Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 5205530, 9 pages
https://doi.org/10.1155/2021/5205530

Research Article

IFFO: An Improved Fruit Fly Optimization Algorithm for Multiple Workflow Scheduling Minimizing Cost and Makespan in Cloud Computing Environments

Ambika Aggarwal,[1] **Priti Dimri**,[2] **Amit Agarwal**,[3] **Madhushi Verma**,[4] **Hesham A. Alhumyani**,[5] **and Mehedi Masud**[6]

[1]*School of Computer Science, University of Petroleum and Energy Studies, Dehradun, India*
[2]*Department of Computer Applications, Govind Ballabh Pant Engineering College, Pauri, India*
[3]*Dr. APJ Abdul Kalam Institute of Technology, Tanakpur, India*
[4]*Department of Computer Science Engineering, Bennett University, Uttar Pradesh, India*
[5]*Department of Computer Engineering, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia*
[6]*Department of Computer Science, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia*

Correspondence should be addressed to Mehedi Masud; mmasud@tu.edu.sa

Received 25 April 2021; Accepted 14 May 2021; Published 4 June 2021

Academic Editor: Vijay Kumar

Cloud computing platforms have been extensively using scientific workflows to execute large-scale applications. However, multiobjective workflow scheduling with scientific standards to optimize QoS parameters is a challenging task. Various metaheuristic scheduling techniques have been proposed to satisfy the QoS parameters like makespan, cost, and resource utilization. Still, traditional metaheuristic approaches are incompetent to maintain agreeable equilibrium between exploration and exploitation of the search space because of their limitations like getting trapped in local optimum value at later evolution stages and higher-dimensional nonlinear optimization problem. This paper proposes an improved Fruit Fly Optimization (IFFO) algorithm to minimize makespan and cost for scheduling multiple workflows in the cloud computing environment. The proposed algorithm is evaluated using CloudSim for scheduling multiple workflows. The comparative results depict that the proposed algorithm IFFO outperforms FFO, PSO, and GA.

1. Introduction

Cloud is an infinite pool of configurable computing resources (storage, network, processor, bandwidth, etc.) with some functionalities such as an on-demand pay-per-use model, high availability, scalability, and reliability [1, 2]. It also supports the distributed architecture for geographically distributed heterogeneous resources and provisioning them to clients through virtualization for hosting large-scale applications. These applications are deployed in the form of workflows which are further divided into smaller tasks.

Due to continuously increasing workloads and the rise in their difficulty levels, workflow scheduling has become a widely studied cloud computing problem that attracts many researchers. As depicted in Figure 1, workflow scheduling is used to allocate the required resources to the appropriate tasks to complete the execution process. During scheduling tasks on the virtual machine (VM), the client's QoS constraints must be fulfilled. Different clients may have different QoS requests in terms of cost, time, security, and so forth.

Multiple workflow scheduling comes into consideration to maximize the cloud architecture's throughput when

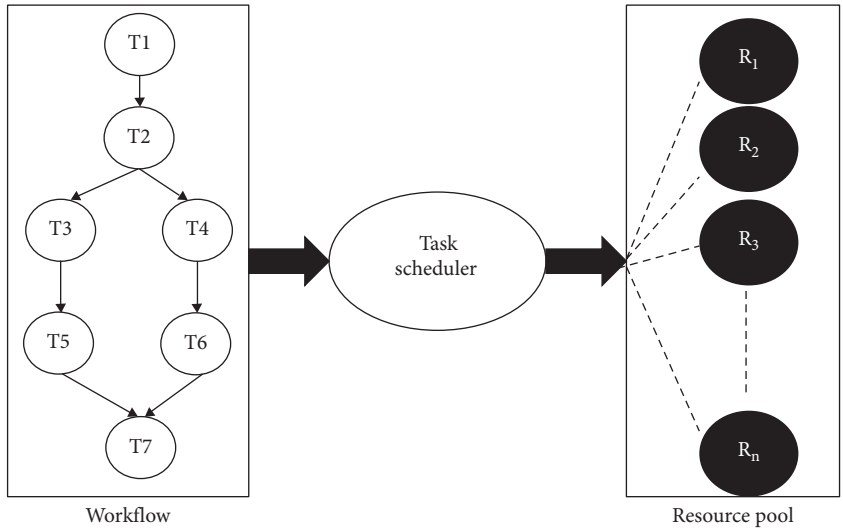

Figure 1: Basic concept of workflow scheduling.

various client requests are received simultaneously. Similar tasks can be identified to be allocated on a similar set of resources [3]. Strategies must be adopted to enhance the system's performance and ensure that all client requests are completed before the deadline.

An efficient scheduling technique maintains a trade-off between user requirements and resource utilization [4]. Maintaining this trade-off becomes challenging when some tasks have a parent-child relationship where a child task can only begin executing once its parent task has finished and all the output data from the predecessor task has been communicated to the child task [5, 6]. Various list-based algorithms cannot be directly implemented in cloud computing environments because of the resource heterogeneity, varied QoS constraints, and cloud's dynamic nature [2]. Several metaheuristic algorithms such as FFO [7], PSO [8], GA [9, 10], have been explored for solving workflow scheduling problems. However, optimizing multiple objectives is still a challenging task for the CCE [11–14]. Optimizing one QoS parameter often results in compromising with the other QoS parameter. Thus, scheduling multiple workflows on the cloud while maintaining a trade-off among multiple QoS parameters remains a problem that needs to be solved.

