A New Hybrid Multi-objective Optimization Algorithm for Task Scheduling in Cloud Systems

Arslan Nedhir Malti[1*], Mourad Hakem[2] and Badr Benmammar[1]

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

*Corresponding author(s). E-mail(s):
arslannedhir.malti@univ-tlemcen.dz;

Abstract

Nowadays, cloud computing is widely used in various fields and is booming day by day with different services offered to users according to their needs and contracts. However, this has brought many challenges and constraints that organizations must to be aware of and address to fully harness its power. In practice, the most important issue that has gained significant influence in improving system performances is task scheduling. Unfortunately, it is known that this problem is NP-hard and the use of both heuristics and metaheuristics is required to obtain near optimal solutions but in a reasonable amount of computation time. Despite the fact that several studies have been published in the literature, there are still interesting and relevant questions to be addressed. For instance, when it comes to the stagnation phenomenon of local solutions and the premature convergence of the search process, it is crucial to execute the exploration and exploitation stages carefully as improperly performed stages may result in inefficient task mapping solutions. Consequently, to overcome the limitations of existing techniques in terms of local optimality trap and immature convergence, a novel hybrid optimization algorithm is proposed to deal with multi-objective task scheduling in heterogeneous IaaS cloud environments. It is based on the combination of the pollination behavior of flowers with the search exploration capability of the grey wolf optimizer strategy. In addition, it makes use of the evolutionary algorithms crossover operators to strike a good balance between exploring new solutions and exploiting the already discovered ones. Based on the CloudSim framework, different test-bed scenarios and both synthetic and standard workload traces were considered to assess the performance of the proposed algorithm by evaluating its objective function in terms of four optimization criteria, namely time

makespan, resource utilization, degree of imbalance and throughput. Our proposal was compared to the well-known optimization-based scheduling techniques in the literature, like TSMGWO, GGWO, LPGWO and FPA approach. The obtained results corroborate the merits of the new designed hybrid algorithm.

Keywords: Cloud computing, Task scheduling, Multi-objective optimization, Flower pollination algorithm, Grey wolf optimizer, Metaheuristics

1 Introduction

Cloud computing is a promising paradigm that has gained popularity for business and scientific purposes. It is a model for delivering Information Technology (IT) services in which resources are retrieved from the internet through web-based tools and applications. Thess services provide on-demand access to shared pools of configurable system resources, such as networks, servers, storage, and applications. This allows organizations to consume and pay for IT services as needed rather than having to set up and maintain their own costly infrastructure.

To effectively harness the potential of the cloud system, Cloud Service Providers (CSPs) such as Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform, Cisco, IBM and Accenture are striving to meet the diverse requirements of their clients by offering various services that can be broadly classified into three categories, namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [1]. At the bottom of the cloud computing stack is the IaaS model which involves the delivery of raw computing resources, such as virtualized servers, storage, and networking. It is the most basic level of cloud computing and allows customers to rent computing resources based on their needs. Some of the most popular IaaS cloud service providers include AWS, Microsoft Azure, and Google Cloud Platform. PaaS builds on top of IaaS by providing a platform for customers to develop, run, and manage their own applications. It includes everything from the operating system to the middleware, databases, and programming languages. Examples of PaaS providers are AWS Elastic Beanstalk, Microsoft Azure App Service, and Google App Engine. The third type of cloud service model, SaaS, is the most advanced level of cloud computing, in which customers can access fully-featured software applications. These applications are typically used through a web browser and are maintained and managed by the service provider. Salesforce, Microsoft Office 365, and Google Workspace are some examples of SaaS providers.

Although Cloud computing technology offers users higher computation advantages in terms of cost, flexibility, and availability, it rises some challenges that organizations must be aware of and address to fully leverage its power. One of these main challenges is task scheduling [2] which plays an important key role in ensuring efficient use of the cloud's computing resources. It involves orchestrating appropriately the request submitted by the users to the set of resources under specific constraints in such a way that the overall performance of the system is optimized.

Basically, the cloud system is made up of two main entities: consumers and cloud service providers. Both must adhere to their commitments as outlined in the Service Level Agreement (SLA) to avoid penalties. Cloud service providers must keep their commitments in terms of computing power, service reliability, security, and resource availability to satisfy customers while striving to maximize their profits. On the other hand, consumers seek to accomplish their tasks in the shortest amount of time. However, service quality constraints can be divided into two main categories: provider needs and consumer desires. Therefore, scheduling strategies should not only aim to satisfy user-defined Quality of Service (QoS) requirements through SLAs, but also ensure that provider profits are not significantly impacted [3].

In recent years, several researchers have attempted to tackle the task scheduling problem by using various optimization-based approaches which include exact and heuristics methods depending on the size and complexity of the targeted problem [4]. Metaheuristic optimization approaches are another strategies that are widely used to address diverse and real-world complex optimization issues, including task scheduling in parallel and distributed systems. Some popular metaheuristic techniques include Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Bat Algorithm (BA), Grey Wolf Optimizer (GWO), Flower Pollination Algorithm (FPA), Dragonfly Algorithm (DA), Salp Swarm Algorithm (SSA), Harmony Search (HS) and Shuffled Frog Leaping Algorithm (SFLA) [5–9]. These methods differ in their approach to exploring (diversification) or exploiting (intensification) the research space. Each has a unique combination of diversification and intensification strategies in finding the global solution. However, this class of optimization techniques has some weak points and all these algorithms potentially suffer from the premature convergence phenomenon and the difficulty to overcome the trap of local optimality when faced with a large solutions space which may affects the quality of service and the performance of the cloud system. Consequently, it is important to seek for efficient and adaptive task scheduling strategies that can achieve the right balance between exploration and exploitation while fulfilling the requirements of the cloud system.

Despite an abundant literature works and the results achieved in the field of multi-objective task scheduling in cloud computing, a number of concerns remain still not investigated [5]. For instance, combining various heuristics or metaheuristics to exploit their strengths and address their limitations is one promising research direction. To this purpose, a new hybrid optimization algorithm is introduced, in this paper, to tackle the problem of multi-objective task scheduling problem in cloud systems, considering both service provider and consumer requirements. To the best of our knowledge, no previous studies in the literature have examined the improvements proposed in this work by hybridizing the key features of the considered metaheuristics to deal with multi-criteria scheduling in cloud computing environments. The aim is to leverage the strengths of various approaches to improve the overall performances of the task scheduling process in terms of the achieved trade-offs between the provider's benefits and the user's QoS requirements using four main metrics: time makespan, resource utilization, degree of imbalance and throughput.

The proposed algorithm is made upon a hybridization of the bio-inspired *FPA algorithm* with the swarm intelligence-*grey wolf optimizer* and uses the GA crossover

operators to provide efficient task mapping in order to satisfy the needs of the service provider and the end-user, respectively. The main contributions of this paper can be summarized as follows:

1. We formulate the task scheduling in clouds as a multi-objective optimization problem that takes into account not only the measures needed to meet the user-defined QoS requirements through SLAs, but also considers the measures that ensure the profitability of the service provider.
2. We design a new hybrid strategy that combines the pollination behavior of flowers and the grey wolf optimizer to achieve better global performances of the task scheduling process in heterogeneous cloud computing environments.
3. We use the evolutionary algorithms crossover operators in order to achieve a suitable balance between the stages of exploration and exploitation throughout the search process.
4. Both synthetic and standard workload traces are investigated to assess and analyse the performance of our approach by evaluating its objective function in terms of four optimization criteria, namely time makespan, resource utilization, degree of imbalance and throughput.
5. Based on the CloudSim framework, a series of test-bed scenarios and QoS metrics were considered and the obtained results show that the new hybrid strategy outperforms its direct competitors, such as GGWO [10], LPGWO [11], TSMGWO [12] and the standard FPA approach [13].

The rest of this paper is organized as follows: Section 2 presents an overview of the relevant techniques proposed in literature on task scheduling in cloud computing environments. Section 3 provides a comprehensive formal model outlining the scheduling problem being addressed. Section 4 examines the standard FPA and the GWO algorithms which form the basis of our research. In Section 5, we detail our proposed approach, which is made upon the pollination behavior of the flowers and grew wolf optimizer as well as the use of the crossover operators of evolutionary algorithms. We report the experimental details and a comparative performance analysis of our proposal in Section 6. Finally, conclusion and outlines for potential future research directions are given in Section 7.

2 Related works

Many approaches have been proposed in the literature to solve the problem of task scheduling in parallel and distributed systems, based on heuristics, metaheuristics and hybrid scheduling strategies. In this section, we do not provide an exhaustive state of the art, but rather focus mainly on recently proposed algorithms that are widely adopted in cloud computing.

Pirozmand et al. [14] introduced a new hybrid algorithm, known as GAECS, which combines genetic algorithm and Energy-Conscious Scheduling (ECS) model to efficiently schedule multi-objective tasks in cloud computing systems with a focus on time and energy efficiency. The purpose of this algorithm is to balance the trade-off between

time and energy consumption when assigning tasks to processors. The performance of GAECS algorithm was analyzed using MATLAB and found to be more accurate than other existing algorithms. An independent task scheduling algorithm, known as BA-ABC, was developed by Bezdan et al. [15]. It combines two metaheuristic methods, a standard BA search with the Artificial Bee Colony (ABC) algorithm's exploitation technique during search iterations. Moreover, the Quasi-Reflection-Based Learning (QRBL) methode is utilized to enhance both the convergence rate and the variety of solutions. The effectiveness of BA-ABC was conducted on the HPC2N and NASA Ames iPSC/860 standard parallel workload traces. The experimental performances were compared to four other optimization algorithms and the obtained results showed that the authors' proposal has a relative advantage compared to other algorithms.

A bi-objective optimization for independent task scheduling on cloud virtual resources was presented by Gupta et al [13]. It is based on the concept of pollination in nature and was compared with three other metaheuristic methods: PSO, GA, and Gravitational Search Algorithm (GSA). The designed approach involves mapping tasks to the available virtual machines through an effective pollen representation scheme and a dedicated process for retrieving the task-VM matching from a given pollen. The authors aim to optimize both the time makespan and the average cloud resource utilization. According to the simulation results, the FPA-based task scheduling approach outperforms other concurrent metaheuristic methods. However, it should be noted that the suggested work lacks dynamic flexibility as it only addresses static independent task scheduling and virtual machines. In [16], Bezdan et al. proposed an improved version of the flower pollination algorithm for independent task scheduling in cloud computing systems, named the Exploration-Enhanced FPA (EEFPA). The goal of the EEFPA is to maintain quality of service by focusing solely on the time makespan objective. The authors' study showed that the standard FPA approach struggles to effectively search the solution space in the early stages. To address this, the authors suggest replacing the worst individuals in the population with new random solutions during the first 30% of iterations. The reported results indicate that this technique decreases the makespan and enhances the convergence speed compared to FPA, Performance Budget ACO (PBACO), ACO, Min-min and First Come First Serve (FCFS) techniques.

