with the help of environmental factors [45]. Whereas the later, also known under the name of global pollination, occurs over long distances when pollen is delivered to a flower from another plant by direct or indirect intervention of pollinators following Levy's flight behavior [46].

From biological evolution point of view, the main objective of flower pollination is the optimal reproduction of plants through the survival of the fittest flowers in the flowering plants [45].

During the optimization process, the exploration of the search space of the FPA algorithm, which we study in this work, is done by biotic and cross pollination where the movement of the pollen is represented by the Markovian stochastic Lévy flight process. The latter is a random walk interspersed by long jumps from its current position according to a power law, based on a random step of the Lévy distribution to effectively mimic the characteristic of long distance movement of insects. Therefore, we can idealize the characteristics of the pollination scheme, flower constancy and pollinator behavior based on four main rules listed below:

- **Rule 1:** biotic and cross-pollination acting as a global pollination process via Levy flight.
- **Rule 2:** abiotic and self-pollination are considered as local pollination.
- **Rule 3:** consistency of flowers may be involved due to the similarity of two flowers.
- **Rule 4:** local pollination (exploitation) and global pollination (exploration) are controlled by a switching probability $p \in [0, 1]$.

Based on these four rules, the main steps of the standard FPA algorithm are parented in Algorithm 1.

Algorithm 1 FPA pseudo-code

```
Objective function min or max f(x), x = (x_1, x_2, ...)
Define a switch probability p ∈ [0, 1]
Generate initial population of flowers randomly
Find the best solution g_* in the initial population
while stop criterion do
    for each i = 1 : n (all n flowers in the population) do
        if rand() < p then
            Draw a step size L that obeys a Lévy distribution
            Global pollination via x_i^{t+1} = x_i^t + L(x_i^t − g_*)
        else
            Draw ε from a uniform distribution in [0, 1]
            Choose x_j^t and x_k^t randomly from all solutions
            Local pollination via x_i^{t+1} = x_i^t + ε(x_j^t − x_k^t)
        end if
        Evaluate the new solution
        if new solution is better then
            Update x_i^t with x_i^{t+1}
        end if
    end for
    Find the current best solution g_*
end while
```

The algorithm begins by randomly generating the initial population that will be evaluated to determine the best current solution. To calculte a new solution, the type of pollination must first be defined according to a recomputed probability (Rule 4), i.e. a random number is generated and if it is less than p, then the overall pollination and flower constancy (Rule 1 and Rule 3) can be applied as follows:

$$x_i^{t+1} = x_i^t + L\left(x_i^t - g_*\right) \tag{5}$$

where x_i^t is a solution i at iteration t, x_i^{t+1} is the solution vector generated at iteration $t + 1$, g_* is the current best solution found among all solutions in the generation. The parameter L is a step size based on Lévy flights. Since insects can move over a long distance with various distance steps, this is modeled by using a Lévy distribution [46] according to the following equation:

$$L \sim \frac{\lambda \Gamma(\lambda) \sin\left(\frac{\pi\lambda}{2}\right)}{\pi} \frac{1}{s^{1+\lambda}}, (s > 0) \tag{6}$$

In the above equation, $\Gamma(\lambda)$ is the standard gamma function, and this distribution is valid for large steps $s > 0$ which are

generated using the formula below:

$$s = \frac{U}{|V|^{1/\lambda}} \tag{7}$$

where U and V are two random samples from a normal Gaussian distribution with an average of zero and standard deviations of σ_u and σ_v:

$$U \sim \left(0, \sigma_u^2\right), V \sim \left(0, \sigma_v^2\right) \tag{8}$$

$$\sigma_u = \left[\frac{\Gamma(1+\lambda)}{\lambda\Gamma\left((1+\lambda)/2\right)} \frac{\sin\left(\frac{\pi\lambda}{2}\right)}{2^{(\lambda-1)/2}}\right]^{1/\lambda}, \sigma_v = 1 \tag{9}$$

Otherwise, if the generated random number is greater than p, local pollination and flower constancy (Rule 2 and Rule 3) are performed as follows:

$$x_i^{t+1} = x_i^t + \varepsilon\left(x_j^t - x_k^t\right) \tag{10}$$

where x_j^t and x_k^t are two different solutions chosen randomly and $\varepsilon \in [0, 1]$ is a random number to make this selection closer to a local random walk [45]. Then, a new solution will be evaluated based on its fitness (objective function). The new generation will be also evaluated to select the most promising one and the search process will be repeated until the stopping criterion is satisfied.

4.2. Pareto optimality based approach

Multi-objective optimization requires that the relative importance of each objective should be specified in advance, which needs prior knowledge of possible solutions. But, using the Pareto concept, it is possible to avoid the constraint to know in advance the possible solutions. Indeed, this is also one of the reasons for the popularity of these approaches based on the concept of Pareto [47].

In the following, we will define the concepts related to Pareto-based multi-objective optimization.

- **Dominance:** in a problem of minimization, a solution X dominates a solution Y if $\forall i = 1, 2, ..., k.\ f_i(X) \leq f_i(Y) \wedge \exists k \in \{1, 2, ..., k\}$ such as $f_k(X) < f_k(Y)$.