In this paper, an enhanced metaheuristic optimization technique IFFO has been proposed for scheduling multiple workflows on cloud computing environments to optimize multiple QoS parameters. The results of the proposed technique were generated using CloudSim and compared with existing FFO, PSO, and GA algorithms to validate the performance of IFFO in terms of makespan and cost parameters.

The rest of the paper is organized as follows: Section 1 describes the introductory concepts of cloud computing related to workflow scheduling. A crisp and concise literature survey on workflow scheduling for QoS parameters is discussed in Section 2. Section 3 highlights the problem formulation and problem definition. A novel framework using the IFFO algorithm is proposed in Section 4. The proposed algorithm's experimental results are given in Section 5, and the conclusion is mentioned in Section 6.

2. Background

Yassa et al. have presented a new multiobjective approach, called DVFS-MODPSO [15], for scheduling workflows on the cloud computing environment. The presented algorithm is a hybridization of PSO with Heterogeneous Earliest Finish Time (HEFT), aiming to optimize multiple objectives like makespan, cost, and energy consumption. Dynamic Voltage and Frequency Scaling (DVFS) is used for energy optimization, and the results show better Pareto optimal solutions than HEFT.

CGA^2 [16] is a technique proposed by Liu et al., which includes an adaptive penalty function for scheduling deadline constrained workflows in the cloud computing environment, addressing the limitations of previously proposed evolutionary algorithms. The proposed algorithm prevents premature convergence, unlike several existing static techniques, by applying adaptive crossover and mutation probabilities and generates solutions that are able to meet deadline constraints. CGA^2 is compared with traditional algorithms such as PSO, HEFT, GA, and Random to demonstrate better performance in terms of meeting deadlines under strict constraints and reducing the overall workflow execution cost.

HSGA [17] is a GA-based hybrid workflow scheduling technique adopted by Delavar et al., which utilizes the optimization characteristics of Round Robin (RR) and Best Fit (BF) scheduling algorithms. Initially, the proposed technique does the priority ranking of tasks based on their dependencies, and then the resource allocation is done by implementing RR and BF for appropriate VM selection. The experimental results depicted better performance of HSGA in terms of reducing makespan, lowering failure rate, and

balancing the load when compared with LAGA and NGA scheduling algorithms.

CDMWS [18] is a dynamic optimization technique for scheduling multiple workflows on the cloud, proposed by Delavar et al., which aims at improving CPU utilization, reducing makespan, and improving the makespan-deadline meeting ratio. The proposed technique is also divided into two stages. The first stage is responsible for estimating the execution time for each task by considering workflow deadline and task dependencies. The second stage is responsible for dynamic VM allocation, where VMs can reuse for tasks having similar requirements. VM reusability is implemented to lower power consumption and increase resource utilization. CDMWS is compared with two other algorithms, EWSA and RR, to verify its superiority.

Another list-based heuristic, MOWS [19], introduced by F. Abazari et al., adopts the greedy approach for prioritizing tasks and allocating appropriate resources to them. The proposed technique aims at improving the security of the overall cloud architecture while maintaining the execution time of workflows. The first phase of the algorithm designs the solution based on task prioritization and assesses security risk. The second part proposes an algorithm to deal with various security threats while scheduling a workflow on the cloud.

GA-based multiobjective workflow scheduling algorithm, MOGA [20], presented by Attiqa Rehman et al., aims at optimizing a diverse range of objectives, including makespan, budget, resource utilization, deadline, and energy efficiency. A gap search algorithm was also introduced in this work that finds gaps in the schedule generated for a particular workflow and fills them with independent tasks to maximize resource utilization. MOGA was compared with three GA-based algorithms and one PSO-based algorithm (MOPSO) to validate its superiority. Table 1 shows the summary of the related works.

3. Problem Formulation

Any large-scale application that needs to be deployed on a cloud platform is generally represented in the form of a workflow $W = (\mathrm{T}, \mathrm{E})$. A workflow can be pictorially represented using a directed acyclic graph (DAG) where $T = \{T_1, T_2, .., T_n\}$ denotes the set of tasks. The complete application is divided into several dependent and independent subtasks. The tasks in a workflow are represented at different levels having a parent-child relationship where a child task cannot begin execution until all its parent tasks have finished execution, and all output data has been transferred to the child task. An edge E_{ij} from T_i to T_j represents that T_i is the parent of T_j and $\exists$ a dependency between T_i and T_j.

Consider a sample workflow depicted in Figure 2. The entire application is divided into seven subtasks T_1, T_2, .., T_7 falling at five different levels, that is, Level 0 to Level 4. At level 2, Tasks T_3 and T_4 are independent of each other since they are at the same level, so they can be executed concurrently on different resources. However, their execution can only begin once T_2 at Level 1 has finished its execution and transferred all output data to T_3 and T_4. Similarly, the execution of T_2 depends on its predecessor task, T_1. Since T_1 has no predecessor, it will be the first task to be executed. Workflow execution can consider being completed once the last task in the DAG, T_7, has finished its execution and generated its output.