In a recent paper, Miglani et al. [17] proposed an elastic and persuasive task scheduler based on a modified flower pollination algorithm in a cloud environment. The proposed approach, called multi-objective reliability-based workflow scheduler, aims to map tasks to the most suitable machines in terms of makespan, efficiency and incurred cost. Simulation results showed that this approach improves efficiency and performance compared to other existing methods, such as FPA, GWO and GA. Walia et al. [18] proposed an energy-efficient scheduling algorithm (HS), which combines both FPA and GA algorithms to distribute the computing resources among tasks with less energy consumption. The authors' study evaluates different performance metrics such as resource utilization, completion time, energy consumption, and computational cost in homogeneous and heterogeneous cloud platforms. The obtained results using ASP.NET reveal that HS outperforms existing algorithms like GA and FPA in terms

of efficient task scheduling and resource management. However, the HS fairness index value fell below the prescribed threshold, indicating an unfair allocaion of resources.

In the work of Gokuldhev et al. [11], a Local Pollination-based Gray Wolf Optimizer (LPGWO) with an energetic concept was designed by combining two optimizer algorithms, GWO and the concept of pollination in nature. The authors' goal is to effectively balance makespan and resource energy consumption. The LPGWO consists of three phases: initialization phase, GWO phase and the FPA phase. The initialization phase employs chaotic mapping and the Opposition-Based Learning (OBL) method to provide an appropriate initial task scheduling solution. The GWO phase was used to improve convergence speed, while the FPA phase distributes data to the next set of candidate solutions through local pollination. The proposed LPGWO was evaluated on physical machines with low and high heterogeneity, and the results showed that it outperformed related existing approaches such as GA, PSO, Bacteria Foraging Algorithm (BFA), and Multi-Hybrid Bacteria Foraging Algorithm (MHBFA). In another study with similar lines, a new multi-objective task scheduling algorithm, called Local Pollination based Moth Search Algorithm (LPMSA), has been developed to address heterogeneity and dynamicity in cloud systems [19]. LPMSA combines the local search of the FPA approach with the Moth Search Algorithm (MSA) to enhance its exploitability and avoid local optima. The effectiveness of LPMSA was evaluated on low and high heterogeneous machines with uniform and non-uniform parameters. The results indicate a significant improvement in makespan, energy consumption, and convergence speed of the optimization process compared to other previous approaches.

In [12], a Multi-Objective Task Scheduling based on Grey Wolf Optimizer is presented. The proposed technique, which is called TSMGWO, aims to find an optimal or near-optimal solution to the task scheduling problem in IaaS cloud systems by considering multiple conflicting objectives such as makespan, resource utilization, degree of imbalance, and throughput simultaneously. The algorithm's performances were evaluated using three different benchmark datasets, namely GoCJ, Synthetic and HSCP data set, and the numerical results show that this method surpasses earlier heuristics FCFS and Modified Throttle (MT) methods as well as metaheuristics PSO, GA and WOA significantly. Natesan and Chokkalingam [20] developed a more accurate and efficient version of the standard grey wolf optimization algorithm called Mean-GWO. The comparison results using the CloudSim toolkit on two datasets left-skewed and right-skewed showed that the improved circling and hunting equations had a potential to perform better than competing techniques such as PSO and the standard GWO in terms of makespan and energy consumption.

A Pareto-based multi-objective GWO (PGWO) algorithm was proposed by Khalili et al. [21] for scheduling workflows in cloud systems. The goal is to balance conflicting objectives of minimizing both makespan and cost, and maximizing resource throughput for the provider. The algorithm's performance was evaluated using two workflow patterns: imbalanced (Montage) and balanced (Epigenomics). Simulation results demonstrate that the extended multi-objective algorithm produced a better trade-off between the considered objectives with a maximum spread of solutions and greater coverage ratio than Strength Pareto Evolutionary Algorithm 2 (SPEA2). Gobalakrishnan and Arun [10] have developed a hybrid multi-objective approach, known

as Genetic Gray Wolves Optimization (GGWO). Their goal is to improve the workflow scheduling performances in terms of total time, migration cost, load utilization and energy consumption as well as balancing the load among the computing resources. The authors use GA within GWO to enhance performance and to speed up the optimization process. The GGWO analysis was conducted using five common scientific workflows, namely LIGO, Montage, Epigenomics, SIPHT and Cybershake. Experiments revealed that GGWO can improve the performances of the task scheduling process compared to standard GWO and GA algorithm.

Khurana and Singh [22] proposed a hybrid bi-objective scheduling approach for scientific workflows that combines bio-inspired and swarm intelligence algorithms. The algorithm uses GWO and FPA with the PEFT algorithm for global and local optimization. The aim is to minimize both monetary cost and task execution time in the cloud by assigning the submitted tasks to the available virtual machines. The effectiveness of the proposed technique compared to flower pollination and genetic algorithm was demonstrated through numerical simulations. In another study, Amer et al. [23] have introduced a modified Harris hawks optimizer (HHO), called Elite Learning Harris Hawks Optimizer (ELHHO) to address the multi-objective scheduling problem. The purpose of this strategy is to explore more positions in the search space and improve the exploration process of the HHO algorithm. The proposed ELHHO technique combines two scientific intelligent methods: Elite Opposition-Based Learning (EOBL) and Minimum Completion Time (MCT). The EOBL strategy is used in the exploration process to improve the capacity of the global search in the HHO to balance between exploration and exploitation, while MCT generates the initial scheduled list to serve as the initial population. The simulations perfomed with the CloudSim toolkit made clear that the authors' work efficiently assigns the submitted tasks to the available VMs with a high balance degree, throughput, and resource utilization while reducing schedule length and resource execution cost. Table 1 illustrates the summary of the latest published state-of-the-art scheduling strategies, which are designed with various optimization techniques.

Overall, the previously studied bio-inspired and swarm optimization methods have great potential for task scheduling in cloud computing systems. Nevertheless, further improvements are necessary to achieve the right balance between exploration and exploitation through the use of suitable operators. Consequently, to overcome the limitations and drawbacks of existing optimization-based scheduling techniques, a new hybrid optimization algorithm for multi-criteria task scheduling in cloud systems is presented in this paper. Moreover, to maintain consistency and coherence with previous studies, the experimental set up and the parameters used are in line with those used in [12]. Specifically, the same QoS metrics and data set are used for this purpose.

3 Problem statement

The IaaS cloud is a commonly used model for managing resources, making scheduling in these systems an important topic of investigation for the research community. This model provides virtualized computing resources that are accessible to customers over the internet. Indeed, virtualization is one of the primary enablers for cloud computing.

Table 1 An overview of the related works

Contributions	Scheduling Type	Parameters	Simulation tool	Year	Ref
A new meta-heuristic method based on the GA and the ECS model	Task scheduling	Makespan Energy	MATLAB	2021	[14]
A method based on BA and diversification of the ABC algorithm	Task scheduling	Makespan Execution cost	CloudSim	2020	[15]
An efficient pollen representation scheme to determine the task-VM mapping	Task scheduling	Makespan Resource utilization	MATLAB	2017	[13]
Enhanced FPA for task scheduling in cloud	Task scheduling	Makespan	CloudSim	2021	[16]
Reliability-based multi-objective workflow scheduler upon modified flower pollination algorithm in cloud	Workflow scheduling	Makespan Efficiency Cost	N/A	2022	[17]
An energy-efficient hybrid algorithm that relies on both FPA and GA algorithms	Task scheduling	Completion time Resource utilization Energy and Cost	ASP.NET	2021	[18]
An energetic method of local pollination based on GWO algorithm	Task scheduling	Makespan Energy	MATLAB	2020	[11]
Local pollination-based moth search algorithm for task scheduling heterogeneous cloud	Task scheduling	Makespan Energy	CloudSim	2022	[19]
A metaheuristic GWO based approach for finding an optimal or near optimal solution to the task scheduling problem	Task scheduling	Makespan Resource utilization Degree of imbalance Throughput	CloudSim	2021	[12]
An improved version of the gray wolf optimization algorithm	Task scheduling	Makespan Energy	CloudSim	2019	[20]
A pareto-based multi-objective GWO algorithm	Workflow scheduling	Makespan Cost Throughput	CloudSim	2017	[21]
Hybridization of grey wolf optimizer and genetic algorithm	Task scheduling	Processing time Load utilization Migration cost Energy	CloudSim	2018	[10]
Hybridization of the GWO and FPA along with the PEFT algorithm	Workflow scheduling	Makespan Cost	CloudSim	2021	[22]
An enhanced version of HHO using the EOBL strategy and the MCT heuristic algorithm	Task scheduling	Makespan Cost Throughput Resource utilization Balance degree	CloudSim	2022	[23]

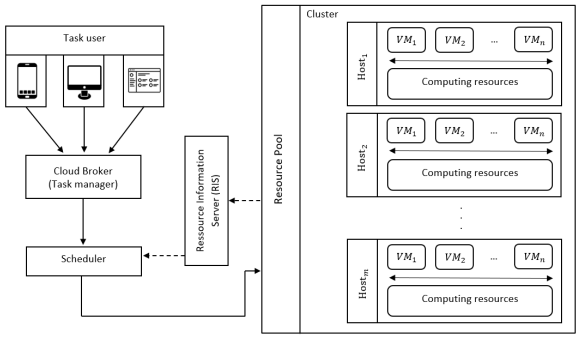

Fig. 1 Task scheduler framework in cloud

It allows cloud providers to represent physical resources as Virtual Machines (VMs). A typical cloud computing framework is illustrated in Figure 1. In cloud scheduling theory, as shown in Figure 1, when user requests are submitted to the cloud, they are forwarded to a cloud broker called also a task manager. The function of the cloud broker is to manage all the enlisted tasks and present them to the scheduler. This scheduler plays an essential role of assigning the available set of VMs to the user's submitted tasks based on the employed policy. To optimize the assignment process, the scheduler works in conjunction with the Resource Information Server (RIS) to gather information about the resources and capabilities of each VM; allowing the scheduler to identify the most suitable resources to fulfill the user's requests.

Our research focus on a task scheduling model where an application is represented as a collection of tasks that can be performed simultaneously. The following assumptions are made.

(1) Each task must be assigned exclusively to one VM.
(2) Tasks can be executed on any suitable VM, and the achieved performance depends on the capabilities of the virtual machine.
(3) Specific parameters need to be calculated for each task on each designated resource.
(4) The tasks are non-preemptive. In other terms, any interruptions will not be allowed once the task is assigned.
(5) A single VM may handle more than one task simultaneously.

For the sake of clarity, an application is represented as a set of independent tasks $T = \{T_1, T_2, \ldots, T_n\}$. Each task T_i has a specific processing requirement, known as task length, measured in Million Instructions (MI). Consider a set of m virtual machines $S = \{VM_1, VM_2, \ldots, VM_m\}$. Each virtual machine VM_j is a set of computing entities with limited capabilities such as CPU power, memory space, storage capacity, and network bandwidth. It's assumed that the VMs are heterogeneous and their CPU processing speed, measured in Millions of Instructions Per Second (MIPS), are used to estimate the execution time of user requests.