- **Pareto-optimal solutions:** a solution X is called Pareto-optimal if it is not dominated by any other feasible solutions. Pareto-optimal solutions, also called non-dominated solutions, are the ones that do not dominate each other. i.e, it is impossible to find a solution that improves the value of one objective without having a detrimental effect on other objectives. These solutions are defined as follows:

$$P^* = \{x \in X \ / \ \nexists\, \acute{x} \in X, \acute{x} < x\} \tag{11}$$

Since the dominance relationship of Pareto defines a partial order, the solution of a multi-objective optimization problem is a set of undivided points. This set is called the Pareto front as illustrated in Figure 3. It is defined as:

$$PF^* = F(x), x \in P^* \tag{12}$$

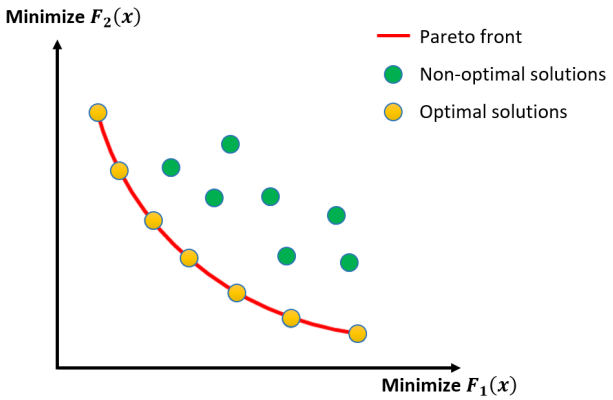


Figure 3: Example of dominated, non-dominated and pareto front solution set for a bi-criteria optimization problem.

When we deal with multi-criteria optimization, we seek to explore the complete Pareto front or an approximation of this set whose objective function values are close to the Pareto-optimal solutions. Often, the Pareto front contains an exponential number of solutions, where it is necessary to quantify the interest of a set of non-dominated solutions to assess the quality of the obtained result. Consequently, the use of a multi-criteria method such as TOPSIS is needed to achieve efficient trade-off solutions of the Pareto front. The TOPSIS method is a multi-criteria decision analysis technique that was originally developed by Hwang and Yoon [48] in 1981. It is used for ranking purposes and to obtain the best performance in multi-criteria decision making. Its main feature is to choose the action having the smallest distance to the ideal solution and the greatest one from the anti-ideal solution. See subsection 4.3 for more details of the employed method.

4.3. TOPSIS technique

TOPSIS approach is based on two potential alternatives i.e., Positive Ideal Solution (PIS) and Negative Ideal Solution (NIS). The former maximizes the benefit criteria and minimizes the cost criteria, whereas the later maximizes the cost criteria and minimizes the benefit criteria. The benefit criteria is for maximization, while the cost criteria is for minimization. The best alternative is the one that has the closest distance to PIS and the farthest from NIS. The different steps of the TOPSIS algorithm to find the best solution are outlined below.

Given a Decision Matrix (DM) with q alternatives and r criteria: makespan (MS), cost (C), reliability (R).

$$DM = \begin{pmatrix} MS_{T11} & C_{T12} & R_{T13} \\ MS_{T21} & C_{T22} & R_{T23} \\ \vdots & \vdots & \vdots \\ MS_{Tq1} & C_{Tq2} & R_{Tq3} \end{pmatrix} \quad (13)$$

- **Step 1:** normalize the decision matrix (NDM).

$$NDM_{ij} = \frac{X_{ij}}{\sqrt{\sum_{i=1}^{q} X_{ij}^2}} \quad for \quad j = 1, 2, \ldots, r \quad (14)$$

where X_{ij} denotes the i^{th} alternative value relative to the j^{th} criteria.

- **Step 2:** calculate the weighted normalized decision matrix.

$$\begin{aligned} MS_T &= w_1 * MS_T \\ C_T &= w_2 * C_T \\ R_T &= w_3 * R_T \end{aligned} \quad (15)$$

where, w_1, w_2, w_3 represent the prescribed weight values given to each criteria.

- **Step 3:** find the ideal and the anti-ideal solutions, PIS S^+ and NIS S^- are identified with the value of the criteria having respectively a positive and a negative impact on the solution.

$$\begin{aligned} S^+ &= \{V_1^+, V_2^+, \ldots, V_r^+\} \\ S^- &= \{V_1^-, V_2^-, \ldots, V_r^-\} \end{aligned} \quad (16)$$

where, S_j^+ and S_j^- respectively denotes the best and the worst value of criteria j for every ideal alternative.

- **Step 4:** compute for each alternative, the separation measures using Euclidean distance from PIS S^+ and NIS S^-.

$$\begin{aligned} D_i^+ &= \sqrt{\sum_{j=1}^{r} (X_{ij} - S_j^+)^2} \\ D_i^- &= \sqrt{\sum_{j=1}^{r} (X_{ij} - S_j^-)^2} \end{aligned} \quad (17)$$

- **Step 5:** evaluate the Relative Closeness (RC) to the ideal solution. More is important, more the alternative is closer to the positive ideal and farther from the negative ideal one.

$$RC_i = \frac{D_i^-}{D_i^+ + D_i^-} \quad for \quad i = 1, 2, \ldots, q, \quad 0 \le RC_i \le 1 \quad (18)$$

- **Step 6:** rank the q alternatives according to there RC_i. Here relative closeness is considered as the fitness function. The VM that gets the highest fitness value is considered as best solution for a given task.