Resource heterogeneity will also be considered as provided by any IaaS cloud provider. Different types of VMs will be available based on different configurations. Any cloud service provider who joins the cloud marketplace provides a two-dimensional bid $B_{VMi} = (P_{VMi}, C_{VMi})$ where P_{VMi} represents the processing capacity of the VM measured in terms of MIPS and C_{VMi} is the cost of execution on that VM. The pricing model is based on the current Amazon EC2 standards, where a full cycle consists of 60 minutes, and one extra minute will count for one complete cycle. So, if a resource is consumed for 61 minutes, the user will be charged for two complete cycles, that is, 120 minutes.

Execution time ET_j^l of a task T_j on a resource VM_l, where $T_j = \{T_1, T_2, T_3, \ldots .., T_J\}\ \forall\ j \in \{1, 2, 3, \ldots, J\}$, is calculated using equation (1) by considering the size S_{T_j} of task T_j in terms of MIPS and the performance variation factor P_{var_l} of VM_l introduced while adjusting the processing capacity of a VM. Since a workflow involves task dependency, the data transfer time DT_{ab} from tasks T_j to T_k can be calculated using equation (2) where Dout_{T_j} is the amount of output data generated by T_j which is assumed to be known in advance for each task, and bw is the bandwidth between each VM. Also, the data transfer rate between two tasks scheduled on the same resource will be zero. Hence, the total processing time PT_j^l of each task T_j on a resource VM_1 is calculated using equation (3) where e is the number of edges connected to a parent task T_j and $Q_e = 0$ if T_j and T_k are scheduled on the same VM, else 1:

$$\mathrm{ET}_j^l = \frac{S_{T_j}}{\left(P_{\mathrm{VM}_l} * \left(1 - P_{\mathrm{var}_l}\right)\right)}, \tag{1}$$

$$\mathrm{DT}_{jk} = \frac{\mathrm{Dout}_{T_j}}{bw}, \tag{2}$$

$$\mathrm{PT}_j^l = \mathrm{ET}_j^l + \left(\sum_1^e \mathrm{DT}_{jk} * Q \right). \tag{3}$$

Similarly, multiple workflow scheduling problems can also be formulated. Consider the diagram shown in Figure 3. It consists of three small workflows, all of which can be combined to form a single large workflow by adding $Dummy_{start}$ and $Dummy_{end}$ as the starting and ending tasks, respectively. The execution time of both of these tasks will be zero as they are included only for merging the smaller workflows.

Optimal task scheduling and resource provisioning can be done based on various objectives. This work focuses on optimizing two scheduling objectives, that is, makespan and cost, by finding a schedule $S = (\mathrm{Res}, \mathrm{Map}, Z_{ct}, Z_{ms})$ for scheduling workflow on cloud computing environment, where $\mathrm{Res} = \{r_1, r_2, .., r_c\}$ is the set of available resources, Map depicts the task to resource mapping in the form

Table 1: The summary of the related works.

Proposed Algorithm	Objectives considered	Algorithm type	Workflow type	Scheduling type (*S*/*D*)	Tool used	Year
DVFS-MODPSO [15]	Makespan, cost, and energy	HEFT + particle swarm optimization	Simple and scientific	Static	CloudSim	2013
HSGA [17]	Makespan, failure rate, and load balancing	Genetic algorithm	Complex	Static + dynamic	Simulator	2013
CGA^2 [16]	Constrained deadlines and cost	Genetic algorithm	Scientific	—	CloudSim	2016
CDMWS [18]	CPU utilization, makespan, deadline	List-based	Scientific (multiworkflow)	Dynamic	Simulator	2017
MOGA [20]	Makespan, budget, deadline, energy efficiency, resource utilization	Genetic algorithm	Real-life	Static	Simulator	2018
MOWS [19]	Execution time and security	List-based greedy approach	Scientific (multiworkflow)	—	WorkflowSim	2019

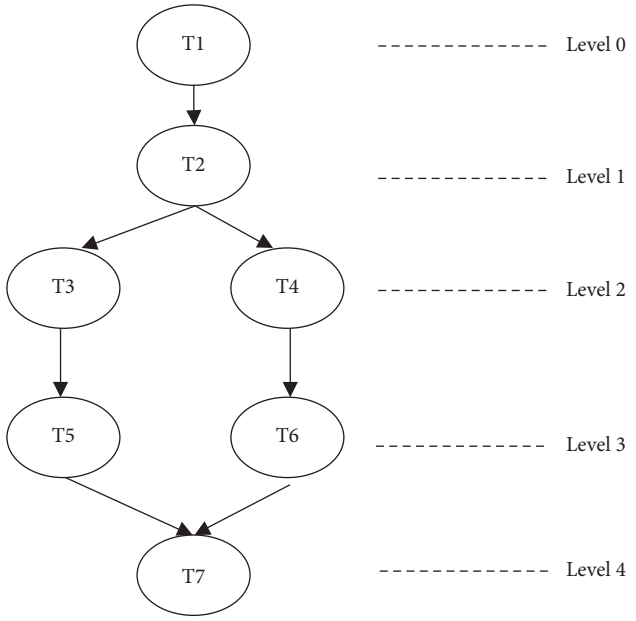

Figure 2: Sample workflow.