The main purpose of scheduling is to handle user requests by allocating the suitable resources that optimize one or more objectives. Commonly, literature reviews

have focused on end-user QoS-based criteria such as makespan and execution cost [24]. However, other service provider QoS parameters that may impact overall system performance are often neglected [23]. Both consumer and provider desires must be considered in the design and implementation of cloud services to ensure that the service is both cost-effective for the consumer and profitable for the provider. This can be a balancing act and requires tradeoffs from both sides. Thus, the aim of this study is to takes these objectives into account by evaluating measures such as makespan, resource utilization, throughput, and balance degree which will be further explained below.

Makespan: it measures the total time required to complete a set of tasks. Commonly sought-after objective in parallel and distributed computing systems to evaluates the efficiency of the designed algorithms in terms of task completion time reduction. It is calculated from the earliest starting time of the tasks to the latest completion time of the tasks. In other words, it represents the longest time taken to complete the assigned application tasks on all available VMs. Mathematically, Makespan is expressed as follows:

$$Makespan = \max_{j \in 1...m} \sum_{i=1}^{n} CT(T_i, VM_j) \tag{1}$$

where m and n are the number of machines and assigned tasks, respectively. The completion time, denoted as $CT(T_i, VM_j)$, represents the execution time of task T_i on a virtual machine VM_j, and can be computed as follows:

$$CT(T_i, VM_j) = \frac{T_i.MI}{VM_j.MIPS} \tag{2}$$

where $T_i.MI$ is the length of the task T_i and $VM_j.MIPS$ is the processing power of the assigned virtual machine VM_j.

Resource utilization: it measures the efficiency and effectiveness of a system's use of its resources. This metric is particularly important for cloud providers as it directly affects their profitability. In this study, we evaluated the performance of the proposed approach using resource utilization in terms of average Resource Utilization Rate (RUR) according to the following equation [9, 13, 25, 26]:

$$RUR = \frac{\sum_{j=1}^{m} CT_j}{Makespan * m} \tag{3}$$

where CT_j represents the completion time of the $j\text{-}th$ virtual machine after executing its last task. A higher rate indicates that the system is making better use of its resources, while a lower rate suggests that the resources are being underutilized.

Degree of imbalance: it is a metric that assesses the imbalance of cloud workload distribution across the available virtual machines based on their capabilities [27]. It aims to ensure an even distribution of workload and prevent bottlenecks in the system, so that no resources are overloaded while others are under loaded or idle. The degree of imbalance may be calculated as follows.

$$Deg\_Imbalance = \frac{VM\_CT_{max} - VM\_CT_{min}}{VM\_CT_{avg}} \tag{4}$$

where VM_CT_{max} and VM_CT_{min} represent the achieved longest and shortest completion time, respectively. VM_CT_{avg} stands for the average completion time of all virtual machines. Higher $Deg\_Imbalance$ indicates a greater imbalance among resources, which can negatively impact system performance. Conversely, a lower $Deg\_Imbalance$ indicates a more balanced distribution of resources, which can lead to better performance.

Throughput: it is a measure of the rate at which a particular system or process is able to complete workloads. It is usually measured in terms of the number of completed tasks or the amount of data processed within a specific time frame. A high throughput value indicates that a system or a process can handle a large amount of data in a short amount of time. This is generally considered to be a desirable characteristic, as it can result in faster response times and increased productivity for the system as a whole. Conversely, a low throughput value may reflect that a system or process is unable to keep up with the assigned demands, which can lead to delays and decreased productivity. The throughput is calculated by the following formula [28].

$$Throughput = \frac{Number\ of\ tasks}{Makespan} \tag{5}$$

In summary, given (1) an IaaS infrastructure provider that offers a set of virtual machines and (2) a workload consisting of a set of tasks that need to be executed on those virtual machines. Our study aims to orchestrate the assignment of tasks to VMs while satisfying trade-offs relationship between the provider's benefits and the user's QoS requirements. The QoS parameters considered are makespan, resource utilization, degree of imbalance and throughput. Due to the conflicting nature of these QoS, the workload scheduling can be presented as the following multi-objective optimization problem.

$$\begin{cases} Minimize\ Makespan \\ Maximize\ Resource\ Utilization\ Rate \\ Minimize\ Degree\ of\ Imbalance \\ Maximize\ Throughput \end{cases}$$

4 Background information on FPA and GWO algorithms

In this section, we outline the main features of the flower pollination algorithm as well as the gray wolf optimizer, which will be the focus of our proposal.

4.1 Flower pollination overview

Flower Pollination Algorithm (FPA) is a metaheuristic optimization algorithm inspired by the pollination process in flowers. It was originally introduced in 2012 by Xin-She Yang [29]. Similar to other metaheuristic algorithms, FPA is a population-based algorithm that maintains a group of solutions called 'flowers' and iteratively improves them through the pollination process.

The pollination process can take two forms, biotic and abiotic, depending on the pollen transfer mechanisms. Biotic pollination, which occurs in about 90% of flowering plants, refers to the transfer of pollen from one plant to another by living organisms, such as insects, birds, bats, bees or other animals [29]. On the other hand, abiotic pollination refers to the transfer of pollen by non-living agents, such as wind or water.

In the context of the FPA, the term 'pollination' refer to the process of exchanging information between solutions in order to update and improve their quality. This process is inspired by the way in which pollination occurs in nature, but it does not involve the transfer of pollen or any other physical substance. There are two types of pollination that can occur [29]:

Global pollination: it involves exchanging information between solutions across the entire population process using a Markovian stochastic process called a Lévy flight [30]. 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.

Local pollination: it entails the information exchange between solutions within a specific region using neighborhood search techniques.

Both types of pollination can be used in FPA to improve the quality of the solutions. Global pollination allows a wide range of solutions to be explored and identified, while local pollination refines and improves the quality of solutions within a specific region of the solution space. The appropriate type of pollination to use will depend on the characteristics and requirements of the optimization problem to be solved. The pseudo-code of FPA approach is presented in Algorithm 1.

The algorithm begins by initializing a population of solutions, where each solution is represented by a flower, and each flower is expressed by a set of decision variables. The algorithm then mimics the pollination process to iteratively enhance the solutions. The pollination process is done by moving each flower's decision variables towards the decision variables of a randomly chosen flower, or to the best flower (global best solution found so far) with certain probability. In other terms, the application of global or local pollination is determined by a random number generation process. If the generated number is less than a certain probability p, then the overall pollination will be applied as follows:

$$x_i^{t+1} = x_i^t + L\left(x_i^t - g_*\right) \tag{6}$$

where x_i^t is a solution i at iteration t, x_i^{t+1} is the solution vector generated at step $t+1$. This latter is determined using the previous solution vector x_i^t and the best

Algorithm 1 FPA pseudo-code

Objective function min or max $f(x), x = (x_1, x_2, ...)$
Define a switch probability $p \in [0, 1]$
Generate initial population of flowers randomly
Find the best solution g_* in the initial population
while Not (stopping_criteria) **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 $x_i^{t+1} = x_i^t + L\left(x_i^t - g_*\right)$
        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 $x_i^{t+1} = x_i^t + \varepsilon\left(x_j^t - x_k^t\right)$
        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
return g_*

current solution found among all the solutions in the generation g_*. The parameter L is a step size involving the use of a Lévy distribution [30], which is a method used to simulate unpredictable insect movement patterns over long distances, that can be mathematically formulated as follows:

$$L \sim \frac{\lambda \Gamma(\lambda) \sin\left(\frac{\pi\lambda}{2}\right)}{\pi} \frac{1}{s^{1+\lambda}}, \quad (s > 0) \tag{7}$$

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

$$s = \frac{U}{|V|^{1/\lambda}} \tag{8}$$

where the values U and V are derived from a normal Gaussian distribution with a mean of zero and standard deviations σ_u and σ_v, respectively.

$$U \sim \left(0,\ \sigma_u^2\right), \quad V \sim \left(0,\ \sigma_v^2\right) \tag{9}$$

$$\sigma_u = \left[\frac{\Gamma\left(1+\lambda\right)}{\lambda \Gamma\left((1+\lambda)/2\right)} \frac{\sin\left(\frac{\pi\lambda}{2}\right)}{2^{(\lambda-1)/2}}\right]^{1/\lambda}, \quad \sigma_v = 1 \tag{10}$$

Otherwise, if the generated random number is greater than the value p, then the local pollination procedure is performed to create new solutions by combining existing ones according to the following equation:

$$x_i^{t+1} = x_i^t + \varepsilon \left(x_j^t - x_k^t \right) \tag{11}$$

where x_j^t and x_k^t are two different solutions chosen randomly among the population and $\varepsilon \in [0, 1]$ is a random number to make the selection closer to a local random walk [29]. Then, the fitness value of the new solution is evaluated, and subsequently, the best solution from the current population is selected by evaluating the new generation. This best solution is carried over to the next iteration, and the search process will be repeated until the stopping criterion is satisfied.

4.2 Grey wolf optimizer overview

Grey Wolf Optimizer (GWO) is a metaheuristic optimization algorithm inspired by the social hunting behavior of grey wolves in the wild. Originally, proposed in 2014 by Mirjalili and Lewis [31], it works by maintaining a group of potential solutions called "wolves", which are used to explore the search space and identify the best solution. The mathematical model of the GWO algorithm's is split into four phases which are liste as follows [31, 32]:

Social Hierarchy: the gray wolves in a pack establish a social hierarchy through a dominance-based mechanism, where the strongest wolf, the alpha, has the most control and makes decisions for the rest of the pack. The second strongest one, the beta, acts as a consultant of the alpha. The third strongest Wolf, the delta, is a subordinate who submits to the upper levels (alpha and beta) while serving as a leader to the omega wolf, the lowest-ranking members of the pack.

Each wolf's fitness value defines its ranking in the hierarchy during the social hierarchy phase. The strongest wolves are designated as α, beta β and delta δ, respectively, which play specific roles in the following stages and the remaining pack members considered as omega(s).

Encircling Prey: once the pack leaders are established, the pack begins to encircle their prey by moving in a circular motion. The leaders guide the pack towards the prey, and the other members follow them. This encircling behavior is mathematically modeled in the following equations.

$$\begin{aligned} \vec{X}(t+1) &= \vec{X_p}(t) - \vec{A}.\vec{D} \\ \vec{D} &= |\vec{C}.\vec{X_p}(t) - \vec{X}(t)| \end{aligned} \tag{12}$$

where, $\vec{X}$ represents the position of the current gray wolf and $\vec{X_p}$ the position of the prey. The coefficient vectors $\vec{A}$ and $\vec{C}$ are computed as follows:

$$\vec{A} = 2.\vec{a}.\vec{r_1} - \vec{a} \tag{13}$$

$$\vec{C} = 2.\vec{r_2} \tag{14}$$

where, the vector $\vec{a}$ decreases progressively in linear way to emphasize exploration and exploitation, respectively, while $\vec{r_1}$ and $\vec{r_2}$ are arrays of random real numbers generated within the range of $[0, 1]$.

