# A Comparative Study of Metaheuristics Based Task Scheduling in Cloud Computing

Arslan Nedhir Malti, Badr Benmammar, Mourad Hakem

▶ **To cite this version:**

Arslan Nedhir Malti, Badr Benmammar, Mourad Hakem. A Comparative Study of Metaheuristics Based Task Scheduling in Cloud Computing. 7th International Symposium on Modelling and Implementation of Complex Systems (MISC 2022), Oct 2022, Mostaganem, Algeria. pp.263-278, 10.1007/978-3-031-18516-8_19 . hal-03932829

**HAL Id: hal-03932829**
**https://hal.science/hal-03932829**

Submitted on 10 Jan 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# A Comparative Study of Metaheuristics Based Task Scheduling in Cloud Computing

Arslan Nedhir Malti[1][0000−0002−2530−2158], Badr Benmammar[1][0000−0002−3930−2095], and Mourad Hakem[2]

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

**Abstract.** Cloud computing is a standard way of hosting software applications and services that is booming day by day thanks to the different utilities offered to users according to their needs and contracts. Most of these services are in the form of tasks and their execution in such environments requires efficient scheduling strategies that take into account both algorithmic and architectural features. The objective is to orchestrate the suitable assignment of the submitted tasks to the available resources on the basis of various functional requirements of end users. To overcome the scheduling issue, which is an NP-hard problem, various metaheuristic algorithms are used in literature to achieve near optimal solution. For this purpose, this paper aims to perform a comparative investigation of three common metaheuristic algorithms in the optimization process such as Shuffled Frog Leaping Algorithm (SFLA), Flower Pollination Algorithm (FPA) and Gray Wolf Optimization (GWO). Both standard and synthetic workloads are employed to analyze the performance of these algorithms by evaluating its objective function in term of two metrics which are makespan and resource utilization rate. The simulation results obtained using the CloudSim framework are very satisfactory and clearly show the value of our study.

**Keywords:** Cloud computing · Task scheduling · Qos · Optimization · Metaheuristique.

## 1 Introduction

Cloud computing is a promising technology that makes it easier to run scientific and commercial applications by giving users transparent, on-demand network access to a shared set of computer and storage resources. It is a simple consumer-provider service model that allows computer resources to be delivered as flexible, scalable resources and rented on an elastic pay-per-use model without any geographical restrictions.

Generally, cloud computing can provide three types of services: SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a

Service). These are offered with different levels of Quality of Service (QoS) to meet the needs of different users. Although many cloud computing services have similar functionality (e.g. compute services, storage services, network services, etc.), they differ from each other in their non-functional QoS, such as execution time, cost, energy consumption, utilization resource, service availability, and so on. These QoS parameters can be defined and offered by different SLAs (Service Level Agreements) contracts. which is a sort of negotiation that specifies the resource requirements, minimum expectations and obligations that exist between cloud user and Cloud Service Providers (CSPs) in order to meet the various user requests and improve the overall cloud system performance.

Despite the fact that concept of cloud computing is widely used today, due to the various benefits and services offered to the end-users according to their needs and contracts, there are several issues that must be handled by developing new technical solutions to address new challenges and constraints. One of the important research issues is task scheduling, which reflects the process of assigning user submitted tasks to the available resources on the basis of various functional requirements of end users while respecting the specific constraints of the IaaS environment. In this context, scheduling tasks becomes a multi-objective optimization issue and a successful scheduler must strike an efficient trade-off solutions that satisfy all objectives.

On the other hand, task scheduling is considered as a combinatorial optimization problem in which standard algorithms fail to identify the global optimal solution. Unfortunately, this property makes it as an NP-complete problem [14,12,15]. Therefore, to achieve efficient trade-off solutions that satisfy all objectives and better system performance, an innovative scheduling strategies should be performed and implemented.

Even though several works have been proposed by different researchers to address the scheduling problem, most of them have mainly focused on metaheuristics approaches such as Evolutionary Algorithms (EAs), Human-based, Physics-based or Swarm Intelligence (SI) algorithms [14]. Regardless of the variety of these algorithms, there is a common feature exploration (diversification) and exploitation (intensification). A well-organized optimizer should be able to find an acceptable balance between the exploration and exploitation. Otherwise, the possibility of being trapped in local optima and immature convergence drawbacks increases. For this purpose, the aim of this paper is to conduct a comparison of three swarm intelligentce algorithms. Considered as one of the best metaheuristic algorithms that have attracted the interest of researchers in various types of complex problems such as in the field of engineering, geology, industries, medical sector, and so on [10,24,3,11]. In this study, we are interested in scheduling the upcoming load, applications, or tasks to cloud resources in such a way that the client may complete their task in least time while balancing the load on the virtual machines and ensuring that their release time is homogeneous. Concisely, the main focal points of the paper are as follows:

1. Adaptation of three recent metaheuristics for independent task scheduling in a heterogeneous cloud computing environment, namely FPA, SFLA and GWO.