$M_{t_j}^{r_c} = (t_j, r_c, \mathrm{ST}_{t_j}, \mathrm{ET}_{t_j})$ for each task in the workflow which means that a task t_j is scheduled on resource r_c and it will begin execution at start time ST_{t_j} and will finish execution at the end time ET_{t_j}. Z_{ct} and Z_{ms} represent total execution cost and total execution time and can be calculated with the help of the following two equations, respectively:

$$Z_{ct} = \sum_{c=1}^{|R|} C_{\mathrm{VM}_{r_c}} * \left\lfloor \frac{\left(\mathrm{LET}_{r_c} - \mathrm{LST}_{r_c}\right)}{\alpha} \right\rfloor, \tag{4}$$

$$Z_{ms} = \max\left\{\mathrm{ET}_{t_j}\right\}, \tag{5}$$

where α denotes one cycle of the time unit for which the VM is charged, LET_{r_c} is the lease end time for resource r_c, and LST_{r_c} is the lease start time. A sample schedule generated for workflow shown in Figure 2 is depicted in Figure 4.

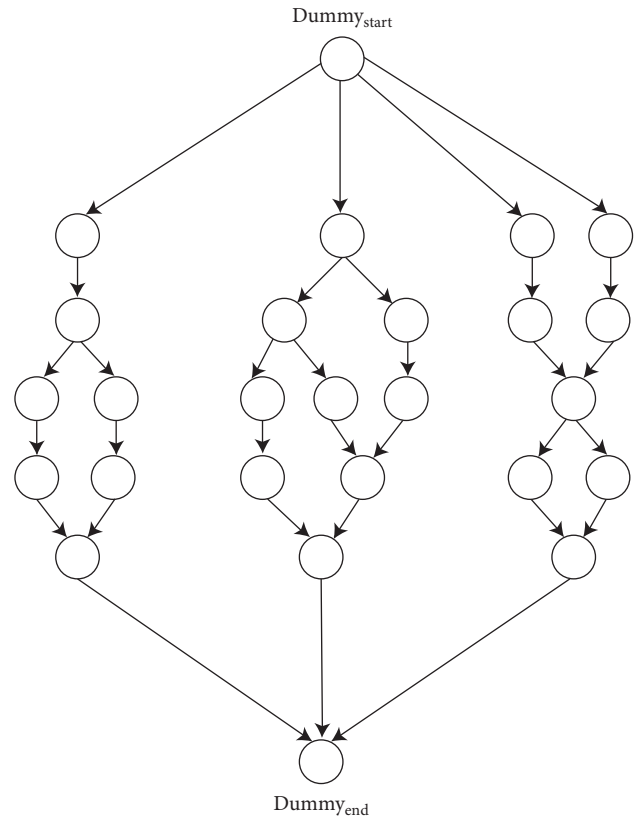

Figure 3: Representation of multiple workflows as a single workflow.

Hence, this paper works on finding an optimal schedule *S* to minimize total execution cost and total execution time based on the definitions given so far.

4. Fruit Fly Optimization

Fruit Fly Optimization Algorithm [7] has been widely adopted for solving global optimization problems because of its simple structure and lesser number of parameters. The algorithm is inspired by the food-finding ability of fruit flies, where they can smell food even at a distance of 40 km. Once

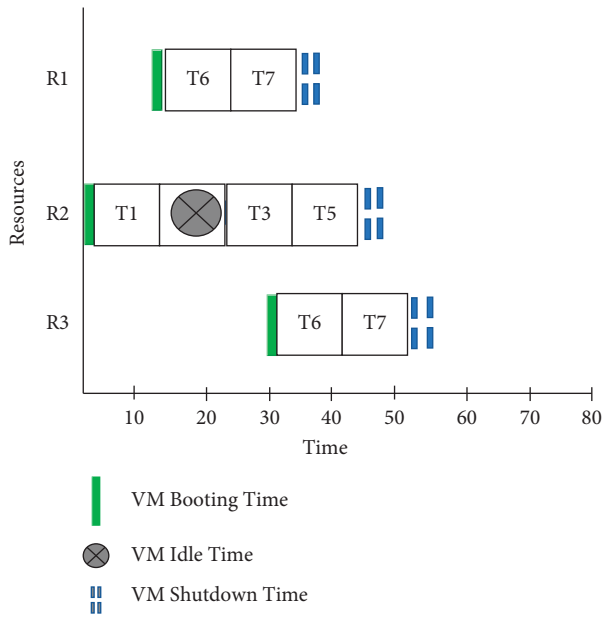

Figure 4: Sample schedule.

they get closer to the food source, they use their sensitive vision for flying towards the food direction.

FFO works in various phases as follows:

(1) The first phase is the initialization phase, where the fruit flies are randomly distributed in the search space, and their location (X_init, Y_init) is initialized

(2) In the second phase, each fruit fly is given some random direction and distance (X_init + RandomValue, Y_init + RandomValue) to move towards the food source

(3) Next, the distance between each fruit fly and the food location is estimated, and the smell concentration is calculated, which is the reciprocal distance

(4) The algorithm then goes into the fitness evaluation phase, which is a function based on the smell concentration

(5) The maximum smell concentration of the individual fruit fly is retained, and the swarm updates its position to move in that direction

(6) These steps are iteratively repeated, and the result of each iteration is compared with the previous one to check whether optimized results are obtained or not

FFO algorithm became popular because of its easy-to-implement structure and quick convergence. However, it is not found suitable for complex optimization problems as it could get trapped in the local optima at later evolution stages and might not reach the global optima. Also, the convergence rate of the algorithm for complex optimization could be improved.

Hence, this paper presents an enhanced version of the traditional FFO algorithm, which could be implemented for complex optimization problems such as scheduling multiple workflows in the cloud computing environment.