Hunting: as mentioned above, the leader wolves hold the most favorable positions in the population and play a crucial role in guiding the hunt. The first three best agents are recorded, and a reference position for the prey is calculated using α, β, and δ. This estimated prey position serves as a guide for the other wolves to adjust their positions coordinately, as outlined in the following equations.

$$\vec{D_\alpha} = |\vec{C_1}.\vec{X_\alpha} - \vec{X}|, \quad \vec{D_\beta} = |\vec{C_2}.\vec{X_\beta} - \vec{X}|, \quad \vec{D_\delta} = |\vec{C_3}.\vec{X_\delta} - \vec{X}| \tag{15}$$

$$\vec{X_1} = \vec{X_\alpha} - \vec{A_1}.\vec{D_\alpha}, \quad \vec{X_2} = \vec{X_\beta} - \vec{A_2}.\vec{D_\beta}, \quad \vec{X_3} = \vec{X_\delta} - \vec{A_3}.\vec{D_\delta} \tag{16}$$

$$\vec{X}(t+1) = \frac{\vec{X_1} + \vec{X_2} + \vec{X_3}}{3} \tag{17}$$

Attacking: In this final phase, the pack attacks their prey and attempts to catch and kill it. The gray wolves then use their abilities to track it and potentially find the best possible outcome. More specifically, the abilities of the wolves can result in finding efficient global solution. Since the coefficient $\vec{A}$ plays a crucial role in this process, a value $|A| < 1$ prompts the wolves to focus on capturing the current prey (exploitation), while a value $|A| > 1$ means that the gray wolves will diverge from the prey to seek more suitable prey (exploration).

The four phases of GWO optimization algorithm are used to find the near-optimal solution for a given problem. Each phase corresponds to the actions of the grey wolves in the pack, where leader wolves correspond to the best solution found so far and the followers represent to the remaining solutions in the population. The pseudo-code of GWO algorithm is outlined in Algorithm 2.

5 The proposed approach

To present our approach, two fundamental concepts must be set up (1) a suitable representation of the solutions handled by the algorithm and (2) the evaluation of these solutions in the objective space, while remaining generic enough to be compatible and consistent with a large number of QoS measuring points.

5.1 Solution representation

The efficient design of a metaheuristic-based approach requires a relevant representation of a solution, which is crucial in determining the algorithm's scuccess. Indeed, the representation has an important impact on the way the solutions will be handled by the search operators and the evaluation stage. Several representations can be found for a given problem. As an illustration, Figure 2 presents a simplified version of the solution. The discrete value vector is adopted, where task T1 will be assigned to VM4, task T2 will be assigned to VM1, and so on.

Algorithm 2 GWO pseudo-code

Initialize the grey wolf population $X = (x_1,\ x_2,\ ...)$
Initialize $a,\ A$ and C
Compute the fitness of each search agent
First best agent denoted as X_α
Second best agent denoted as X_β
Third best agent denoted as X_δ
while stop criterion **do**
    for each $i = 1 : n$ (all n search agent in population) **do**
        Update the position of the current search agent using Eq. (17)
    end for
    Update a, A and C
    Calculate the fitness value of all search agent
    Update the positions of X_α, X_β and X_δ
end while
return X_α

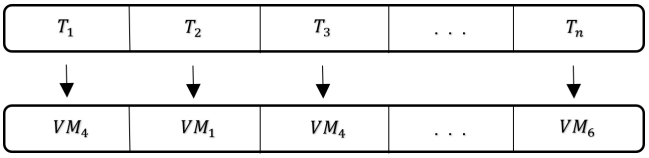

Fig. 2 A simplified representation of a solution

5.2 Fitness objectif

The evaluation step is a crucial aspect in the design of an optimization algorithm. It allows to guide the method towards the right solutions of the search space. In a multi-objective optimization problem, there are several objective functions, thus the evaluation involves assigning a vector of values to each solution, with the vector size matching the number of objectives considered. Each element of the vector represents the quality of the solution with respect to the corresponding objective function (see section 3). In our work, we employed a weighted sum aggregation fitness function, which is widely used in the literature and can be adapted to various numerical objectives due to its simplicity. The used Multi-Objective Function (MOF) considers the four aforementioned objectives as defined in Eq. (18).

$$MOF = w_1\ Makespan + w_2\ RUR + w_3\ Deg\_Imbalance + w_4\ Throughput \quad (18)$$

where w_1, w_2, w_3 and w_4 represent the weighting factors that reflects the importance or requirement for each criterion. In this study, with all objectives given an equal weight of 0.25, to ensure fairness and avoid any bias towards any specific objective.

It should be noted that, we have four QoS metrics to optimize; these metrics are divided into two categories: QoS metrics to maximize (resource utilization rate, throughput) and those to minimize (makespan, degree of imbalance). For both cases,

the QoS have very different scales and values. Therefore, normalization is necessary so that the fitness function is not biased by one of the metrics having a high value. In our approach, the normalized value for a given quality of service metric QoS(i) is given by Eq. (19) as in [33, 34].

$$QoS_{iNormalized} = \begin{cases} \frac{Max\_QoS_i - QoS_i}{Max\_QoS_i - Min\_QoS_i} & \text{if } QoS_i \textit{ to be minimized} \\ \\ \frac{QoS_i - Min\_QoS_i}{Max\_QoS_i - Min\_QoS_i} & \text{if } QoS_i \textit{ to be maximized} \end{cases} \tag{19}$$

where, the value Max_QoS_i and Min_QoS_i are updated continously at the time of scheduling process mirroring the worst and the best metric QoS_i at a particular iteration.

5.3 Proposed algorithm

Recall that our main aim is to assign each task to a specific virtual machine in order to optimize performance in a cloud environment. This includes reducing the time makespan or schedule length, degree of imbalance, and increasing resource utilization as well as throughput objective.

The basic assumption behind this research is that tackling cloud scheduling optimization problems can be further enhanced by using swarm intelligence algorithms. For this purpose, the flower pollination algorithm was adopted, with the exploration phase enhanced by the grey wolf optimizer and the crossover operators of genetic algorithms to avoid getting stuck in local optima and balance exploration and exploitation. The pseudo-code of the proposed hybrid multi-objective approach is presented in the Algorithm 3.

The algorithm starts by defining its control parameters, including switching probability, population size, maximum number of iterations, the number of submitted tasks and the number of available VM. In steps 2 and 3, a random set of solutions, which relates to the initial population, is generated. The solution is created based on a number of task-user and virtual machines according to the proposed solution encoding. Initially, these tasks are randomly sent to the different virtual machines available in the data centre. Then, the solutions are updated based on the employed strategy. After generating the solutions, we evaluate each one using the fitness function outlined in Eq. (18). The best solution is then identified and saved as S_α. At each iteration step in the inner while loop (lines 4 to 33), for every solution, a new solution is generated using either global or local pollination, depending on a randomly generated switching probability p. The new solution is examined and compared to the current one and incorporated into the population if it is better. Otherwise, the proposed algorithm employs an evolutionary algorithm crossover operator (line 18) as in MODE-RMO [35, 36] to enhance the exploitation process and discover better solutions in the neighborhood by combining the current solution with the reference vector at a given iteration.

Algorithm 3 Proposed hybrid algorithm

Initialize parameters: population size (n), maximum number of iteration, and switch probability $p \in [0, 1]$
Generate the initial population $S_i,\ i = 1,\ 2,\ \ldots,\ n$
Find the best solution S_α in the initial population
while stop criterion **do**
    for each element in the population **do**
        if $rand() < p$ **then**
            Perform levy flight and draw L (step vector)
            Perform global pollination using Eq. (6)
        else
            Draw ε from a uniform distribution in $[0, 1]$
            Randomly choose two elements S_j and S_k from all solutions in the population
            Perform local pollination using Eq. (11)
        end if
        Evaluate the new solution $S_i^{'}$
        if new generated solution is better **then**
            replace S_i with $S_i^{'}$
        else
            Apply the crossover operation on the current element S_i of the population using Eq. (20), and then evaluate the new solution $S_i^{'}$
            if new generated solution is better **then**
                replace S_i with the newly generated $S_i^{'}$
            end if
        end if
    end for
    Sort population in the order of fitness
    Update the best element S_α
    Select the second best element S_β
    Select the third best element S_δ
    Determine the worst element S_w
    Apply Eq. (17) on the worst candidate solution
    if new generated solution better **then**
        Replace worst solution S_w by new solution $S_w^{'}$
    end if
end while
return S_α as the best solution in the serach space

The implemented crossover operator requires a reference vector R and a target vector T in a D-dimensional space, where D represents the workload size. The former is chosen as the best solution from the current population. The crossover operation is

shown in Eq. (20) with $i = 1,\ 2,\ 3,\ \ldots,\ D$.

$$T_i^{'} = \begin{cases} T_i \ \text{ if } (rand \leq CR \ or \ i == s) \\ \\ R_i \ \text{ otherwise} \end{cases} \tag{20}$$

where, the crossover rate constant CR is a value between $[0, 1]$, s and $rand$ are random numbers generated from $[1, D]$ and $[0, 1]$, respectively.

The proposed algorithm will then employs the GWO strategy to improve the worst-case solution (lines 24 to 29). According to this strategy, the three best solutions are selected based on their fitness values, and a new solution is calculated using Eq. (17) by varying the value of a during each iteration of the algorithm execution. This strengthens the exploration process to find other potential solutions in the space, while ensuring that the solutions' quality improves over time and does get trapped in a local optimum. Finally, this process is repeated until the algorithm converges or the maximum number of iteration is reached. Note that the arrived tasks, will not be immediately assigned to the VMs, a pre-simulation phase of the mapping process using the proposed hybrid version is required to find the suitable VM for each task before sending it to the broker.

5.4 Time complexity analysis

Since meta-heuristics heavily rely on several tuning parameters, computing the precise asymptotic complexity value may not be obvious. However, we can establish a lower bound for the worst-case execution time. In the basic flower pollination algorithm, the time complexity is mainly influenced by the maximum number of iterations ($IterMax$), the population size (N) and the search agent dimension (D). Consequently, the time complexity of FPA can be expressed as $\mathcal{O}(IterMax * N * D)$. Now, let's calculate the time complexity of the proposed hybrid algorithm compared to FPA. The evolutionary algorithms' crossover operators contribute to a time complexity increase of $\mathcal{O}(IterMax * N * D)$, while the gray wolf strategy leads to an increase of $\mathcal{O}(IterMax.D)$. Therefore, the total time complexity of the proposed algorithm can be expressed as $\mathcal{O}(IterMax * N * D) + \mathcal{O}(IterMax * N * D) + \mathcal{O}(IterMax * D)$.

Therefore, if we need to cope with reasonable instance sizes, we derive a lower bound of $\mathcal{O}(IterMax * N * D)$ for the asymptotic time complexity of the proposed hybrid algorithm, which is equivalent to the computational time complexity of FPA and GWO.

6 Simulation results and discussions

In this section, we discuss the experimental evaluation of the proposed hybrid optimization algorithm. We detail the different parameters concerning the user workload, the provider IaaS infrastructure model, as well as the metrics of the compared algorithms.