4.4. FPA's adaptation

The FPA metaheuristic is adapted to our problem by considering a set of virtual machines as the FPA population and a virtual machine as a new solution candidate generated during the optimization process, in order to find the appropriate virtual machine for each submitted task. The pseudo code of the adapted FPA-based algorithm is outlined in Algorithm 2.

Algorithm 2 Adapted FPA-based task scheduling algorithm

```
1:  Initialize the algorithm parameters: population size, maximum number of iteration and switch probability p ∈ [0, 1]
2:  while T ≠ φ do
3:      Initialize the initial population with random VMs
4:      Find the best virtual machine V_best for task T_i in the initial population
5:      while stop criterion do
6:          for each virtual 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 ε from a uniform distribution in [0, 1]
12:                 Randomly choose two virtual machines V_j and
                    V_k from all solutions
13:                 Perform local pollination as V_i' = V_i + ε(V_j − V_k)
14:             end if
15:             Evaluate the 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 machine in the population)
21:     end while
22:     Update T
23: end while
```

The algorithm begins by identifying its control parameters in terms of switching probability, population size, and maximum number of iterations. In steps 3 and 4, a random set of virtual machines, referred to as an initial population, is generated. Following that, the fitness value is calculated to evaluate each VM's performance. The best population then identified and saved as V_{best}. At each iteration step in the inner while loop (lines 5 to 21), for every machine, a random number between 0 and 1 is generated and if the obtained value is greater than the switching probability parameter p, a global pollination using levy distribution is involved to compute a new solution; otherwise, local pollination using uniform distribution is applied. Then, the new generated solution is examined whether it is superior to the current solution or not, if yes, then the population is updated. Finally, the process is iterated until there are no remaining tasks to schedule. Note that the arrived tasks, will not be assigned to the VMs immediately. A pre-simulation phase of

the mapping process using the adapted FPA is necessary to find the suitable VM for each task before sending it to the broker.

To determine the best compromise solutions, we recall that, as stated in the introduction Section, the fitness function proceeds in two stapes: first, it uses the concept of Pareto optimality to find a non-dominated set of solutions and then extracts the most preferred one according to some criteria from the Preto-front of pairs (task, VM) by employing the TOPSIS technique.

5. Weighted sum method (WSM)

In order to compare our proposal to the mutli-criteria scheduling based WSM approach [10], which is, to our knowledge, the closest work to the one presented in this work, we briefly outline hereafter the key features of this method.

In decision theory, WSM technique is the best known multi-objective method due to its simplicity and speed. It allows to react efficiently, in real time, in order to evaluate a number of alternatives according to the considered decision criteria. It is applicable only when all the criteria are quantitative, i.e. they all have exactly the same unit. The main idea of this method is to merge the different objectives to be optimized into a single-objective function. We note that a weighting parameter can be introduced if needed to reflect the relative importance of an objective compared to others. In this way, once the problem formulation and the importance of each criteria are given, only one solution is generated as given by the following equation.

$$f(x) = \sum_{i=1}^{n} w_i f_i \tag{19}$$

where the weights $w_1, w_2, ..., w_n$ must meet the following two constraints:

$$\sum_{i=1}^{n} w_i = 1, \;\; 0 \leq w_i \leq 1 \tag{20}$$

Given the conflicting nature of the targeted objectives, it is often not always possible to optimize fairly these objectives. Thus, the introduction of the weighting parameters will allow the user to express his preferences by favoring some objectives. The basic idea is that a criterion which is in conflict with a favored one receives the minimum value among the possible weights (the available weights after operation: 1 - the weight of the relevant criterion; [49]). Therefore, the optimization process seeks for a compromise solution between all the objectives, while favoring the ones that match the user's preferences.

6. Results and discussion

The experimental results of the proposed algorithm are described in this section. They are performed using the CloudSim toolkit which is widely used by researchers for simulating and modeling cloud infrastructures. It is a toolkit for simulating distributed systems of any scale, that supports both system modeling and the behavior of cloud computing components such as data centers, virtual machines, cloudlets (tasks), and brokers [50].

Three parameters are defined for the FPA based approach: the population size, the control parameter λ, and the switching probability p that determine the percentage of diversification and intensification during the search. For all simulations, we used a population size equal to 20% of the created VMs. The switching probability and the control parameter are set to 0.8 and 1.5, respectively, as in [45].

As mentioned above, tasks are grouped in clusters where their dependency indicates the order of their execution. In our simulations, there are 15 precedence constraints to satisfy since we use 15 clusters. All the parameters of the performed simulations are shown in Table 2. Note that the experimental set up and the parameters used in our study, such as workload distribution, tasks submission time and cloud definition, are chosen in such a way that they are representative and are in line with those used in the literature [51, 52, 53].