2. A comparison investigation of the three metaheuristics in a bi-objective framework, aims to orchestrate the trade-offs relationship between the makespan and resource utilization rate.

The rest of this paper is organized as follows: Section 2 offers a review of the relevant approaches that have been proposed in the literature. In section 3, the detailed description of the metaheuristic algorithms are discussed. Section 4 describes our approach based on the adaptation of metaheuristics according to criteria that we have defined for this purpose. Section 5 focuses on the simulation setup and the experimental results. Finally, some concluding remarks are made in Section 5.

## 2 Related works

In this section, some recent works on scheduling problem for cloud environments are highlighted. Karpagam et al. [16] have proposed a vertical node partitioning approach based on a heuristic and novel SFLA clustering algorithm for scheduling scientific workflows. SFLA with clustering and without clustering have been explored. The proposed technique with clustering achieves a higher optimization ratio on makespan and resource utilization compared to Opportunistic Load Balancing (OLB) and SFLA without clustering. However, the proposed work has some remaining gaps in trapping, local diversity, premature convergence and high computational cost. In the same cloud IaaS and task workflow modeling, Kaur and Mehta [18] have introduced a novel optimization techniques called Augmented Shuffled Frog Leaping Algorithm (ASFLA). The proposed approach is an improvement of the basic SFLA algorithm that aims to optimize the running cost of the application while meeting the deadline constraint. It was shown that the proposed technique is able to reduce the overall execution cost compared to those of SFLA and Particle Swarm Optimization (PSO) algorithm. However, this reduction leads to an increase in execution time. As a result, the authors' work assumed that ASFLA is more appropriate when minimizing the execution cost is the main concern.

In the work of Durgadevi and Srinivasan [8], a hybrid optimization algorithm which combines SFLA and Cuckoo Search (CS) optimization algorithm is designed. The proposed algorithm is formulated to diminish the knapsack issue for the server side resource allocation mechanism in cloud computing environment. Furthermore, it addresses the limitations of previous works such as the HABCCS algorithm, the GTS task algorithm, and the krill herd algorithm in terms of high execution time, throughput, and delay, which can result in a significant degradation in overall system performance. Kaur and Sidhu [17] have developed TSFPA, an independent Task Scheduling method based on the Flower Pollination Algorithm. The authors' goal is to solve the above problem by assigning tasks to the virtual instance types in a heterogeneous cloud environment so that makespan could be minimized. The effectiveness of the TSFPA was compared with three well-known existing algorithms, namely Genetic Algorithm (GA), First Come

First Serve (FCFS) and Round Robin (RR) approach. The simulation results performed with CloudSim toolkit reveal that the author's proposal provides better results in terms of makespan.

Recently, Bezdan et al. [7] have proposed an improved 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 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. A bi-objective optimization for independent task scheduling on cloud resources is presented by Gupta et al. [13]. It is based on the FPA optimization algorithm, and compared to three other metaheuristic approaches: Gravitational Search Algorithm (GSA), GA and PSO. It uses an efficient pollen representation scheme and a dedicated process to determine the task-VM mapping from a given pollen so that the average cloud resource utilization and makespan are optimized. The simulation experiments reveal that task scheduling based on FPA is better than the compared concurrent metaheuristics approaches. However, the suggested work lacks in terms of dynamicity as it deals only with static independent task scheduling and virtual machines.

A hybrid bi-objective scheduling algorithm based on bio-inspired and swarm intelligence algorithms for scheduling scientific workflows is proposed by Khurana and Singh [19]. The GWO and FPA was used with the PEFT algorithm for global and local optimization. It seeks to reduce both monetary cost and execution time value to generate task-VM mapping in a cloud environment. The performance of the proposed algorithm is validated through numerical simulations with flower pollination and genetic algorithm. Alzaqebah et al. [4] employed the GWO method to solve task scheduling problems by modifying the fitness function to handle multi-objectives in a single fitness function; the makespan and cost are the objectives included in the fitness. This method's main purpose is to decrease both cost and makespan. The simulation results performed with CloudSim tool illustrated that the given strategies have good effects on the performance compared to traditional GWO and Whale Optimization Algorithm (WOA).