4.1. Proposed Framework. Figure 5 presents the proposed framework for IFFO. Consider a cloud provider with a set of virtual machines $VM_l = \{VM_1, VM_2, VM_3, \ldots\ldots, VM_L\}\ \forall\ l \in \{1, 2, 3, \ldots\ldots, L\}$ having some computational capacity. In a cloud computing scenario, there are multiple workflows $W_i = \{W_1, W_2, W_3, \ldots\ldots, W_I\}\ \forall\ i \in \{1, 2, 3, \ldots\ldots, I\}$ that need to be scheduled with optimized QoS parameters. Keeping this in mind, multiple W_i are merged and converted into a unified workflow, S_w and E_w tasks are added at the starting and ending position of W_i. $\forall$ workflows W_i: there may be several tasks $T_j = \{T_1, T_2, T_3, \ldots\ldots, T_J\}\ \forall\ j \in \{1, 2, 3, \ldots, J\}$.

Each resource VM_l is available on-demand, is accessible from a shared pool of computing resources, and has some QoS parameter associated with it. For the present research work, the authors have considered makespan and cost as QoS parameters. Cost of running each VM is $VMC_m = \{VMC_1, VMC_2, VMC_3, \ldots\ldots, VMC_M\}\ \forall\ m \in \{1, 2, 3, \ldots, M\}$ and makespan $VMM_s = \{VMM_1, VMM_2, VMM_3, \ldots\ldots, VMM_S\}\ \forall\ s \in \{1, 2, 3, \ldots, S\}$.

Maximum cost and makespan are calculated by the addition of cost and makespan of each task. The objective of the abovementioned problem is to minimize the cost and makespan for executing the entire workflow. Thus, these multiple objective optimization problems try to find out the Pereto optimal solution in each iteration. Once the cost and makespan are optimized while $T_j \neq T_J$, the IFFO algorithm is applied on input values to find out the best smell function in each iteration. Based on updated smell values, the entire swarm population updates their smell concentration and becomes ready for the next iteration. When maximum iterations are completed, or Pareto optimal solution is achieved, all the tasks are sent for simulation purposes.

4.2. Proposed IFFO Algorithm. The proposed IFFO optimization algorithm is an enhanced version of the traditional Fruit Fly Optimization Algorithm. This algorithm is used to optimize multiple objectives, that is, cost and makespan for multiple workflow scheduling in the cloud environment. The proposed algorithm continuously optimizes the old solution using the smell concentration function. This paper also shows an improvement in coverage rate by updating appropriate positions in each iteration.

The main steps of the proposed method are described as follows.

(1) Input constraints: let n be the swarm size of fruit fly population; the initial position of each fruit fly is $SP_{loc\ p} = \{SP_{loc\ 1}, SP_{loc\ 2}, SP_{loc\ 3}, \ldots, SP_{loc\ P}\}\ \forall\ p \in \{1, 2, 3, \ldots, P\}$. Here, each swarm particle represents a possible solution, moving towards a random direction with randomized distance R_v. As per traditional FFO, the maximum iteration should be {20–40}; for current research work, the maximum iteration (distance) of fruit fly movement is $Q = \{20–40\}$.

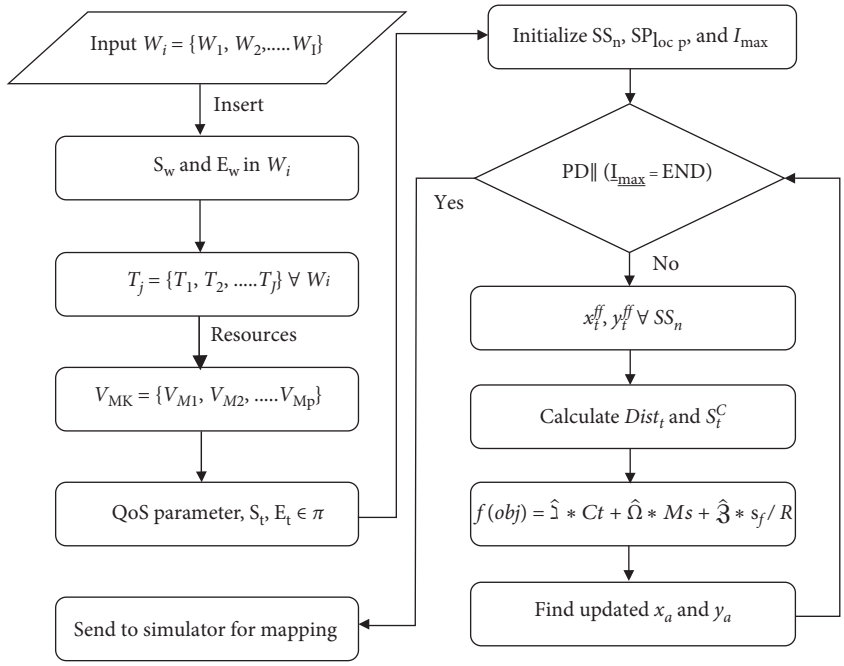

Figure 5: Flowchart of the proposed IFFO algorithm.

(2) Output constraints: the target is to find out rank-1 Pareto optimal solution for given input values. Here, $f(P) = \{S_1, S_2, S_3, \ldots..., S_N\} \forall N \in \{1, 2, 3, \ldots, K\}$ is a set of solutions with lower bound LB and upper bound UB. Z_{CtMs} is the total cost and makespan of workflows that need to be minimized.