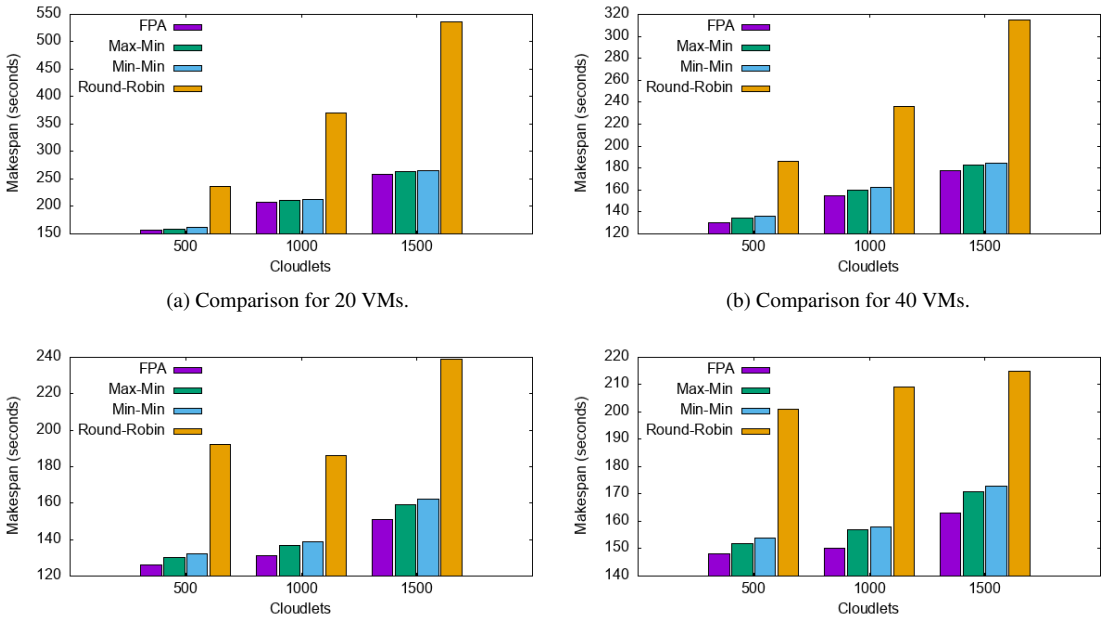
Entity type	Parameter	Value
Data-center	No. of data-centers	4
Host	No. of hosts	8
	PES	4 (Quad core)
	MIPS	6 000
	RAM	20 GB
	Storage	1 TB
	Bandwidth	10 GB
Virtual Machine	No. of VMs	20 - 40 - 60 - 80
	MIPS	1 000 to 5 000
	Cost	1.0 to 30.0
	Failure rate	10^{-5} to 10^{-7}
	RAM	1 GB to 5 GB
	Storage	10 GB
	Bandwidth	100 MB to 500 MB
	Policy type	Time Shared
Cloudlets	No. of cloudlets	500 - 1 000 - 1 500
	Length	3 000 to 10 000
	No. of clusters	15
	Type	Heterogeneous
	Submission time	Poisson distribution of parameter λ

Table 2: Cloudsim simulation parameters.

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.

Number of VMs	Cloudlets	Methode used			
		FPA	Max-Min	Min-Min	Round Robin
20	500	157	159	162	236
	1 000	208	211	213	370
	1 500	259	263	266	537
40	500	130	134	136	186
	1 000	155	160	162	236
	1 500	178	183	184	315
60	500	126	130	132	192
	1 000	131	137	139	186
	1 500	151	159	162	239
80	500	148	152	154	201
	1 000	150	157	158	209
	1 500	163	171	173	215

Table 3: Comparative results of proposed algorithm to other scheduling heuristics.



(a) Comparison for 20 VMs.

(b) Comparison for 40 VMs.

(c) Comparison for 60 VMs.

(d) Comparison for 80 VMs.

Figure 4: Comparison between FPA and heuristics.

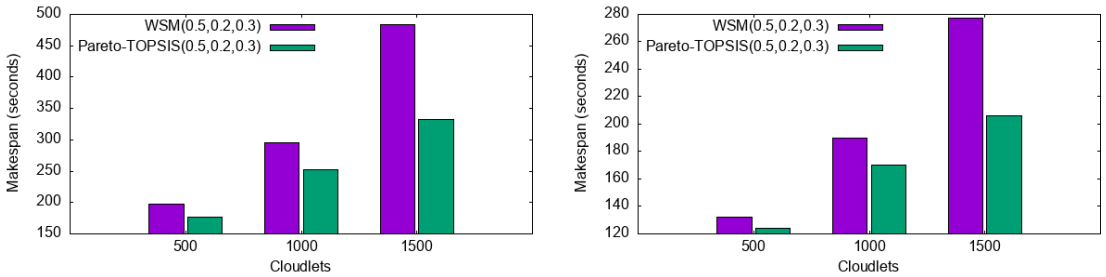
6.1. Single-objective evaluation

The comparison results between the proposed algorithm and the heuristics Max-Min, Min-Min and Round Robin are presented in Table 3. We recall that, in this group of experiments, we consider only one criterion which is the time makespan value. The number of VMs is varied from 20 to 80 with increments of 20 and three sets of simulations with the same number of tasks are carried out: 500, 1000 and 1500. The obtained results are shown in Figures 4(a)(b)(c)(d).

As expected, we can observe that the makespan objective increases as the number of tasks increases. This is due to the fact that the capacity of the resources to execute the tasks decreases gradually as the workload increases over time. It can also be seen that our proposal behaves better than all other heuristic techniques especially for large number of tasks. For instance, in the case of 60 VMs, the performance of adapted FPA is higher than Max-Min by 4 and 8 seconds for the sets of 500 and 1500 tasks respectively. This can be explained by the fact that the adapted FPA's policy explores efficiently the search space based on Lévy flights. This leads to a noticeable improvement of the global optimization objective compared to the performances of other heuristics, regardless of the number of the used VMs or the submitted tasks during the scheduling process.