Table 2 Task type of the synthetic dataset

Task type	MI range	Distribution
Tiny	200	35%
Small	1 000	40%
Medium	5 000	5%
Large	15 000	15%
Extra large	45 000	5%

Table 3 Task type of the GoCJ dataset

Task type	MI range	Distribution
Small	15 000 - 55 000	20%
Medium	59 000 - 99 000	40%
Large	101 000 - 135 000	30%
Extra large	150 000 - 337 500	6%
Huge	525 000 - 900 000	4%

6.1 Datasets and compared algorithms

In order to investigate and assess the efficiency of the proposed algorithm, a well-studied set of various benchmark data sets are selected from the literature [37, 38]. This benchmark data sets covers a realistic cloud workload and two traces of standard benchmark data sets commonly used for performance evaluation in parallel and distributed systems, namely the High-Performance Computing Center North (HPC2N) and the Google Cloud Jobs (GoCJ) dataset.

6.1.1 Synthetic data set

Synthetic data refers to artificially generated workloads used as a model to evaluate a system under study. Researchers in the scheduling community use synthetic datasets with tasks requiring different processing times to assess the effectiveness of their approaches. [16, 20, 23, 33, 39–41]. Likewise, for the sake of comparison, we have also employed synthetic workloads which are similar and in line with those used in [12]. The synthetic data set is generated by using different proportions and distributions of tiny, small, medium, large and extra-large sized cloud workloads, measured in MI, as shown in Table 2.

6.1.2 GoCJ data set

Hussain and Aleem [37] have made a publicly available real cloud dataset derived from the workload behaviors observed in google cluster traces. A sample original GoCJ dataset is formulated and archived on the Mendeley data repository [42] to allow other researchers to compare the performance of their newly developed algorithms using an open-source benchmark. GoCJ is made up of different files containing data about task sizes in terms of MI. Each file contains different set of rows, equivalent to the number of jobs, denoting task length in MI. These values range from 15 000 MI to 900 000

Table 4 Cloudsim simulation parameters

Entity type	Parameter	Value
Data-center	No. of data-centers	2
Host	No. of hosts	5
	PES	4 (Quad core) 26 (Quad core)
	MIPS	4 000
	RAM	8 GB
	Storage	1 TB
	Bandwidth	2800
Virtual Machine	No. of VMs	25
	MIPS	100 - 1 000
	RAM	1 GB
	Storage	10 000
	Bandwidth	1000
	Policy type	Time Shared

MI. This dataset is composed of five types of tasks with different proportions based on the analysis of Google cluster traces, as presented in Table 3.

6.1.3 HPC2N data set

High Performance Computing Center North (HPC2N) is a national center for Scientific and Parallel Computing that is part of the Swedish National Infrastructure for Computing (SNIC). It provides high-performance computing resources and services to researchers and organizations in various fields, including engineering, social and life sciences. The center offers training and support for researchers on how to use the resources as well as access to a wide range of software tools for data analysis and simulation. As recommended, the cleaned log HPC2N-2002-2.2-cln.swf is used in our experiments, which took three and a half years of accounting records from the HPC2N in Sweden [38].

The results and the performance of the proposed algorithm are compared to the well-known and well-established optimization strategies, namely the TSGWO [12], GGWO [10], FPA [13] and LPGWO [11] algorithms, using four evaluation metrics: time mekespan, cloud resource utilization, degree of imbalance and throughput. These algorithms cover both recently proposed techniques such as OBL and chaotic mapping [43], along with relatively the most utilized optimizers in the field such as GA, FPA and GWO.

6.2 Experimental setup

Our experiments are conducted using CloudSim framework [44], which is widely used by the research community for modeling and simulating large-scale cloud infrastructures [45]. Here, for more realism, the parameters of the performed simulations are carried out in a heterogeneous cloud environment as put forward in [12]. Details of the experimental setup are also reported in Table 4.

Table 5 Parameter settings of the algorithms

Parameter	Value
Population size	100
p	0.8
λ	1.5
CR	0.7

For the sake of comparison, we assessed and recorded the results based on the performance of the evaluated algorithms over 10 independent runs and then the average of the result is reported. The control parameters of the compared scheduling algorithms are depicted in Table 5. The parameter values of our approach were determined experimentally on three real workload datasets using different values, with the chosen values yielding the best overall results. To ensure the fairness of the experiments, the initial population of all algorithms is set to 100. Furthermore, it is worth noting that the parameters settings used in our study of the compared algorithms are chosen in such a way that they are representative and are in line with those used in the literature [10], [13], [12] and [11].

6.3 Results discussion

This sub-section presents the discussion of the experimental results of the proposed hybrid task scheduling approach. The conducted experiments aim to assess the effectiveness of our proposal in generating a set of trade-off solutions between different QoS metrics. As stated above, we compare the performance of the proposed algorithm with four well-known metaheuristic algorithms, namely FPA [13], LGWO [11], GGWO [10] and TSMGWO [12]. Moreover, an assessment of three workload datasets is used to investigate the impact of workload on the behavior of the proposed algorithm while plotting graphs on the basis of the considered performance metrics: makespan, resources utilization, degree of imbalance and throughput. We note that the experimental results are performed on 100, 200, 300, 400, 500 and 600 tasks, no matter which workload traces used. In the further, we show the detailed results of the proposed hybrid algorithm compared to the other concurrent metaheuristic methods.

Table 6 Results of makespan for the proposed approach and compared algorithms

Dataset	Cloudlets	Methode used				
		TSMGWO	GGWO	FPA	LPGWO	Proposed approach
Synthetic	100	108.321	92.861	83.345	82.468	69.469
	200	219.925	169.835	160.571	151.447	98.605
	300	323.474	248.674	232.676	223.067	134.33
	400	428.047	331.827	316.556	288.724	175.618
	500	611.74	445.393	385.27	364.556	222.535
	600	735.826	557.136	461.94	443.692	255.253
GoCJ	100	2819.195	1933.13	1907.907	1802.508	1234.286
	200	5500.452	4138.394	3807.895	3652.926	2175.773
	300	9173.203	7036.496	5740.561	5541.863	3442.468
	400	12765.681	9875.689	7873.961	7336.892	4234.891
	500	16244.657	12325.339	9925.119	8631.041	5448.076
	600	22082.362	16088.261	13281.753	11875.349	6747.17
HPC2N	100	6480.608	4842.292	4465.57	4375.041	3057.856
	200	10620.089	7220.306	6842.588	6732.044	4531.662
	300	16283.01	11166.643	10224.7	10195.421	5878.406
	400	22457.034	17177.39	15336.954	14195.432	8705.584
	500	31517.353	23746.061	20246.416	18666.703	11434.802
	600	35501.863	26025.866	23894.727	20847.556	12033.746

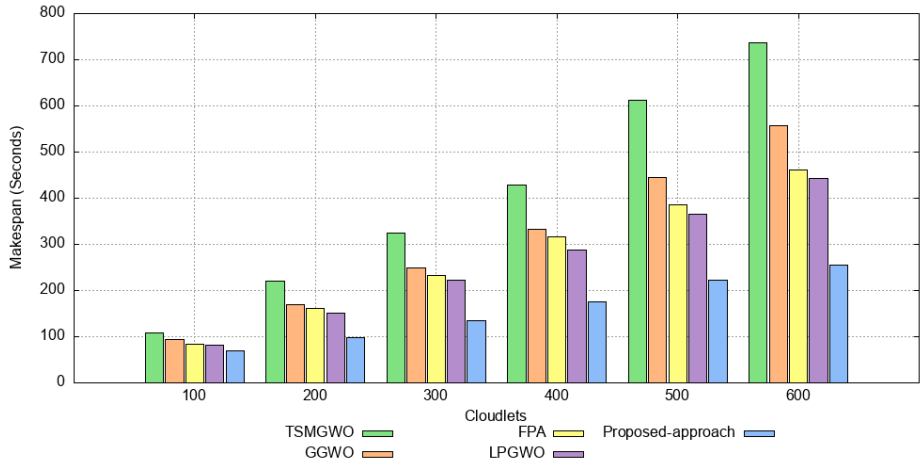

Fig. 3 Comparative results in terms of makespan for synthetic workload

As shown in Figures 3, 4 and 5, which represents the results of the time makespan obtained for the synthetic workload, GoCJ, and the HPC2N real workloads respectively, it can be observed that for all identified benchmark datasets, the proposed algorithm achieves the best results and its performance remains consistently superior to other competitors. This clearly shows that the competing algorithms are unable to reach the optimal or a point close to it. On the contrary, the proposed algorithm produces the best time makespan values across all tested instances with different number of tasks.

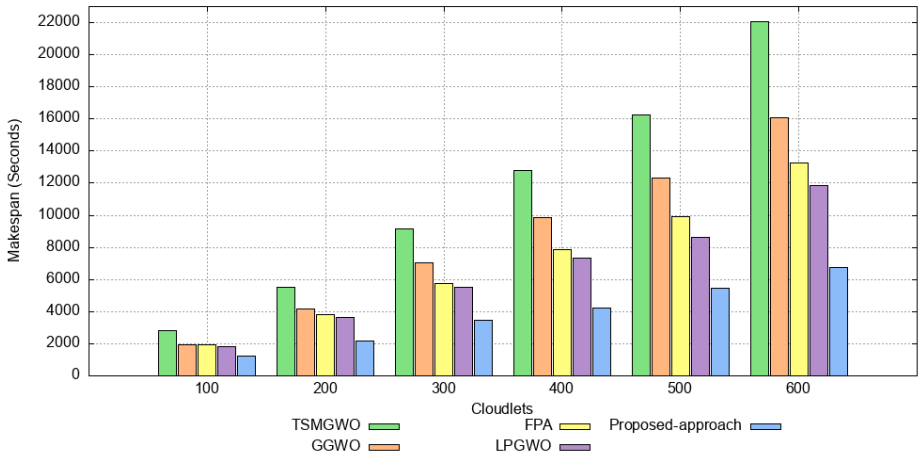

Fig. 4 Comparative results in terms of makespan for GoCJ workload

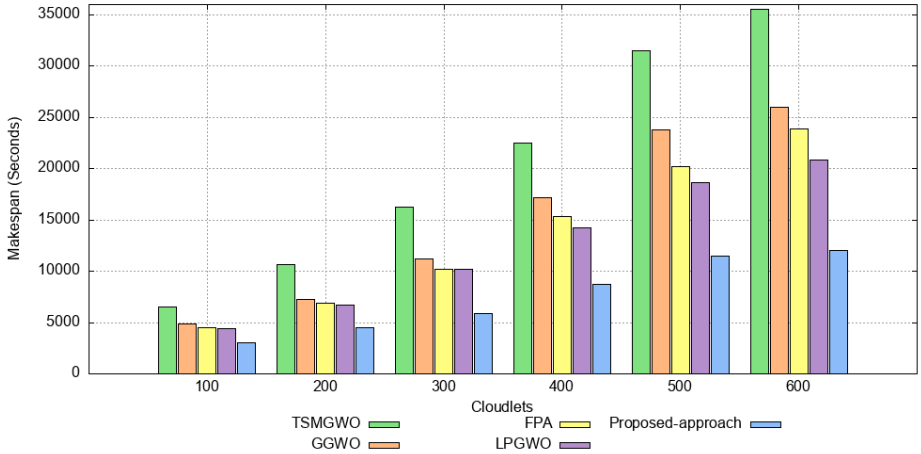

Fig. 5 Comparative results in terms of makespan for HPC2N workload

According to the results in Table 6, we can see that the proposed approach outperforms its direct competitor techniques with different rates, e.g., for the synthetic workload implementation, the reduction rate of our proposal can reach 65.34%. Similarly, with the implementation of GoCJ and HPC2N workloads, this rate may rise to 69.44% and 66.10% respectively. We can also observe from Table 6, that the performance results of the compared methods decreases significantly when the number of tasks goes up. For instance, in HPC2N workload, the time makespan of the proposed algorithm is 3057.856, 4531.662, 5878.406, 8705.584, 11434.802 and 12033.746 for 100, 200, 300, 400, 500 and 600, respectively. However, this is not surprising, given the fact that VMs have more load which influences the schedule length of the task mapping.