A mean grey wolf optimization variant algorithm is developed by Natesan and Chokkalingam [22], with the aim of increasing the accuracy and performance of the standard GWO algorithm. Following the comparison performed between the authors' work and other concurrent techniques such as existing PSO and standard GWO using the CloudSim toolbox for two datasets left-skewed and right-skewed showed that the introduced technique offers better performance in terms of makespan and energy consumption. This is due to the encirclement and hunting equations that have been improved. In another study, Abed-Alguni and Alawad [2] presented a discrete variant of the Distributed Grey Wolf Optimizer

(DGWO) for scheduling scientific workflows applications in cloud environments. The DGWO is a parallelized variant of GWO algorithm, which is modeled as a minimization problem with two objectives: computation and data transmission costs. The proposed algorithm was experimentally tested and compared to two well-known optimization-based scheduling algorithms PSO and GWO using two types of workflows: balanced and imbalanced workflows, with the results obtained using WorkflowSim demonstrating that the authors' proposal has a relative advantage over the compared algorithms.

## 3 Presentation of the SFLA, FPA and GWO algorithms

This section consists of a brief introduction about the shuffled frog leaping algorithm, flowers pollination algorithm and gray wolf optimizer.

### 3.1 Shuffled frog leaping algorithm

The SFLA is a population-based, cooperative search algorithm created by Eusuff and Lansey [9]. The algorithm is inspired by frogs' social behaviour. Frogs' natural habitats are wetlands, and they usually live in groups. The frogs' main purpose is to investigate the searching area in order to reach the most foods with the least attempts. In order to accomplish this, the frogs in their wetlands are partitioned into different groups (memeplexes). Each frog has its own set of data, referred to as a meme. In each memeplex, there is a local search strategy that seeks to update the positions of the worst frogs to a better position by using the location of the local best or global best frog [9,6].

The SFLA's phases begin with initialization of parameters. The fitness value of each frog is then calculated after the initial population of frogs is generated randomly in decision space. Following that, the population is sorted on the basis of fitness values provided that the best member in the first index. As a next step, the sorted population is separated into memeplexes by using Eq. 1 and each memeplex undergoes a memetic evolutionary process, i.e. during the local exploration, the worst frog $X_w$ jumps to the best solution $X_b$ as formulated in Eq. 2

$$Y_k = [(X_i)_k | X_i = X(k + m * (i - 1)), i = 1, \ldots, n], k = 1, \ldots, m \tag{1}$$

where $m$ denotes the number of memeplexes and $n$ the number of members in each memeplex. It is assured that members are distributed evenly among the memeplexes. Suppose that m = 3; the first member goes to first memeplex, the second member to second memeplex, the thrid member to thrid memeplex, the fourth member to first memeplex and so on.

$$\begin{aligned} S &= r * |X_b - X_w| \\ X_{w'} &= X_w + S \end{aligned} \tag{2}$$

The memeplexes are gathered and shuffled after the memetic evolutionary process. Unless the termination requirement is met, the subsequent iteration begins with population re-sorting. The SFLA steps are shown in Fig. 1.

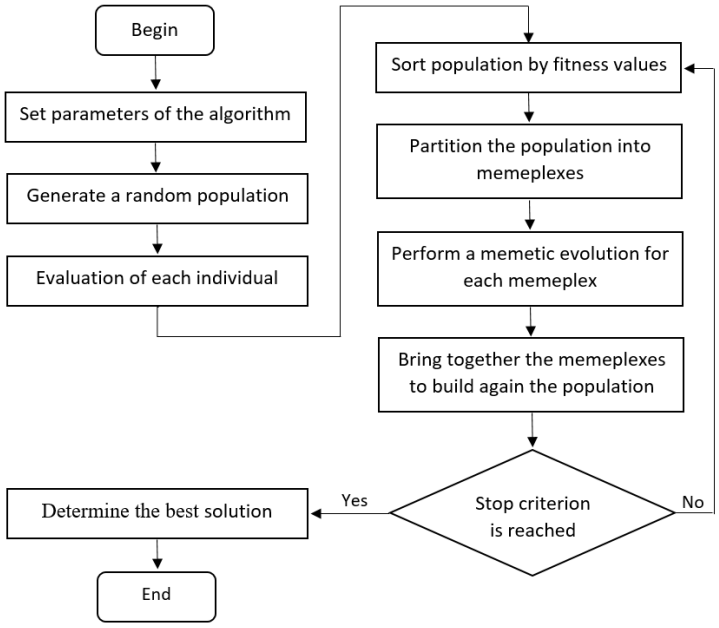


**Fig. 1.** The flowchart of the SFLA.

### 3.2 Flower pollination algorithm

Xin-She Yang in 2012 [25] proposed a natural bio-inspired flower pollination algorithm drawing its metaphor from the pollination process of flowering plants. Pollination can be achieved by self-pollination or cross-pollination. The first type of pollination, also known as local pollination, occurs when pollen from one flower pollinates the same flower or other flowers of the same plant with the help of environmental factors [25]. While cross-pollination, also known as global pollination, occurs over long distances when pollen is delivered to a flower from another plant through direct or indirect intervention by pollinators following Levy flight behavior [23].