Number of VMs	Cloudlets	Technique used		Reduction rate (%)
		WSM [10]	Pareto-TOPSIS	
20	500	198	177	10.6
	1 000	295	252	14.5
	1 500	484	333	31.1
40	500	132	124	6
	1 000	190	170	10.5
	1 500	277	206	25.6
60	500	146	139	4.7
	1 000	176	162	7.9
	1 500	214	178	16.8
80	500	124	119	4.0
	1 000	144	135	6.2
	1 500	163	151	7.3

Table 4: Makespan performance comparison with $\{0.5, 0.2, 0.3\}$ weight vector.



(a) Comparison for 20 VMs.

(b) Comparison for 40 VMs.

(c) Comparison for 60 VMs.

(d) Comparison for 80 VMs.

Figure 5: Comparison in terms of makespan.

6.2. Multi-objective evaluation

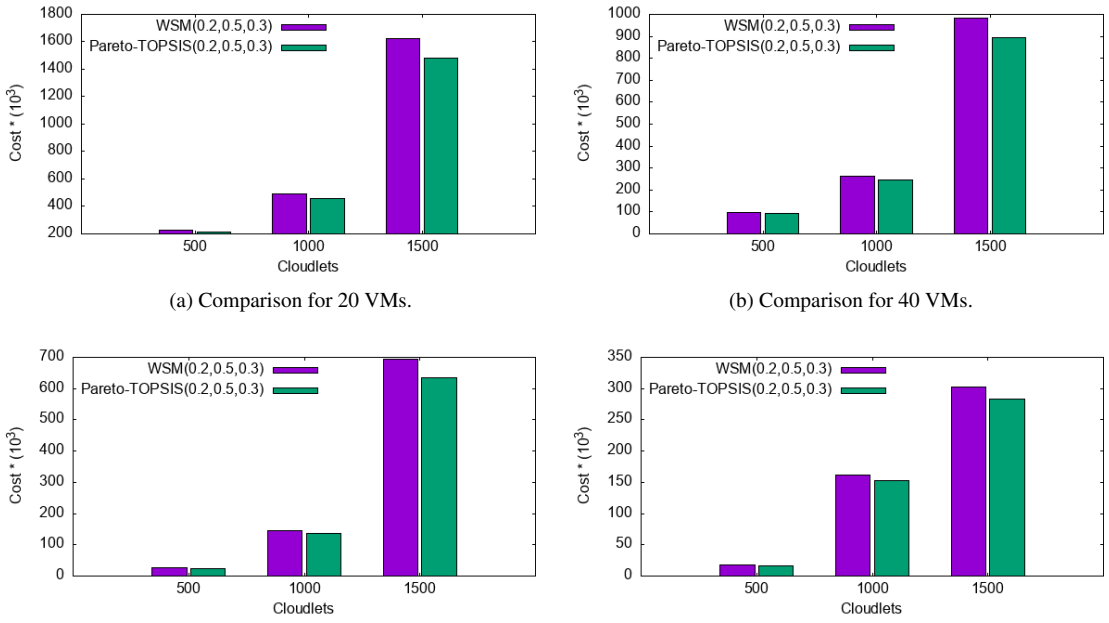
The second set of experimentations is devoted to the multi-objective scheduling case. A comparison was carried out using two variants of the compromise function to evaluate the quality of the FPA' solutions. As reported in the introduction section, the objective is to evaluate the efficiency of the concept of Pareto optimality with TOPSIS technique compared to the aggregate weighted sum [10] which merges the different metrics (see Section 3) of QoS.

First, we evaluate the time makespan of the compared approaches. As presented in Table 4 and Figures 5(a)(b)(c)(d), it is apparent that the Pareto-TOPSIS approach has higher performances with the weight vector {0.5, 0.2, 0.3} which favors the time makespan objective. For instance, compared to the weighted sum method, the reduction rate of our proposal in the case of 40 VMs is 6%, 10.5% and 25.6% for 500, 1000, and 1500 tasks, respectively.

This experiment shows that Pareto-TOPSIS strategy alleviates this problem by finding better compromise solutions between the considered objectives. This can be explained by the Pareto dominance concept which does not allow one objective to dominate an other. We can also observe from Table 4, that

Number of VMs	Cloudlets	Technique used		Reduction rate (%)
		WSM [10]	Pareto-TOPSIS	
20	500	225 694	213 626	5.3
	1 000	490 613	460 142	6.2
	1 500	1 626 716	1 478 813	9.0
40	500	97 390	92 552	4.9
	1 000	262 573	247 856	5.6
	1 500	984 540	895 000	9.0
60	500	25 297	24 542	2.9
	1 000	143 927	135 419	5.9
	1 500	694 443	634 251	8.6
80	500	17 141	16 776	2.1
	1 000	161 064	152 121	5.5
	1 500	302 622	282 601	6.6

Table 5: Cost performance comparison with $\{0.2, 0.5, 0.3\}$ weight vector.



(a) Comparison for 20 VMs.

(b) Comparison for 40 VMs.

(c) Comparison for 60 VMs.

(d) Comparison for 80 VMs.