Table 7 Results of resource utilization rate for the proposed approach and the compared algorithms

Dataset	Cloudlets	Methode used				
		TSMGWO	GGWO	FPA	LPGWO	Proposed approach
Synthetic	100	36.713	41.615	45.618	46.736	63.012
	200	40.198	46.488	49.006	49.833	79.317
	300	40.052	48.887	52.211	54.64	84.612
	400	41.41	49.185	52.66	54.332	86.087
	500	38.842	47.765	53.522	54.997	85.203
	600	38.239	47.201	53.597	53.944	87.844
GoCJ	100	39.337	50.505	50.998	54.397	77.98
	200	41.692	51.498	54.774	55.722	87.004
	300	39.667	47.858	55.609	56.51	85.657
	400	37.337	45.192	53.937	55.166	86.711
	500	37.08	45.858	53.076	56.812	85.708
	600	35.105	44.822	51.87	54.683	87.011
HPC2N	100	39.989	46.525	51.584	51.921	77.258
	200	39.527	50.347	53.144	54.429	79.726
	300	38.486	50.468	53.814	55.098	88.022
	400	40.452	48.505	53.931	54.53	85.804
	500	40.191	49.073	54.929	57.552	88.654
	600	37.444	47.351	50.167	54.468	88.065

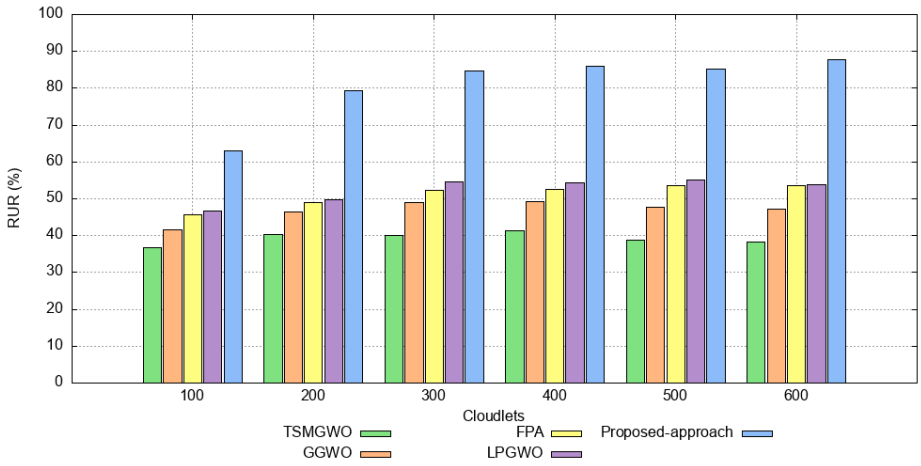

Fig. 6 Comparative results in terms of resource utilization rate for synthetic workload

From Figures 6, 7 and 8, which depict the resource utilization rate of assigning different cloudlets to VMs, it can be seen that the proposed algorithm produces better performance rate over the TSMGWO, LPGWO, GGWO and the conventional FPA algorithm for both synthetic and standard workload traces.

As can be shown in Table 7 which illustrates also the results of RUR values, it is apparent that the proposed hybrid technique can result in a maximum increase in

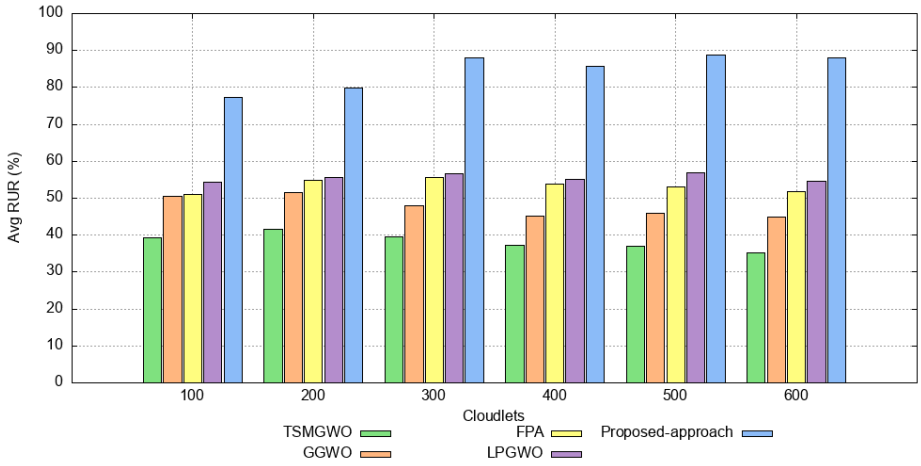

Fig. 7 Comparative results in terms of resource utilization rate for GoCJ workload

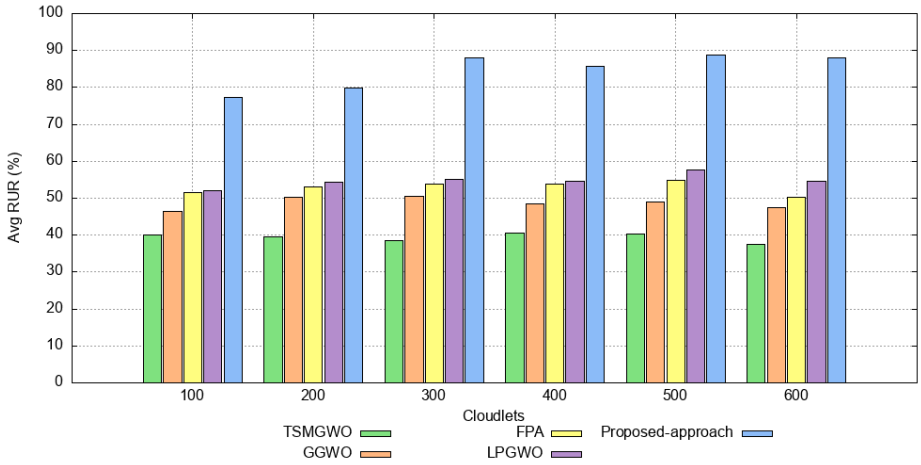

Fig. 8 Comparative results in terms of resource utilization rate for HPC2N workload

RUR up to 59.65% in all the instances considered in this work. Moreover, our proposal gives a higher RUR rate than the TSMGWO, GGWO, LPGWO and FPA with a value greater than 88%. The mains reason behind the better performance of the new designed approach is that it uses more resources to reduce the time makespan which leads to an efficient use of teh cloud resources during the scheduling process.

Comparing the results of the TSMGWO, LPGWO, GGWO and FPA with the obtained ones of our proposal in terms of degree of imbalance, we can observe in Table 8 and from Figures 9, 10 and 11, respectively, that the latter provides high performances with a substantial improvement over the other compared metaheuristics in all tested scenarios. For example, in the case of HPC2N dataset, the proposed hybrid based approach achieves solutions that are 34,49%, 41,73%, 45,35% and 53,33%

Table 8 Results of degree of imbalance for the proposed approach and the compared algorithms

Dataset	Cloudlets	Methode used				
		TSMGWO	GGWO	FPA	LPGWO	Proposed approach
Synthetic	100	5.698	5.636	5.013	4.825	3.423
	200	6.811	6.34	5.838	5.573	3.604
	300	7.394	6.558	6.057	5.828	3.732
	400	8.054	6.586	6.388	5.905	3.78
	500	8.306	7.266	6.37	6.205	3.998
	600	9.396	7.694	6.501	6.287	4.051
GoCJ	100	2.862	2.536	2.457	2.295	1.518
	200	3.295	2.86	2.722	2.6	1.615
	300	3.953	3.362	2.78	2.724	1.787
	400	4.1	3.328	2.78	2.558	1.67
	500	4.297	3.415	2.931	2.529	1.782
	600	4.773	3.685	3.222	2.923	1.83
HPC2N	100	4.293	3.949	3.639	3.595	2.527
	200	5.83	5.318	4.938	4.863	3.452
	300	7.069	5.958	5.395	5.387	3.515
	400	7.215	6.252	5.578	5.231	3.644
	500	7.916	6.584	5.7	5.481	3.703
	600	8.629	7.369	6.912	6.148	4.027

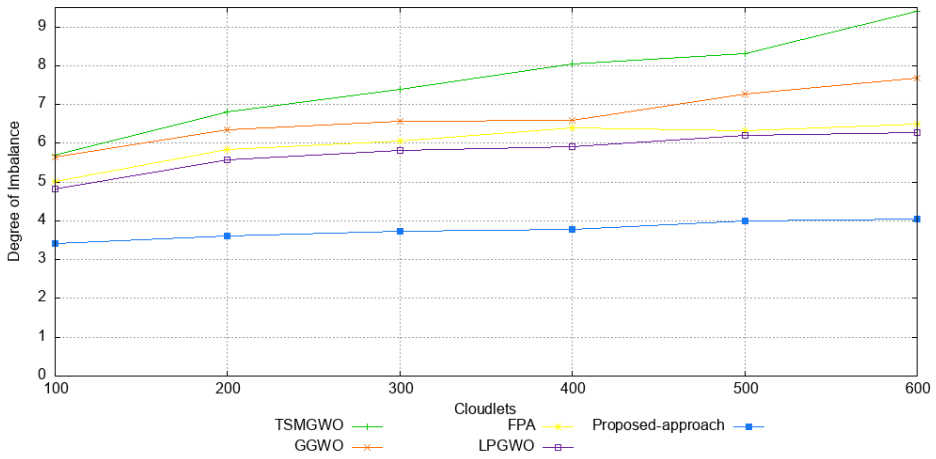

Fig. 9 Comparative results in terms of degree of imbalance for synthetic workload

more efficient in terms of degree of imbalance than those provided by the LPGWO, FPA, GGWO and TSMGWO algorithms respectively. This improvement is due to the policy of the hybrid-based approach which efficiently explores the available number of VMs and strives to ensure a minimal value of the degree of imbalance metric based on resource capabilities. This leads to an efficient balanced distribution of the submitted workload among the available cloud virtual machines, while preventing any VM from being overloaded at any time during the scheduling process.