In FPA and during the optimization process, the exploration of the search space is done by biotic and cross-pollination where the movement of the pollen is represented by the Levy flight. The latter is a random walk based on a random step from the Lévy distribution, causing a much longer movement from its current position. Therefore, we can idealize the characteristics of the pollination

process, flower constancy and pollinator behavior based on four main rules listed as follows:

- **R1:** Biotic, cross-pollination acting as a global pollination process via the Levy flight.
- **R2:** Abiotic and self-pollination are considered local pollination.
- **R3:** Consistency of flowers may be involved due to the similarity of two flowers.
- **R4:** Local pollination and global pollination are controlled by a switching probability $p \in [0, 1]$.

Based on the above four rules, the basic steps of the FPA metaheuristic are presented in Fig. 2.

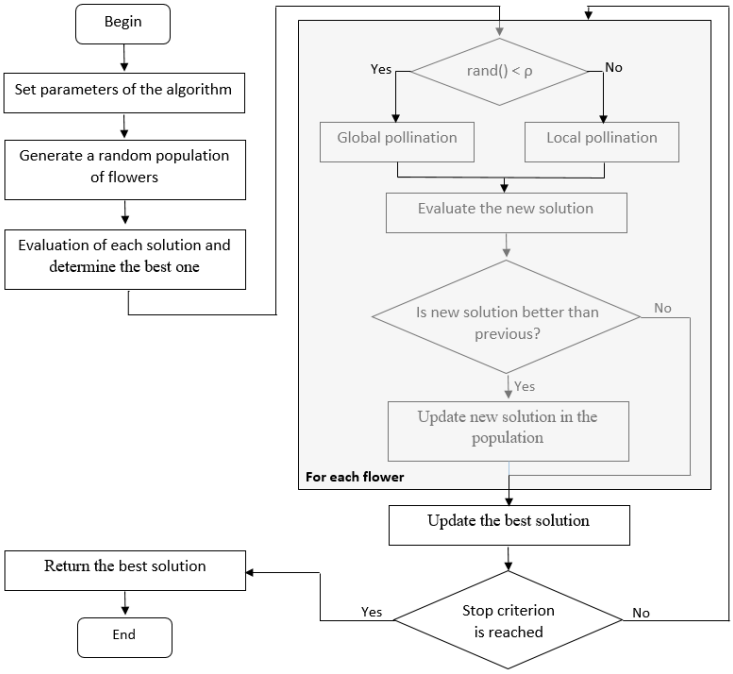


**Fig. 2.** The flowchart of the FPA.

### 3.3 Gray wolf optimizer algorithm

Grey wolf optimization is a recently proposed nature-inspired metaheuristic optimization algorithm [21]. It is a population-based and iterative algorithm that simulates the leadership hierarchy and hunting process of grey wolves in social life. The GWO algorithm's mathematical modeling is split into four phases: social hierarchy, encircling prey, hunting, and attacking prey [20].

**Social hierarchy:** the GWO algorithm assigns wolves four distinct roles: alpha ($\alpha$), beta ($\beta$), delta ($\delta$), and omega ($\omega$), with each wolf having only one of these roles. Surprisingly, they have a highly structured social dominance hierarchy. The best three solutions in the population are identified as $\alpha$, $\beta$, and $\delta$ wolves. The rest of the population is thought to be omega.

The leader (alpha) mostly controls the rest of the population and makes social decisions such as when to wake up, where to relax, and so on. Beta assists the leader (alpha) in making decisions by acting as a consultant. Delta wolves are subordinates who submits to the upper levels (alpha and beta) while ruling the lower levels (omega).

**Encircling prey:** during the hunt, gray wolves encircle their prey. This encircling behavior is mathematically modeled in the following equations.

$$\begin{aligned} \vec{X}(t+1) &= \vec{X_p}(t) - \vec{A}.\vec{D} \\ \vec{D} &= |\vec{C}.\vec{X_p}(t) - \vec{X}(t)| \end{aligned} \tag{3}$$

where $\vec{X}$ represents the gray wolf's current position and $\vec{X_p}$ indicate the prey's position, while $\vec{A}$ and $\vec{C}$ denote the coefficient vectors, which are computed as follows:

$$\vec{A} = 2.\vec{a}.\vec{r_1} - \vec{a} \tag{4}$$

$$\vec{C} = 2.\vec{r_2} \tag{5}$$

where $\vec{a}$ is a real value which is linearly reduced from 2 to 0 over iterations and $\vec{r_1}$, $\vec{r_2}$ are real number arrays randomly generated in the range $[0, 1]$.