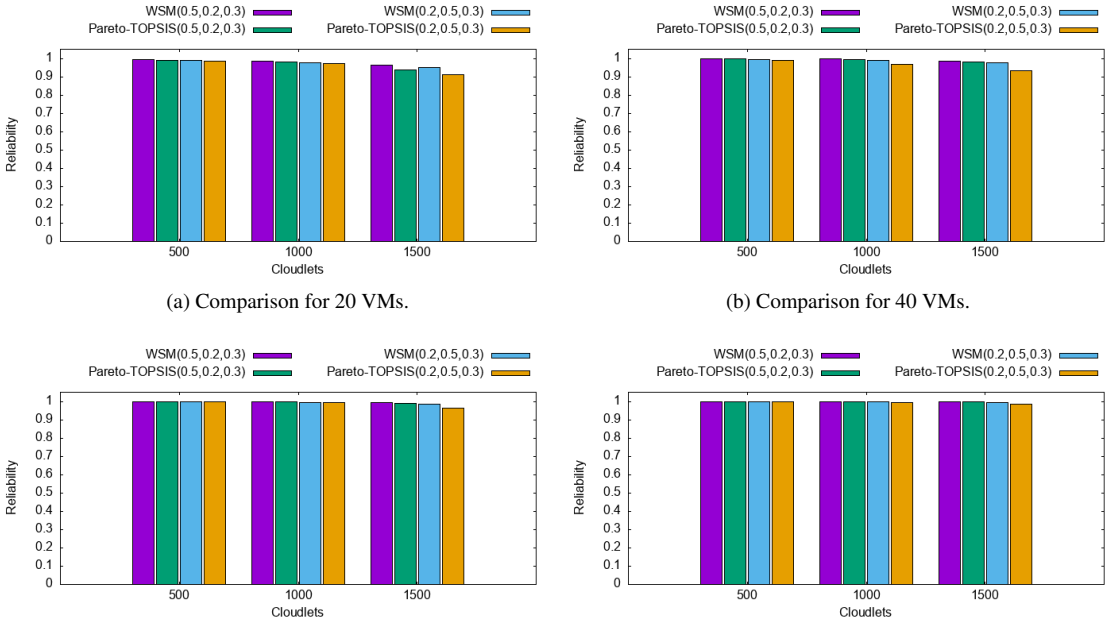
Figure 6: Comparison in terms of cost.

when we fix the number of taks and increase the number of VMs, the reduction rate gradually decreases. For instance, in the case of 500 tasks, the reduction rate is 10.6%, 6%, 4.7%, and 4.0% for 20 VMs, 40 VMs, 60 VMs, and 80 VMs, respectively. However, this is not particularly surprising in light of the fact that VMs have more ability to influence makespan time.

Comparing the results of the weighted sum with the ones of Pareto-TOPSIS in terms of execution cost, we can observe in Table 5 and from Figures 6(a)(b)(c)(d) respectively, that the latter obtains better performance values than the weighted sum in all the scenarios tested with a weighting vector that favors the execution cost. For example, in the case of 60 VMs the Pareto-TOPSIS based approach achieves solutions that are 8.6% more efficient in terms of execution cost than those provided by the weighted sum method. This improvement is due to the Pareto-TOPSIS based approach tries to ensure efficient trade-offs between the three considered objectives owing to their importance. This leads to effective cost minimization during the scheduling process. However, in terms of reliability optimization, we can notice from Table 6 and Figures 7(a)(b)(c)(d) respectively, that the difference between the achieved performances of the compared methods are almost similar regardless of the number of VMs and the number of tasks.

Number of VMs	Cloudlets	Technique used			
		{0.5, 0.2, 0.3}		{0.2, 0.5, 0.3}	
		WSM [10]	Pareto-TOPSIS	WSM [10]	Pareto-TOPSIS
20	500	0.993	0.99	0.988	0.986
	1 000	0.986	0.979	0.978	0.97
	1 500	0.964	0.937	0.952	0.909
40	500	0.998	0.998	0.994	0.988
	1 000	0.996	0.994	0.988	0.967
	1 500	0.987	0.981	0.977	0.933
60	500	0.999	0.999	0.999	0.998
	1 000	0.998	0.998	0.995	0.992
	1 500	0.994	0.991	0.987	0.962
80	500	0.999	0.999	0.999	0.999
	1 000	0.999	0.998	0.996	0.993
	1 500	0.998	0.997	0.994	0.987

Table 6: Reliability performance comparison.



(a) Comparison for 20 VMs.

(b) Comparison for 40 VMs.

(c) Comparison for 60 VMs.

(d) Comparison for 80 VMs.

Figure 7: Comparison in terms of reliability.

Taken as a whole, for both makespan and cost objectives, it is more interesting to use Pareto-TOPSIS based approach with the appropriate weight vector. Indeed, the weighted sum is a simple projection from the multi-objective optimization to the single-objective one. A purely multi-objective approach like Pareto-TOPSIS is more recommended in this context. Regarding reliability, this is a somewhat special measure because in the cloud, VMs are considered reliable since existing cloud providers offer services with too low failure rates that can go down to 10^{-7}. For this reason, the method used does not have much impact on this objective.