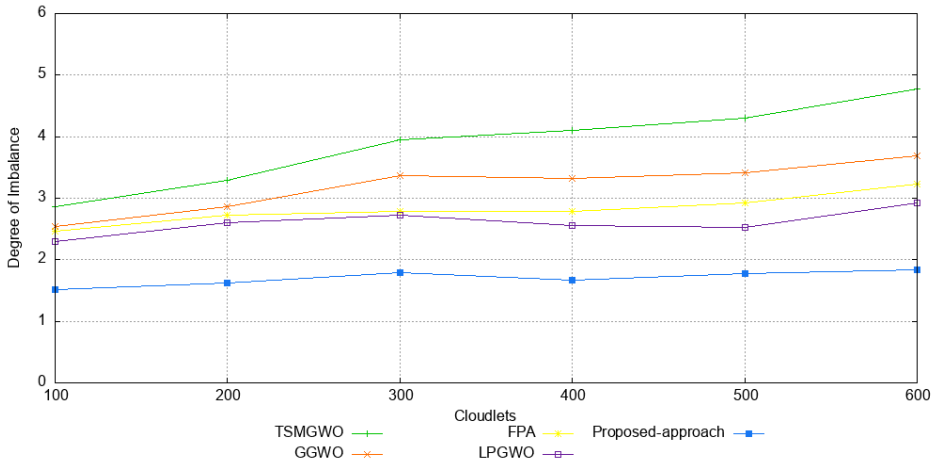

Fig. 10 Comparative results in terms of degree of imbalance for GoCJ workload

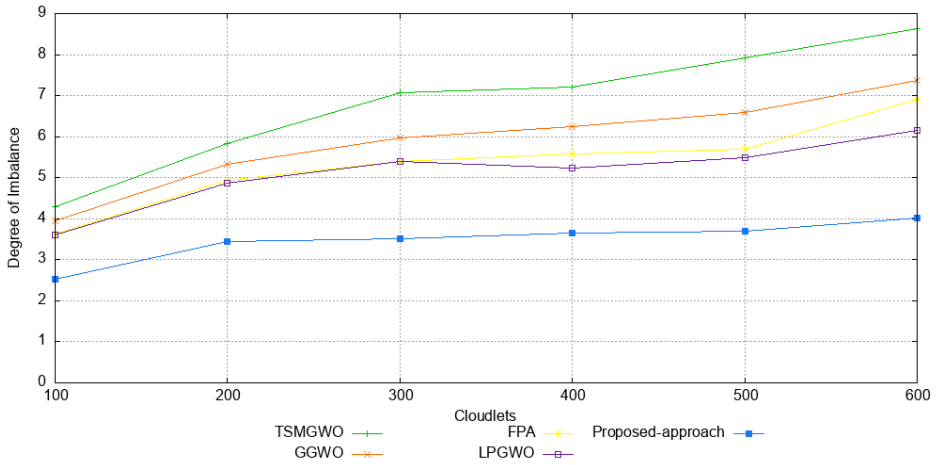

Fig. 11 Comparative results in terms of degree of imbalance for HPC2N workload

According to the findings in Table 9 and from Figures 12, 13 and 14, it is readily apparent that the proposed approach still performs noticeably at maximizing the throughput when compared to those found by LPGWO, TSMGWO, GGWO and FPA optimizers. The results in Table 9 clearly confirm the significant gap between our proposal's outcomes and those of other optimizers across all workload situations examined. For instance, the reduction rate of our strategy in the case of GoCJ traces is 42,04%, 47,72%, 56,81% and 67,04% over than LPGWO, FPA, GGWO and TSMGWO, respectively. This can be explained by the fact that the scheduling policy of the presented study achieves smallest time makespan, which results in an increase of the number of the completed tasks in a given time.

Table 9 Results of throughput for the proposed approach and the compared algorithms

Dataset	Cloudlets	Methode used				
		TSMGWO	GGWO	FPA	LPGWO	Proposed approach
Synthetic	100	0.842	1.074	1.18	1.1209	1.402
	200	0.925	1.18	1.255	1.33	12.041
	300	0.938	1.214	1.296	1.352	2.246
	400	0.955	1.218	1.28	1.395	2.287
	500	0.823	1.127	1.305	1.386	2.252
	600	0.871	1.151	1.253	1.325	2.313
GoCJ	100	0.037	0.052	0.052	0.056	0.08
	200	0.037	0.049	0.053	0.054	0.092
	300	0.032	0.043	0.052	0.054	0.088
	400	0.031	0.042	0.051	0.056	0.093
	500	0.031	0.04	0.051	0.059	0.091
	600	0.029	0.038	0.046	0.051	0.088
HPC2N	100	0.016	0.021	0.021	0.022	0.032
	200	0.02	0.029	0.03	0.03	0.044
	300	0.02	0.027	0.03	0.031	0.05
	400	0.02	0.021	0.027	0.03	0.047
	500	0.017	0.02	0.025	0.028	0.043
	600	0.019	0.022	0.024	0.03	0.049

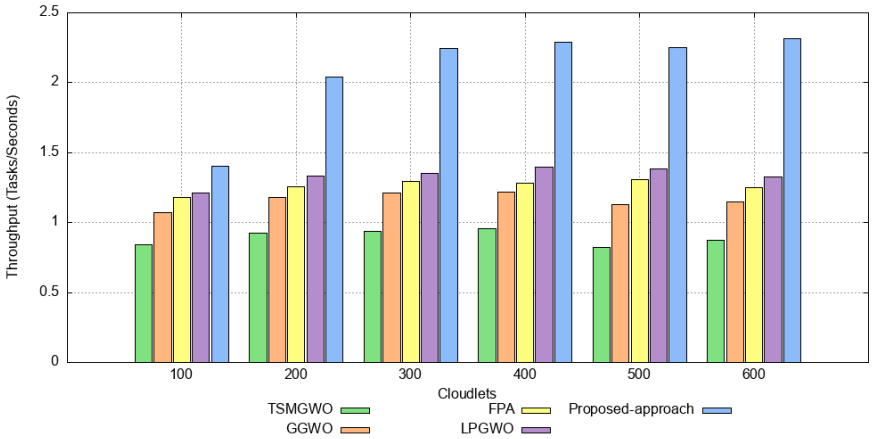

Fig. 12 Comparative results in terms of throughput for synthetic workload

To highlight the effectiveness of our proposed hybrid algorithm, we also compared the convergence behavior of each algorithm over the iterations. The convergence curves of the TSMGWO, LPGWO, GGWO, FPA, and our approach are illustrated in Figure 15. In this experiment, we perform scheduling of 600 tasks from a HPC2N workload. The figure reveal clearly that all the five tested algorithms optimize the objective function. However, the obtained results show the superiority of the new hybrid algorithm in terms of the global convergence behaviour compared to the competing approaches, which tend to be trapped in local optima.

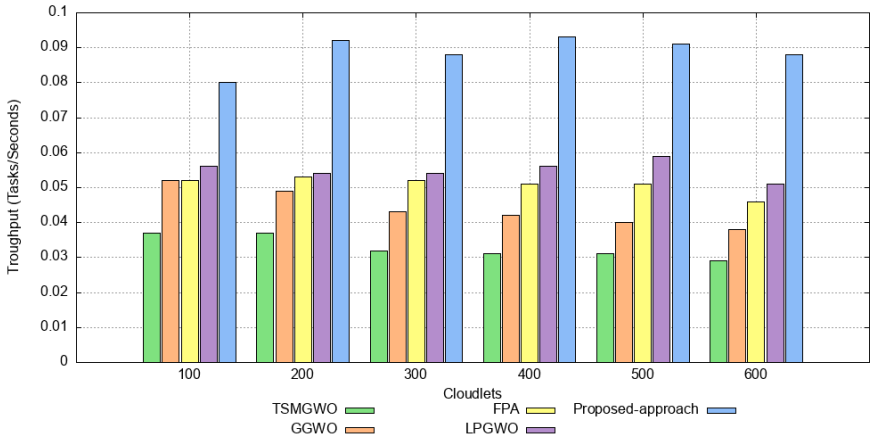

Fig. 13 Comparative results in terms of throughput for GoCJ workload

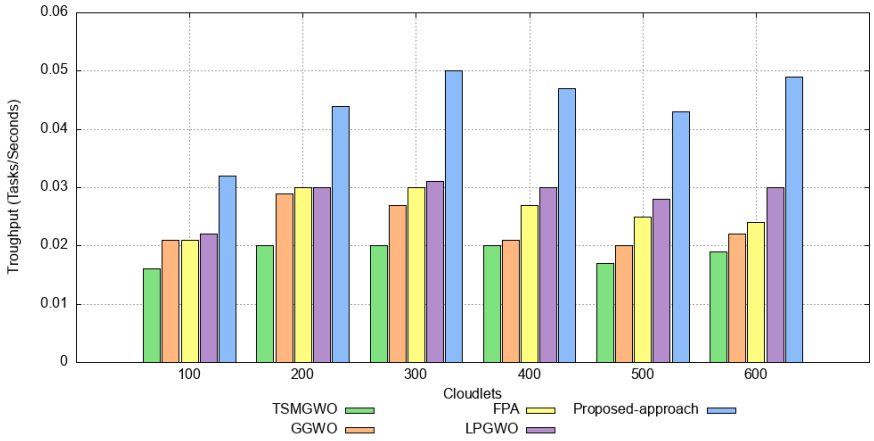

Fig. 14 Comparative results in terms of throughput for HPC2N workload

Moreover, emphasizing the real-time aspect which is crucial as it demonstrates the cloud system's ability to adapt rapidly to a dynamically changing environment, we can observe that the new approach is able to reach better global solutions in faster convergence times than its direct competitors. Thus, the proposed hybrid algorithm successfully strikes a good balance between the stages of exploration and exploitation throughout the search process.

Taken as a whole, it can be noticed that the scalability results on different real workload datasets highlight that the introduced hybrid-based approach maintains a good balance between its propensities for exploration and exploitation, and as a consequence, its performance is constantly superior. Moreover, the obtained results also indicate that the hybrid algorithm finds better task mapping solutions and succeeds to fulfill its objectives in improving the global performances of the scheduling process

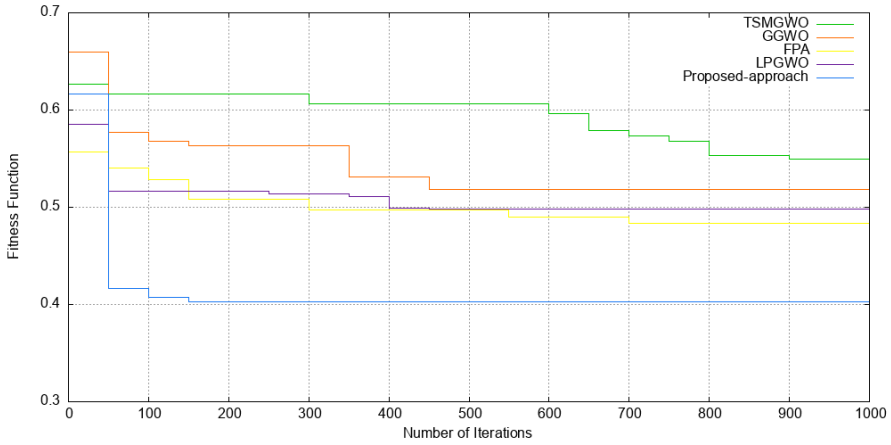

Fig. 15 Convergence behavior for 600 tasks with real HPC2N workload

with respect to various metrics, including time makespan, resource utilization, degree of imbalance, and throughput. The following features provide an explanation for the potential use of the proposed hybrid strategy and its superiority when compared to its direct competitors LPGWO, FPA, GGWO, and TSMGWO.