**Hunting:** as previously stated, the leader wolves have the best places in the population and the hunt is mostly guided by them. The first three best agents are stored, and an estimated prey position is generated by $\alpha$, $\beta$, and $\delta$ to provide a reference position for the other wolves to update their positions randomly, as shown in the equations below.

$$\vec{D_\alpha} = |\vec{C_1}.\vec{X_\alpha} - \vec{X}|, \;\; \vec{D_\beta} = |\vec{C_2}.\vec{X_\beta} - \vec{X}|, \;\; \vec{D_\delta} = |\vec{C_3}.\vec{X_\delta} - \vec{X}| \tag{6}$$

$$\vec{X_1} = \vec{X_\alpha} - \vec{A_1}.\vec{D_\alpha}, \;\; \vec{X_2} = \vec{X_\beta} - \vec{A_2}.\vec{D_\beta}, \;\; \vec{X_3} = \vec{X_\delta} - \vec{A_3}.\vec{D_\delta} \tag{7}$$

$$\vec{X}(t+1) = \frac{\vec{X_1} + \vec{X_2} + \vec{X_3}}{3} \tag{8}$$

**Attacking:** when the prey stops moving, the grey wolves finish the hunt by tracking it. More specifically, wolves' capacity can lead to global optima. Since the value of $\vec{A}$ is significant; in case $|A| < 1$, the grey wolves are forced to attack the prey (exploitation). In the event that $|A| > 1$, the grey wolves will be forced to diverge from the prey to find more suitable prey (exploration). The flow diagram of the GWO algorithm is outlined in Fig.3.

To apply GWO in task scheduling problems, the position of gray wolves updated with respect to their prey. In other words, the mapping of the tasks $T_i$ onto the resources $VM_j$ is processed by updating their positions according to the three best schedules found in the current population: alpha, beta and delta.

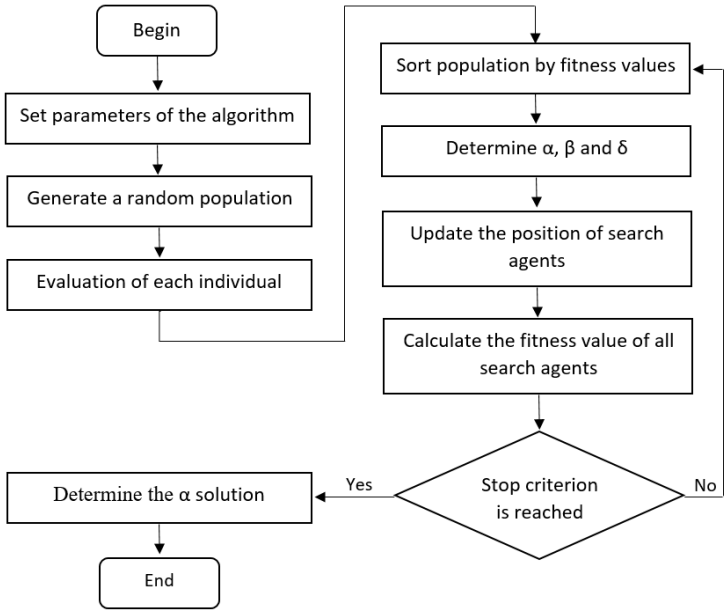


**Fig. 3.** The flowchart of the GWO.

## 4 Adaptation of metaheuristiques

To adapt the three previously mentioned metaheuristic algorithms to our problem, we need to define (1) an appropriate solution structure. (2) a fitness function model that simplifies the problem as much as feasible while remaining sufficiently generic to be consistent with a large number of QoS metrics.

### 4.1 A solution's representation

A valid solution must satisfy the following conditions: (1) each task must be executed on a single selected machine, and each machine can execute several tasks. (2) for each task on each assigned resource, some parameters must be pre-computed. (3) any interruption is ignored once the task has been processed.

Our solution code is as follows: each task $T_i$ is assigned to a virtual machine $VM_j$ in the IaaS cloud. The size of the solution is determined by the number of tasks that make up the workload. A simplified representation of a solution is given in Fig.4. This literally translates as: task $T_1$ is assigned to virtual machine 4. Task $T_2$ is assigned to machine 1, and so on.