(3) This step defines the objective function f(obj), where S_f is the scaling factor, R is a randomized function, and $\hat{\lambda}, \hat{\Omega}$ & $\hat{\beta}$ are arbitrary constant, that is, $\hat{\lambda} + \hat{\Omega} + \hat{\beta} = 1$.

(4) For terminating condition (maximum iteration), I_{max} (t) started from 1 to T.

(5) Calculate the initial position of each swam particle $x_t^{ff} = x_\alpha + R_v$ and $y_t^{ff} = y_\alpha + R_v$, where the initial position $(x_t^{ff}, y_t^{ff})$of each swarm particles $\in SP_{loc\ p} = \{SP_{loc\ 1}, SP_{loc\ 2}, SP_{loc\ 3}, \ldots.., SP_{loc\ P}\}$ and R_v is the random variable ranging from 0 to 1.

(6) Distance between individual swarm and food is calculated by $\sqrt{(x_t^{ff})^2 + (y_t^{ff})^2}$ and smell concentration by $1/Dist_t$.

(7) Calculate $f(S_t^C) \forall SS_q$ where $q = \{1, 2,3, \ldots, n\}$, that is, smell concentration of each individual fruit fly.

(8) Find out the mean of smell concentration $\overline{F}(Smell_t)$.

(9) Update the swarm particles position with updated values of (x_α, y_α), that is, $x_\alpha = x_\alpha + x_\alpha * R_v(0, 1) + x_\alpha * \overline{F}(Smell_t)$ and $y_\alpha = y_\alpha + y_\alpha * R_v(0, 1) + y_\alpha * \overline{F}(Smell_t)$, and go to step 3.

The algorithmic representation of these steps is mentioned below (Algorithm 1).

5. Results and Discussion

5.1. Dataset and Simulation Setup. The experimental analysis was conducted using the CloudSim framework [21], the simulation tool used for simulating cloud environments. The proposed algorithm was implemented for three different datasets, and results were compared with three other metaheuristic optimization techniques, FFO, GA, and PSO. Datasets differ in terms of the number of tasks in a workflow and the number of resources available. Although the cloud environment is considered to have an unlimited set of resources, for arriving at an optimal solution, we need to limit the number of resources as well. We have considered three sample workflows consisting of 15, 25, and 35 tasks. For these three workflows, the number of resources is assumed to be 5, 10, and 15, respectively.

5.2. Performance Analysis. The proposed IFFO algorithm is compared with PSO, GA, and FFO based on two scheduling objectives, makespan, and cost. The algorithms were executed for 20 iterations, and the results depict better performance of IFFO as compared with the other algorithms, both in terms of makespan and cost. The experimental results are presented in the graphs shown in Figures 6–8 for datasets 1, 2, and 3. The blue line represents the cost of execution, while the orange line depicts the makespan. It is clear from these graphs that IFFO outperforms PSO, GA, and FFO in both parameters.

The percentagewise improvement of the proposed algorithm is depicted in Figure 9, which shows that for dataset

(1) Input: SS_n, $SP_{\text{loc }p} = \{SP_{\text{loc }1}, SP_{\text{loc }2}, SP_{\text{loc }3}, \ldots, SP_{\text{loc }P}\}$ and $I_{\max} = \{20\text{-}40\}\ \forall\ p \in \{1, 2, 3, \ldots\ldots, P\}$
//SS_n = Swarn Size, SP_{loc} = initial location of individual swarm particles and $I_{\max}$ = Maximum number of iteration
(2) Output: Pareto optimal solution
$\min_{S_N \in [\text{LB}_N, \text{UB}_N]_{N=1,2,3,\ldots,K}} f(P) = \{S_1, S_2, S_3 \ldots\ldots\ldots S_N\}$
$\therefore \text{QoS} = \sum_{C=1}^{K} \sum_{M=1}^{J} Z_{CtMs}$ and $\text{Out}_{\min} = \min(\text{QoS})$
//S_N are existing solutions, Z_{CtMs} is total cost & makespan of multiple workflows and $\text{Out}_{\min}$ is expected QoS optimized solution
(3) $f(\text{obj}) = \widehat{\Im} * Ct + \widehat{\Omega} * Ms + \widehat{\Im} * s_f / R$
//s_f is scaling factor, $\widehat{\Im} + \widehat{\Omega} + \widehat{\Im} = 1$ and R is a randomized function
(4) for $I_{\max}$ $(t) \leftarrow 1$ to T do
(5) $x_t^{ff} = x_\alpha + R_v$ and $y_t^{ff} = y_\alpha + R_v$
//(x_t^{ff}, y_t^{ff})initial position of each swarm particle and R_v =(0,1)
(6) $\text{Dist}_t = \sqrt{(x_t^{ff})^2 + (y_t^{ff})^2}$ and $S_t^C = 1/\text{Dist}_t$
//Dist_t is distance between individual fruit fly and food, and S_t^C is smell concentration
(7) $\text{Smell}_t = f(S_t^C)$ *//for each individual fruit fly*
(8) $\overline{F}(Smell_t) = 1/\omega \sum_{t=1}^{\omega} f_t(Smell_t)$
(9) Update swarm particles location (x_α, y_α)
9.1. $x_\alpha = x_\alpha + x_\alpha * R_v(0,1) + x_\alpha * \overline{F}(Smell_t)$
9.2. $y_\alpha = y_\alpha + y_\alpha * R_v(0,1) + y_\alpha * \overline{F}(Smell_t)$
9.3. Go to step 3.
(10) **End for**

Algorithm 1: IFFO–QoS optimization for multiple workflow scheduling.