1. Combining the pollination behavior of flowers and the grey wolf optimizer with the use of evolutionary algorithms crossover operators can achieve harmonious balance between exploration and exploitation stages.
2. Gradually improving the worst individuals in the population over iterations, can be beneficial to avoid the fact of being trapped in local optima during the optimization process.
3. The search strategy based on the first three best solutions in the population to update the worst individuals has a constructive impact on the exploration potential of the algorithm.
4. The strength of the genetic algorithm's crossover operators is able to assist the candidate solutions to exploit promising regions in the search space.

7 Conclusion

In this paper, a novel hybrid optimization algorithm is proposed to tackle the problem of multi-objective task scheduling in heterogeneous cloud systems. The goal is to provide a better orchestration of tasks assignment to virtual machines while meeting the requirements of the end-user on the one hand and the service provider on the other hand. To accomplish this, the optimization process should be able to strike a reasonable balance between exploration and exploitation tendencies in the search space of solutions, otherwise the risk of being trapped in local optima and immature convergence increases. To overcome the limitations of the conventional approaches, an effective resource selection technique is designed to avoid the stagnation phenomenon of local solutions of the search process. The introduced algorithm combines the bio-inspired

pollination behavior of flowers with the swarm intelligence-grey wolf optimizer. Moreover, it makes use of the evolutionary algorithms' crossover operators in order to maintain the diversity of the population. The performance of the new approach is compared with other well-known and established optimization techniques from the literature, like LPGWO, GGWO, TSMGWO and standard FPA techniques. The objective function is valuated in terms of four optimization criteria: time makespan, resource utilization rate, degree of imbalance and throughput. The detailed experimental study and the different considered scenarios using CloudSim framework have shown the usefulness and the potential of the proposed hybrid algorithm.

Although the new studied approach has improved the global performances of the task scheduling process, there are still fields and directions to explore, such as its application to workflow scheduling in real cloud environments. It is also worth continuing to investigate other optimization objectives in cloud computing systems such as fault tolerance and energy consumption as well as load balancing, in order to handle unexpected failures and ensure system's stability. Another possible and potential future research is to apply the proposed hybrid algorithm in different optimization contexts than the one targeted in this paper, such as fog computing, sensor networks, image processing, cognitive radio, data clustering and others.

References

[1] Bokhari, M.U., Makki, Q., Tamandani, Y.K.: A survey on cloud computing. In: Big Data Analytics: Proceedings of CSI 2015, pp. 149–164 (2018). Springer

[2] Toosi, A.N., Sinnott, R.O., Buyya, R.: Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using aneka. Future Generation Computer Systems **79**, 765–775 (2018)

[3] Malti, A.N., Hakem, M., Benmammar, B.: Multi-objective task scheduling in cloud computing. Concurrency and Computation: Practice and Experience **34**(25), 7252 (2022)

[4] Abdullahi, M., Ngadi, M.A., Dishing, S.I., Ahmad, B.I., *et al.*: An efficient symbiotic organisms search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment. Journal of Network and Computer Applications **133**, 60–74 (2019)

[5] Houssein, E.H., Gad, A.G., Wazery, Y.M., Suganthan, P.N.: Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends. Swarm and Evolutionary Computation **62**, 100841 (2021)

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

[7] Ghafari, R., Kabutarkhani, F.H., Mansouri, N.: Task scheduling algorithms

for energy optimization in cloud environment: a comprehensive review. Cluster Computing **25**(2), 1035–1093 (2022)

[8] Ibrahim, I.M., *et al.*: Task scheduling algorithms in cloud computing: A review. Turkish Journal of Computer and Mathematics Education (TURCOMAT) **12**(4), 1041–1053 (2021)

[9] Malti, A.N., Benmammar, B., Hakem, M.: A comparative study of metaheuristics based task scheduling in cloud computing. In: Modelling and Implementation of Complex Systems: Proceedings of the 7th International Symposium, MISC 2022, Mostaganem, Algeria, October 30-31, 2022, pp. 263–278 (2022). Springer

[10] Gobalakrishnan, N., Arun, C.: A new multi-objective optimal programming model for task scheduling using genetic gray wolf optimization in cloud computing. The Computer Journal **61**(10), 1523–1536 (2018)

[11] Gokuldhev, M., Singaravel, G., Ram Mohan, N.: Multi-objective local pollination-based gray wolf optimizer for task scheduling heterogeneous cloud environment. Journal of Circuits, Systems and Computers **29**(07), 2050100 (2020)

[12] Alsadie, D.: Tsmgwo: Optimizing task schedule using multi-objectives grey wolf optimizer for cloud data centers. IEEE Access **9**, 37707–37725 (2021)

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

[14] Pirozmand, P., Hosseinabadi, A.A.R., Farrokhzad, M., Sadeghilalimi, M., Mirkamali, S., Slowik, A.: Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing. Neural computing and applications **33**, 13075–13088 (2021)

[15] Bezdan, T., Zivkovic, M., Bacanin, N., Strumberger, I., Tuba, E., Tuba, M.: Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm. Journal of Intelligent & Fuzzy Systems **42**(1), 411–423 (2022)

[16] Bezdan, T., Zivkovic, M., Antonijevic, M., Zivkovic, T., Bacanin, N.: Enhanced flower pollination algorithm for task scheduling in cloud computing environment. In: Machine Learning for Predictive Analysis: Proceedings of ICTIS 2020, pp. 163–171 (2021). Springer

[17] Miglani, N., Sharma, G., Khurana, S.: Multi-objective reliability-based workflow scheduler: An elastic and persuasive task scheduler based upon modified-flower pollination algorithm in cloud environment. Concurrency and Computation: Practice and Experience **34**(22), 7150 (2022)

[18] Walia, N.K., Kaur, N., Alowaidi, M., Bhatia, K.S., Mishra, S., Sharma, N.K., Sharma, S.K., Kaur, H.: An energy-efficient hybrid scheduling algorithm for task scheduling in the cloud computing environments. IEEE Access **9**, 117325–117337 (2021)

[19] Gokuldhev, M., Singaravel, G.: Local pollination-based moth search algorithm for task-scheduling heterogeneous cloud environment. The Computer Journal **65**(2), 382–395 (2022)

[20] Natesan, G., Chokkalingam, A.: Task scheduling in heterogeneous cloud environment using mean grey wolf optimization algorithm. ICT Express **5**(2), 110–114 (2019)

[21] Khalili, A., Babamir, S.M.: Optimal scheduling workflows in cloud computing environment using pareto-based grey wolf optimizer. Concurrency and Computation: Practice and Experience **29**(11), 4044 (2017)

[22] Khurana, S., Singh, R.: Workflow scheduling and reliability improvement by hybrid intelligence optimization approach with task ranking. EAI Endorsed Transactions on Scalable Information Systems **7**(24), 1–10 (2019)

[23] Amer, D.A., Attiya, G., Zeidan, I., Nasr, A.A.: Elite learning harris hawks optimizer for multi-objective task scheduling in cloud computing. The Journal of Supercomputing, 1–26 (2022)

[24] Malti, A.N., Benmammar, B., Hakem, M.: Task scheduling in cloud computing based on fpa metaheuristic algorithm. In: 2022 19th International Multi-Conference on Systems, Signals & Devices (SSD), pp. 41–46 (2022). IEEE

[25] Mathew, T., Sekaran, K.C., Jose, J.: Study and analysis of various task scheduling algorithms in the cloud computing environment. In: 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 658–664 (2014). IEEE

[26] Hussain, A., Aleem, M., Iqbal, M.A., Islam, M.A.: Investigation of cloud scheduling algorithms for resource utilization using cloudsim. Computing & Informatics **38**(3), 525–554 (2019)

[27] Zhong, Z., Chen, K., Zhai, X., Zhou, S.: Virtual machine-based task scheduling algorithm in a cloud computing environment. Tsinghua Science and Technology **21**(6), 660–667 (2016)

[28] Zhu, X., Chen, C., Yang, L.T., Xiang, Y.: Angel: Agent-based scheduling for real-time tasks in virtualized clouds. IEEE Transactions on Computers **64**(12), 3389–3403 (2015)

[29] Yang, X.-S.: Flower pollination algorithm for global optimization. In: International Conference on Unconventional Computing and Natural Computation, pp. 240–249 (2012). Springer

[30] Pavlyukevich, I.: Lévy flights, non-local search and simulated annealing. Journal of Computational Physics **226**(2), 1830–1844 (2007)

[31] Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Advances in engineering software **69**, 46–61 (2014)

[32] Meidani, K., Hemmasian, A., Mirjalili, S., Barati Farimani, A.: Adaptive grey wolf optimizer. Neural Computing and Applications **34**(10), 7711–7731 (2022)

[33] Aktan, M.N., Bulut, H.: Metaheuristic task scheduling algorithms for cloud computing environments. Concurrency and Computation: Practice and Experience **34**(9), 6513 (2022)

[34] Chakravarthi, K.K., Neelakantan, P., Shyamala, L., Vaidehi, V.: Reliable budget aware workflow scheduling strategy on multi-cloud environment. Cluster Computing **25**(2), 1189–1205 (2022)

[35] Chen, X., Du, W., Qian, F.: Multi-objective differential evolution with ranking-based mutation operator and its application in chemical process optimization. Chemometrics and Intelligent Laboratory Systems **136**, 85–96 (2014)

[36] Özkış, A., Babalık, A.: A novel metaheuristic for multi-objective optimization problems: The multi-objective vortex search algorithm. Information Sciences **402**, 124–148 (2017)

[37] Hussain, A., Aleem, M.: Gocj: Google cloud jobs dataset for distributed and cloud computing infrastructures. Data **3**(4), 38 (2018)

[38] The HPC2N Seth log. https://www.cs.huji.ac.il/labs/parallel/workload/l_hpc2n/index.html

[39] Behzad, S., Fotohi, R., Effatparvar, M.: Queue based job scheduling algorithm for cloud computing. International Research Journal of Applied and Basic Sciences ISSN **37853790** (2013)

[40] Mehdi, N.A., Mamat, A., Ibrahim, H., Subramaniam, S.K.: Impatient task mapping in elastic cloud using genetic algorithm. Journal of Computer Science **7**(6), 877–883 (2011)

[41] Attiya, I., Abd Elaziz, M., Xiong, S.: Job scheduling in cloud computing using a modified harris hawks optimization and simulated annealing algorithm. Computational intelligence and neuroscience **2020** (2020)

[42] GoCJ: Google Cloud Jobs Dataset. https://data.mendeley.com/datasets/

b7bp6xhrcd/1

[43] Gaidhane, P.J., Nigam, M.J.: A hybrid grey wolf optimizer and artificial bee colony algorithm for enhancing the performance of complex systems. Journal of computational science **27**, 284–302 (2018)

[44] CloudSim: A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services. http://www.cloudbus.org/cloudsim

[45] Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and experience **41**(1), 23–50 (2011)