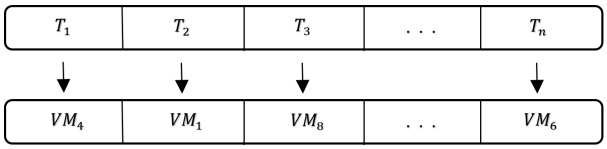


**Fig. 4.** A simplified representation of a solution.

### 4.2 Fitness function

We analyze two conflicting objectives in order to derive an efficient multi-objective fitness function for task scheduling problem in cloud computing. The first objective is to reduce the makespan, while the second objective is to increase resources utilization rate.
**Makespan:** it represents the difference between the submission time of the first task and the reception time of the results (the finish time of the last executed task). It is defined as follows:

$$Makespan = max_{j \in TSK}\left(FinishTime_j\right) - min_{i \in TSK}\left(SubmissionTime_i\right) \tag{9}$$

where $TSK$ is the set of tasks submitted to the scheduler for execution.
**Resource utilization rate:** is the way toward redistributing the all-out workload into individual ressource to guarantee that no ressource is overloaded and no ressources were under loaded or inactive. It is formulated as follows [13]:

$$Resource\ utilization\ rate = \frac{Average\ makespan\ of\ all\ VMs}{Makespan} \tag{10}$$

In this section, we introduce a generic fitness function model that can be easily adapted to optimize any type of digital goal. In this context, we use the weighted sum method to merge both above mentioned criteria into a single-objective for a minimization problem as defined by Eq.11.

$$F = w_1 * Makespan + w_2 * \frac{1}{Resource\ utilization\ rate}\ , \quad w_1 + w_2 = 1 \tag{11}$$

where $w_1$ and $w_2$ are the weights values that reflects the user's importance or requirement for each criterion. In this study, the weight values are 0.8 and 0.2 corresponding respectively to the criteria makespan and resource utilization rate. We have privileged the makespan metric due of it importance that greatly affects the performance of the cloud computing system. It should be noted that, the both makespan value and resource utilization rate of each solution are normalized to a same scale within an interval $[0, 1]$.

## 5 Simulation results and analysis

### 5.1 Experimental environment and datasets

To evaluate the performance of the selected metaheuristic methods, serie of experiments were conducted on the CloudSim simulator [1]. All the experiments

are performed in a heterogeneous environment as listed in Table 1 [5], defining the configuration details for the employed simulation environment.

**Table 1.** CloudSim simulation parameters.

| Cloud entity | Parameter | Value |
|---|---|---|
| Data-center | No. of data-centers | 5 |
| | No. of hosts | 2 |
| Host | PES | 8 (Octa core) |
| | MIPS | 6 000 |
| | RAM | 16 GB |
| | Storage | 1 TB |
| | Bandwidth | 10 GB/s |
| VM | No. of VMs | 50 |
| | MIPS | 100 to 5 000 |
| | RAM | 0.5 GB |
| | Storage | 10 GB |
| | Bandwidth | 1 GB/s |
| | Policy type | Time shared |
| Cloudlets | No. of cloudlets | 100 - 200 - 300 - 400 - 500 - 600 |

For experiments, both synthetic and standard workload traces are utilized. The synthetic workload is generated using a uniform distribution, which presents an equal amount of small, medium, and large-sized tasks. We took into account that each submitted task may require a different amount of processing time, which is measured in Million Instructions Per Second (MIPS). Where task sizes are randomly formed in range of 1 000 to 20 000, similarly like in [5].

In addition to the synthetic workload, HPC2N (High Performance Computing Center North) parallel workloads are used for performance evaluation. The HPC2N set log is one of the most commonly benchmarks for evaluating distributed system performance. For the purpose of comparison, each experiment is running 10 times and takes the averge results. The specific parameter settings of the selected metaheuristics are presented in Table 2. These parameters were tuned and selected experimentally.

**Table 2.** Parameter settings of the algorithms.

| Algorithm | Parameter | Value |
|---|---|---|
| SFLA | Population size | 100 |
| | Number of memeplexes | 10 |
| FPA | Population size | 100 |
| | $\rho$ | 0.8 |
| | $\lambda$ | 1.5 |
| GWO | Population size | 100 |
| | $a$ | $2 \rightarrow 0$ |

### 5.2 Result analysis and discussion

To display the performance of SFLA, FPA and GWO metaheuristic algorithms, we plotted graphs of solution's quality (i.e., makespan and resource utilization rate) versus the number of tasks for two datasets.