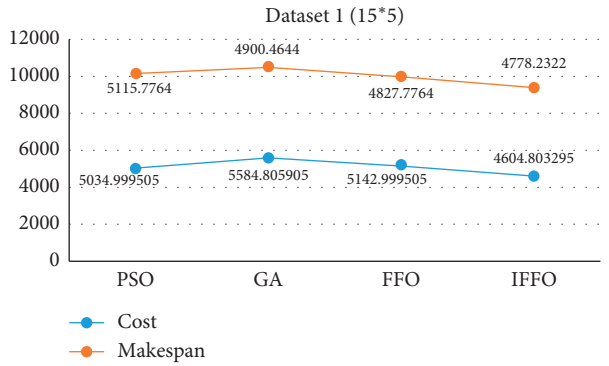

Figure 6: Cost and makespan analysis for dataset 1.

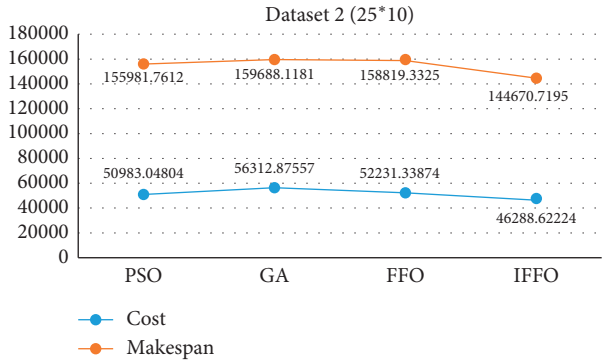

Figure 7: Cost and makespan analysis for dataset 2.

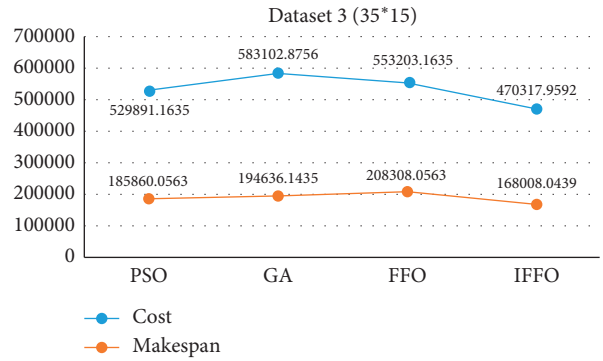

Figure 8: Cost and makespan analysis for dataset 3.

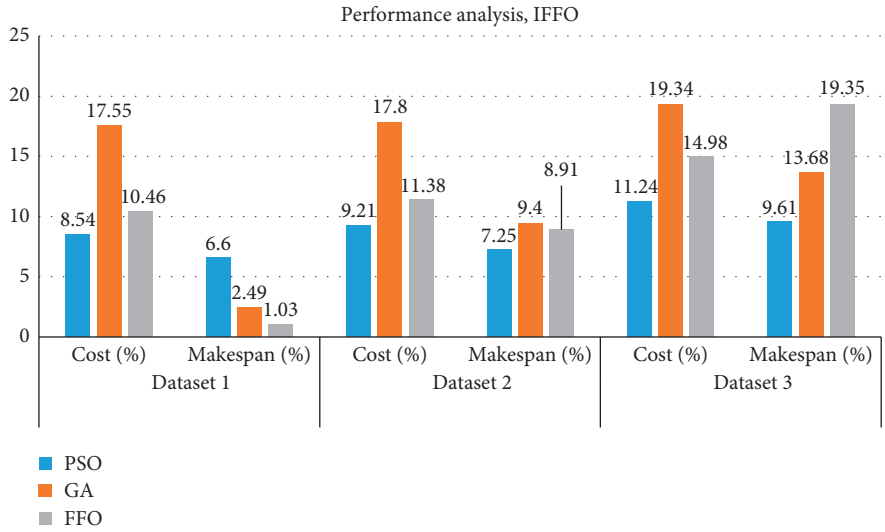

Figure 9: Comparative performance analysis of IFFO with respect to PSO, GA, and FFO.

1, IFFO is 8.54%, 17.55%, and 10.46% better than PSO, GA, and FFO, respectively, in terms of cost and 6.6%, 2.49%, and 1.03% better than PSO, GA, and FFO respectively, in terms of makespan. For dataset 2, the improvement percentage is 9.21%, 17.8%, and 11.38% in terms of cost, and 7.25%, 9.4%, and 8.91% in terms of makespan when compared with PSO, GA, and FFO resp. Similarly, for dataset 3, IFFO showed an improvement of 11.24%, 19.34%, and 14.98% in terms of cost and 9.61%, 13.68%, and 19.35% in terms of makespan when compared with PSO, GA, and FFO, respectively.

The proposed algorithm is capable of optimizing both the parameters simultaneously, unlike many other optimization algorithms where the client has to compromise with one objective while trying to optimize the other. In such cases, a decision has to be made regarding which objective is to be given preference over the other.