7. Conclusion

In this paper, we have addressed the problem of task scheduling in cloud computing environments. The aim is to orchestrate the assignment of tasks to VMs while satisfying users' requirements. To this end, a new scheduling algorithm is proposed to deal with both single and multiobjective optimization variants. It is made upon a combination of a bio-inspired flower pollination and Pareto optimilaty approach. Moreover, it makes use of the TOPSIS technique to ensure efficient trade-offs between the targeted objectives, namely time makespan, cost and

reliability. To the best of our knowledge, this is the first study to investigate the combination of the Pareto front principle as well as the TOPSIS technique with the pollination behavior of flowers to improve the performances of the scheduling process. The detailed experimental evaluation and the different studied scenarios using CloudSim framework corroborate the merits of our proposal.

Other questions remain to be addressed such as extending the studied optimization problem to workflows scheduling in real cloud environments. In this context, we expect a difficult challenging trade-off between communication costs minimization and executing tasks in parallel under different requirements of the end users. Another research direction is to consider other QoS such as fault tolerance and load balancing to cope with unpredictable failures and achieve global stable equilibrium. Furthermore, it would also be interesting to investigate other meta-heuristics such as GA [54], PSO [55] or CS [56] as well as their hybridization with FPA algorithm to analyse and compare the behavior of the task scheduling process on the achieved global performances.

References

[1] M. Masdari, F. Salehi, M. Jalali, M. Bidaki, A survey of pso-based scheduling algorithms in cloud computing, Journal of Network and Systems Management 25 (2017) 122–158.

[2] M. Kumar, S. C. Sharma, A. Goel, S. P. Singh, A comprehensive survey for scheduling techniques in cloud computing, Journal of Network and Computer Applications 143 (2019) 1–33.

[3] A. Pradhan, S. K. Bisoy, A. Das, A survey on pso based meta-heuristic scheduling mechanism in cloud computing environment, Journal of King Saud University-Computer and Information Sciences . (2021) 1–14.

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

[5] A. Benoit, M. Hakem, Y. Robert, Multi-criteria scheduling of precedence task graphs on heterogeneous platforms, Comput. J. 53 (2010) 772–785.

[6] E. Jeannot, E. Saule, D. Trystram, Optimizing performance and reliability on heterogeneous parallel systems: Approximation algorithms and heuristics, J. Parallel Distributed Comput. 72 (2012) 268–280.

[7] I. Assayad, A. Girault, H. Kalla, Tradeoff exploration between reliability, power consumption, and execution time for embedded systems - the TSH tricriteria scheduling heuristic, Int. J. Softw. Tools Technol. Transf. 15 (2013) 229–245.

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

[9] A. Semmoud, M. Hakem, B. Benmammar, J.-C. Charr, Load balancing in cloud computing environments based on adaptive starvation threshold, Concurrency and Computation: Practice and Experience 32 (2020) e5652.

[10] I. Gupta, A. Kaswan, 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.

[11] T. Bezdan, M. Zivkovic, E. Tuba, I. Strumberger, N. Bacanin, M. Tuba, Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm, in: International Conference on Intelligent and Fuzzy Systems, Springer, 2020, pp. 718–725.

[12] M. Sardaraz, M. Tahir, A hybrid algorithm for scheduling scientific workflows in cloud computing, IEEE Access 7 (2019) 186137–186146.

[13] I. Casas, J. Taheri, R. Ranjan, A. Y. Zomaya, Pso-ds: a scheduling engine for scientific workflow managers, The Journal of Supercomputing 73 (2017) 3924–3947.

[14] A. M. Manasrah, H. Ba Ali, Workflow scheduling using hybrid ga-pso algorithm in cloud computing, Wireless Communications and Mobile Computing 2018 (2018).

[15] M. Adhikari, T. Amgoth, S. N. Srirama, Multi-objective scheduling strategy for scientific workflows in cloud environment: A firefly-based approach, Applied Soft Computing 93 (2020) 106411.

[16] M. Imdoukh, I. Ahmad, M. Alfailakawi, Optimizing scheduling decisions of container management tool using many-objective genetic algorithm, Concurrency and Computation: Practice and Experience 32 (2020) e5536.

[17] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints, IEEE transactions on evolutionary computation 18 (2013) 577–601.

[18] K. K. Chakravarthi, L. Shyamala, Topsis inspired budget and deadline aware multi-workflow scheduling for cloud computing, Journal of Systems Architecture 114 (2021) 101916.

[19] R. Medara, R. S. Singh, Energy efficient and reliability aware workflow task scheduling in cloud environment, Wireless Personal Communications (2021) 1–20.

[20] H. Topcuoglu, S. Hariri, M.-Y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE transactions on parallel and distributed systems 13 (2002) 260–274.

[21] Q. Huang, S. Su, J. Li, P. Xu, K. Shuang, 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.

[22] R. Garg, M. Mittal, et al., Reliability and energy efficient workflow scheduling in cloud environment, Cluster Computing 22 (2019) 1283–1297.

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

[24] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, S. Hu, Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based heft, Future Generation Computer Systems 93 (2019) 278–289.

[25] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, IEEE transactions on evolutionary computation 6 (2002) 182–197.

[26] A. K. A. Talukder, M. Kirley, R. Buyya, Multiobjective differential evolution for scheduling workflow applications on global grids, Concurrency and Computation: Practice and Experience 21 (2009) 1742–1756.

[27] S. R. Thennarasu, M. Selvam, K. Srihari, A new whale optimizer for workflow scheduling in cloud computing environment, Journal of Ambient Intelligence and Humanized Computing 12 (2021) 3807–3814.