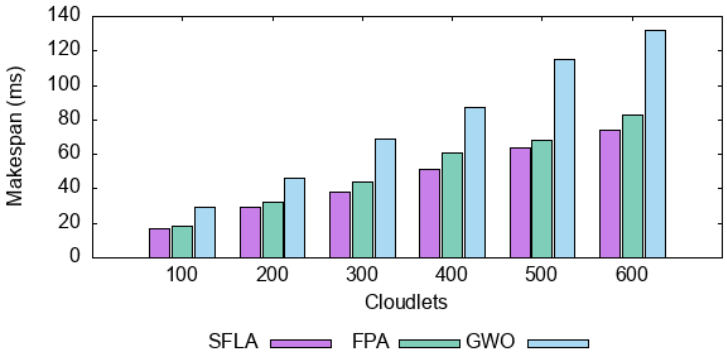


**Fig. 5.** Comparison in terms of makespan for synthetic workload.

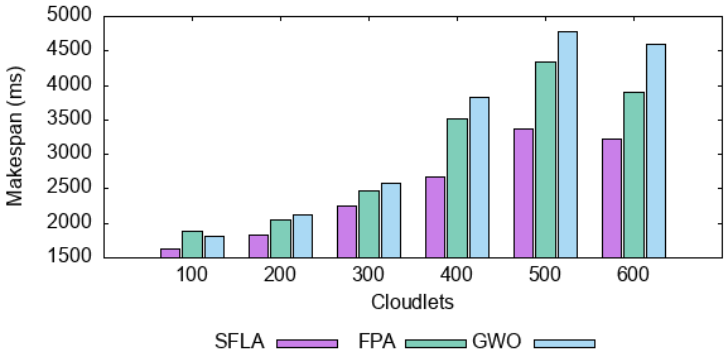


**Fig. 6.** Comparison in terms of makespan for real workload HPC2N.

First, we evaluate the makespan of the compared metaheuristics. As shown in Figs. 5 and 6, which exhibit the values for the synthetic workload and the HPC2N workload respectively, we can observe that the makespan objective increases as the number of tasks increases. This is not particularly surprising in light of 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 SFLA algorithm has higher performance than other evaluated scheduling algorithms, namely FPA and GWO. This means that the SFLA takes less time to execute the submitted tasks and outperforms all the other compared algorithms in all the test cases.

This can be explained by the fact that the SFLA draws its formulation from two other search techniques: the local search of the particle swarm optimization technique; and the competitiveness mixing of information of the shuffled complex evolution technique.

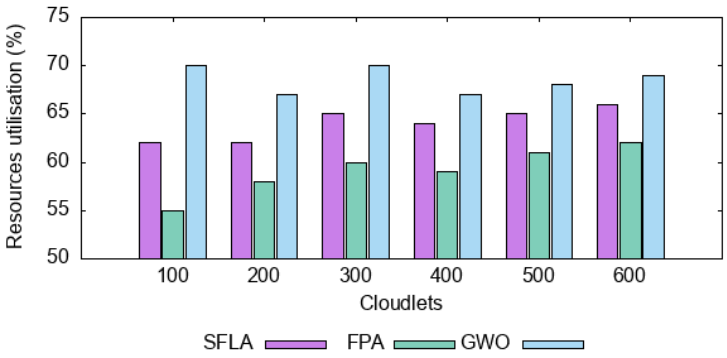


**Fig. 7.** Comparison in terms of resource utilization rate for synthetic workload.

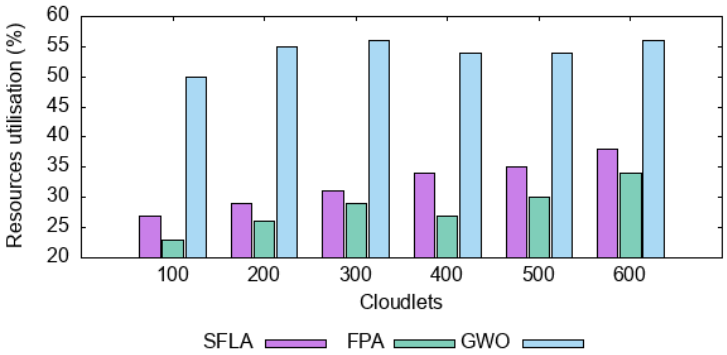


**Fig. 8.** Comparison in terms of resource utilization rate for real workload HPC2N.

Comparing the results in terms of resource utilization rate, we can observe from Figs. 7 and 8 which reflect the ratio obtained for simulated synthetic and the HPC2N workload respectively, that the GWO algorithm produce better performance rate over the SFLA and FPA. This is due to the fact that the GWO process is based on the three best scheduling in the population which leads to an efficient use of resources during the scheduling process.

A further comparison was made to measure the convergence rate of the selected metaheuristics in order to validate the performances obtained. From the results, it showed that SFLA converges faster than other algorithms. More specif-

ically, it appeared that FPA performs a little better than GWO in most of the cases; both of them fall behind the SFLA algorithm.

## 6 Conclusion

In this paper, we have addressed the problem of task scheduling in cloud computing environments. An effective scheduling strategy must not only satisfy the user's QoS requirements, but also should simultaneously improve the service provider's parameters. For this purpose, we have critically examined three widely used metaheuristics in the optimization process by evaluating their objective function in term of two contradictory QoS which are makespan and resource utilization rate. Based on CloudSim framework, it was shown that, when dealing with makespan optimization objective, the application of the SFLA algorithm generates better quality solution and converge faster than other compared metaheuristics algorithms across all the workload instances. Whereas, in term of resource utilization the obtained results are more interesting for GWO by comparing them with the SFLA and FPA algorithms. However, GWO performance is stuck in local optima. To fill this gap, a strategy is required to keep from falling into local optima while maintaining a high convergence rate and QoS awareness. The suggested approach is to develop a hybrid of meta-heuristic approaches with multi-objectives to take advantage of each algorithm.