6. Conclusion

Scientific workflows play a significant role in large-scale cloud-based applications. In workflow scheduling, nature-inspired algorithms elucidate the promising optimized results for multiobjective problems in the cloud environment. But to avoid local optima trapping problems in multiobjective optimization, traditional nature-inspired techniques continuously try to maintain a balance between exploration and exploitation. In this paper, multiple workflows are considered and merged with dummy start and end nodes to represent it as a single monolithic workflow. The proposed IFFO enhanced the traditional FFO algorithm to minimize the "stuck at the local optima" problem by using an enhanced swarm smell function. The activation function used the mean smell function for the generation of new positions of the swarm particles. The IFFO is used for scheduling multiple workflows to minimize cost and makespan parameters while providing a Pareto optimal solution. The proposed algorithm is implemented on the CloudSim platform, and the result for dataset 1 shows that IFFO is better than PSO, GA, and FFO by 15.14%, 20.04%, and 11.47%, respectively, in terms of cost and makespan conjointly. Similarly, for dataset 2, the proposed algorithm shows 16.46%, 27.2%, and 20.29% improvement. And for dataset 3, the improvement is 20.85%, 33.02%, and 34.33% as compared with PSO, GA, and FFO.

The future scope is to implement the proposed IFFO technique with more QoS parameters such as energy efficiency and load balancing to enhance the overall system performance. The IFFO can be applied in various state-of-the-art research areas like sensor networks, IoT, decision-making system, smart agriculture, and ecological engineering problem.

Data Availability

All data are included within this manuscript.

Conflicts of Interest

The authors declare that they have no conflicts of interest to report regarding the present study.

Acknowledgments

The authors would like to acknowledge the support from Taif University Researchers Supporting Project (no. TURSP-2020/216), Taif University, Taif, Saudi Arabia.

References

[1] B. P. Rimal, A. Jukan, D. Katsaros, and Y. Goeleven, "Architectural requirements for cloud computing systems: an enterprise cloud approach," *Journal of Grid Computing*, vol. 9, no. 1, pp. 3–26, 2010.

[2] A. Aggarwal, P. Dimri, and A. Agarwal, "Survey on scheduling algorithms for multiple workflows in cloud computing environment," *International Journal of Computer Sciences and Engineering*, vol. 7, no. 6, pp. 565–570, 2019.

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

[4] A. Aggarwal, P. Dimri, A. Agarwal, and A. Bhatt, "Self adaptive fruit fly algorithm for multiple workflow scheduling in cloud computing environment," *Kybernetes*, vol. 24, 2020.

[5] M. A. Rodriguez and R. Buyya, "Deadline based resurce provisioning and scheduling algorithm for scientific workflows on clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 222–235, 2014.

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

[7] W.-T. Pan, "A new fruit fly optimization algorithm: taking the financial distress model as an example,," *Knowledge-Based Systems*, vol. 26, pp. 69–74, 2011.

[8] I. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters*, vol. 85, pp. 317–325, 2002.

[9] J. McCall, "Genetic algorithms for modelling and optimisation," *Journal of Computational and Applied Mathematics*, vol. 184, no. 1, pp. 205–222, 2005.

[10] A. Gupta, D. Singh, and M. Kaur, "An efficient image encryption using non-dominated sorting genetic algorithm-III based 4-D chaotic maps," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 3, pp. 1309–1324, 2020.

[11] M. Kaur and D. Singh, "Multi-modality medical image fusion technique using multi-objective differential evolution based deep neural networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 2, pp. 2483–2493, 2021.

[12] D. Rani and R. Ranjan, "A comparative study of SaaS, PaaS and IaaS in cloud computing," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 6, pp. 158–161, 2014.

[13] A. Bhatt, P. Dimri, and A. Aggarwal, "Self-adaptive brainstorming for jobshop scheduling in multicloud environment," *Software Praactice and Experience*, vol. 54, pp. 1–18, 2020.

[14] H. Hua, X. Guangquan, P. Shanchen, and Z. Zenghua, "Adaptive multi-objective task scheduling strategy in cloud computing," *Strategies and Schemes*, vol. 13, pp. 162–171, 2016.

[15] S. Yassa, R. Chelouah, H. Kadima, and B. Granado, "Multi-objective approach for energy-aware workflow scheduling in cloud computing environments," *The Scientific World Journal*, vol. 2013, Article ID 350934, 13 pages, 2013.

[16] L. Liu, M. Zhang, R. Buyya, and Q. Fan, "Deadline-constrained coevolutionary genetic algorithm for scientific workflow scheduling in cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 29, pp. 1–12, 2016.

[17] A. G. Delavar and Y. Aryan, "A hybrid heuristic algorithm for workflow scheduling in cloud systems," *Cluster Computing*, vol. 17, pp. 129–137, 2013.

[18] M. Adhikari and S. Koley, "Cloud Computing: A multi-workflow scheduling algorithm with dynamic reusability," *Arabian Journal for Science and Engineering (AJSE)*, vol. 43, pp. 645–660, 2017.

[19] F. Abazari, M. Analoui, H. Takabi, and S. Fu, "Multi-objective workflow scheduling in cloud computing based on heuristic algorithm," *Simulation Modelling Practice and Theory*, vol. 93, pp. 119–132, 2018.

[20] A. Rehman, S. S. Hussain, Z. U. Rehman, S. Zia, and S. Shamshirband, "Multi-objective approach of energy efficient workflow scheduling in cloud environments," *Concurrency Computat Pract Exper*, vol. 32, pp. 1–20, 2018.

[21] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.