[28] L. Abualigah, A. Diabat, A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments, Cluster Computing 24 (2021) 205–223.

[29] Y. Yang, B. Yang, S. Wang, T. Jin, S. Li, An enhanced multi-objective grey wolf optimizer for service composition in cloud manufacturing, Applied Soft Computing 87 (2020) 106003.

[30] S. Seth, N. Singh, Dynamic heterogeneous shortest job first (dhsjf): a task scheduling approach for heterogeneous cloud computing systems, International Journal of Information Technology 11 (2019) 653–657.

[31] T. Menouer, P. Darmon, New scheduling strategy based on multi-criteria decision algorithm, in: 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), IEEE, 2019, pp. 101–107.

[32] T. P. Jacob, K. Pradeep, A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization, Wireless Personal Communications 109 (2019) 315–331.

[33] J. K. Samriya, N. Kumar, An optimal sla based task scheduling aid of hybrid fuzzy topsis-pso algorithm in cloud environment, Materials Today: Proceedings . (2020) .

[34] M. N. Kabir, J. Ali, A. A. Alsewari, K. Z. Zamli, An adaptive flower pollination algorithm for software test suite minimization, in: 2017 3rd international conference on electrical information and communication technology (EICT), IEEE, 2017, pp. 1–5.

[35] S. Ulusoy, S. M. Nigdeli, G. Bekdaş, Novel metaheuristic-based tuning of pid controllers for seismic structures and verification of robustness, Journal of Building Engineering 33 (2021) 101647.

[36] H. R. Patel, V. A. Shah, Application of metaheuristic algorithms in interval type-2 fractional order fuzzy tid controller for nonlinear level control process under actuator and system component faults, International Journal of Intelligent Computing and Cybernetics 14 (2021) 33–53.

[37] Z. A. A. Alyasseri, A. T. Khader, M. A. Al-Betar, A. K. Abasi, S. N. Makhadmeh, Eeg signals denoising using optimal wavelet transform hybridized with efficient metaheuristic methods, IEEE Access 8 (2019) 10584–10605.

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

[39] N. K. Walia, N. Kaur, M. Alowaidi, K. S. Bhatia, S. Mishra, N. K. Sharma, S. K. Sharma, H. Kaur, An energy-efficient hybrid scheduling algorithm for task scheduling in the cloud computing environments, IEEE Access (2021) 117325–117337.

[40] R. Vijayalakshmi, S. Prathibha, A novel approach for task scheduling in cloud, in: 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), IEEE, 2013, pp. 1–5.

[41] X. Wang, C. S. Yeo, R. Buyya, J. Su, Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm, Future Generation Computer Systems 27 (2011) 1124–1134.

[42] H. S. Mahalle, P. R. Kaveri, V. Chavan, Load balancing on cloud data centres, International Journal of advanced research in computer science and software engineering 3 (2013).

[43] K. Etminani, M. Naghibzadeh, A min-min max-min selective algorihtm for grid task scheduling, in: 2007 3rd IEEE/IFIP International Conference in Central Asia on Internet, IEEE, 2007, pp. 1–7.

[44] X. He, X. Sun, G. Von Laszewski, Qos guided min-min heuristic for grid task scheduling, Journal of computer science and technology 18 (2003) 442–451.

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

[46] I. Pavlyukevich, Lévy flights, non-local search and simulated annealing, Journal of Computational Physics 226 (2007) 1830–1844.

[47] B. Benmammar, Y. Benmouna, 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 59 (2019) 152–164.

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

[49] T. R. Newman, B. A. Barker, A. M. Wyglinski, A. Agah, J. B. Evans, G. J. Minden, Cognitive engine implementation for wireless multicarrier transceivers, Wireless communications and mobile computing 7 (2007) 1129–1142.

[50] Cloudsim: A framework for modeling and simulation of cloud computing infrastructures and services, . URL: `http://www.cloudbus.org/cloudsim`.

[51] H. S. Narman, M. S. Hossain, M. Atiquzzaman, h-ddss: Heterogeneous dynamic dedicated servers scheduling in cloud computing, in: 2014 IEEE International Conference on Communications (ICC), IEEE, 2014, pp. 3475–3480.

[52] G. D. C. Rodrigues, R. Calheiros, V. Guimaraes, Gl d. santos, mb de carvalho, lz granville, lmr tarouco, and r. buyya, monitoring of cloud computing environments: Concepts, solutions, trends, and future directions, in: Proceedings of the 31st Annual ACM Symposium on Applied Computing, volume 10, 2016, pp. 2851613–2851619.

[53] R. Y. Shtykh, Y. Zhu, Q. Jin, A context-aware framework for flowable services, in: 2009 Third International Conference on Multimedia and Ubiquitous Engineering, IEEE, 2009, pp. 251–256.

[54] D. E. Goldberg, Genetic algorithms in search, optimization, and machine learning, Addison wesley 1989 (1989) 36.

[55] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-international conference on neural networks, volume 4, IEEE, 1995, pp. 1942–1948.

[56] X.-S. Yang, S. Deb, Cuckoo search: state-of-the-art and opportunities, in: 2017 IEEE 4th international conference on soft computing & machine intelligence (ISCMI), IEEE, 2017, pp. 55–59.