## References

1. Cloudsim: A framework for modeling and simulation of cloud computing infrastructures and services. `http://www.cloudbus.org/cloudsim/`
2. Abed-Alguni, B.H., Alawad, N.A.: Distributed grey wolf optimizer for scheduling of workflow applications in cloud environments. Appl. Soft Comput. **102**, 107113 (2021)
3. Alyasseri, Z.A.A., Khader, A.T., Al-Betar, M.A., Abasi, A.K., Makhadmeh, S.N.: Eeg signals denoising using optimal wavelet transform hybridized with efficient metaheuristic methods. IEEE Access **8**, 10584–10605 (2019)
4. Alzaqebah, A., Al-Sayyed, R., Masadeh, R.: Task scheduling based on modified grey wolf optimizer in cloud computing environment. In: 2019 2nd Int. Conf. New Trends Comput. Sci. (ICTCS). pp. 1–6. IEEE (2019)
5. Attiya, I., Abd Elaziz, M., Xiong, S.: Job scheduling in cloud computing using a modified harris hawks optimization and simulated annealing algorithm. Comput. Intell. Neurosci. **2020**, 1–17 (2020)
6. Benmammar, B.: Quality of service optimization in orthogonal frequency division multiplexing-based cognitive radio systems based on shuffled frog leaping algorithm. Concurr. Comput. Pract. Exp. **34**(1), e6530 (2022)
7. 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, pp. 163–171. Springer (2021)
8. Durgadevi, P., Srinivasan, S.: Resource allocation in cloud computing using sfla and cuckoo search hybridization. Int. J. Parallel Program. **48**(3), 549–565 (2020)

9. Eusuff, M.M., Lansey, K.E.: Optimization of water distribution network design using the shuffled frog leaping algorithm. J. Water Resour. Plann Manag **129**(3), 210–225 (2003)
10. Fahad, M., Aadil, F., Khan, S., Shah, P.A., Muhammad, K., Lloret, J., Wang, H., Lee, J.W., Mehmood, I., et al.: Grey wolf optimization based clustering algorithm for vehicular ad-hoc networks. Comput. Electr. Eng. **70**, 853–870 (2018)
11. Faris, H., Aljarah, I., Al-Betar, M.A., Mirjalili, S.: Grey wolf optimizer: a review of recent variants and applications. Neural Comput. Appl. **30**(2), 413–435 (2018)
12. Ghafari, R., Kabutarkhani, F.H., Mansouri, N.: Task scheduling algorithms for energy optimization in cloud environment: a comprehensive review. Clust. Comput. pp. 1–59 (2022)
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. Springer (2017)
14. 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 Evol. Comput. **62**, 100841 (2021)
15. Ibrahim, I.M., et al.: Task scheduling algorithms in cloud computing: A review. Turk. J. Comput. Math. Educ. **12**(4), 1041–1053 (2021)
16. Karpagam, M., Geetha, K., Rajan, C.: A modified shuffled frog leaping algorithm for scientific workflow scheduling using clustering techniques. Soft Comput. **24**(1), 637–646 (2020)
17. Kaur, J., Sidhu, B.K.: A new flower pollination based task scheduling algorithm in cloud environment. In: 2017 4th Int. Conf. Signal Process. Comp. Cont. (ISPCC). pp. 457–462. IEEE (2017)
18. Kaur, P., Mehta, S.: Resource provisioning and work flow scheduling in clouds using augmented shuffled frog leaping algorithm. J. Parallel Distrib. Comput. **101**, 41–50 (2017)
19. 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) (2020)
20. Meidani, K., Hemmasian, A., Mirjalili, S., Barati Farimani, A.: Adaptive grey wolf optimizer. Neural Comput. Appl. pp. 1–21 (2022)
21. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014)
22. Natesan, G., Chokkalingam, A.: Task scheduling in heterogeneous cloud environment using mean grey wolf optimization algorithm. ICT Express **5**(2), 110–114 (2019)
23. Pavlyukevich, I.: Lévy flights, non-local search and simulated annealing. J. Comput. Phys. **226**(2), 1830–1844 (2007)
24. Ulusoy, S., Nigdeli, S.M., Bekdaş, G.: Novel metaheuristic-based tuning of pid controllers for seismic structures and verification of robustness. J. Build. Eng. **33**, 101647 (2021)
25. Yang, X.S.: Flower pollination algorithm for global optimization. In: International conference on unconventional computing and natural computation. pp. 240–249. Springer (2012)