A parallel multi-objective genetic algorithm for scheduling scientific workflows in cloud computing

Muhammad Sardaraz and Muhammad Tahir

Abstract
Recent developments in cloud computing have made it a powerful solution for executing large-scale scientific problems. The complexity of scientific workflows demands efficient utilization of cloud resources to satisfy user requirements. Scheduling of scientific workflows in a cloud environment is a challenge for researchers. The problem is considered as NP-hard. Some constraints such as a heterogeneous environment, dependencies between tasks, quality of service and user deadlines make it difficult for the scheduler to fully utilize available resources. The problem has been extensively studied in the literature. Different researchers have targeted different parameters. This article presents a multi-objective scheduling algorithm for scheduling scientific workflows in cloud computing. The solution is based on genetic algorithm that targets makespan, monetary cost, and load balance. The proposed algorithm first finds the best solution for each parameter. Based on these solutions, the algorithm finds the superbest solution for all parameters. The proposed algorithm is evaluated with benchmark datasets and comparative results with the standard genetic algorithm, particle swarm optimization, and specialized scheduler are presented. The results show that the proposed algorithm achieves an improvement in makespan and reduces the cost with a well load balanced system.

Keywords
Cloud computing, scheduling, scientific workflows, genetic algorithm

Date received: 7 January 2020; accepted: 2 July 2020

Handling Editor: Francesco Longo

Introduction
Cloud computing is the new paradigm of computing in which, instead of local storage, data are stored and accessed over the Internet. Cloud resources such as storage, processing, and network services are provided on demand to the cloud users. Instead of developing in-house infrastructure, service providers provide services to the users by pay per service and negotiation between cloud users and service providers. Cloud resources are virtualized, which enables both the customization of the platform and the isolation of application from the underlying hardware. These resources are accessed from anywhere and applications can be deployed as per user needs. Cloud may be public, private, or hybrid. In public cloud, resources are made available via the Internet. Private clouds are maintained by organizations, whereas hybrid clouds are a mixture of public and private clouds. Cloud services may be infrastructure as a service (IaaS), software as a service (SaaS) or platform as a service (PaaS). IaaS provides necessary infrastructure such as processing and...
Cloud computing has been used for many large computing problems. Scientific workflows are application domains that have attracted cloud computing. Bioinformatics, physics, astronomy, and so on are domains where exemplary scientific workflows exist. The large number of jobs and dependencies between jobs make it difficult to efficiently schedule cloud resources to scientific workflows. A workflow is represented as $G = (V, E)$ where $V$ and $E$ represent vertices and edges, respectively. A workflow consists of $n$ tasks where tasks are represented as vertices and execution flow and dependencies between tasks are represented as edges. The relation between tasks $x$ and $y$ is shown as directed edge $E(x, y)$. Here task $x$ is the parent task and $y$ is the child task, that is, task $y$ cannot start execution before task $x$ is completed.

The scheduling problem is considered to be NP-hard and influenced by some parameters such as quality of service (QoS), user deadlines, execution time, and cost. Early solutions consist of single-objective optimization. The number of cloud users is increasing and more and more applications are shifting toward cloud computing. Serving a large number of users and applications becomes a challenge for cloud service providers. Efficient scheduling mechanisms are needed to efficiently utilize cloud resources and provide better services to the user. Users need QoS in terms of multiple parameters. The scheduling techniques must take into consideration these requirements.

This article presents a multi-objective scheduling algorithm for scientific workflows in cloud computing environments. The proposed algorithm considers makespan, monetary cost, and load balance to schedule workflows. The algorithm is based on genetic algorithm (GA) with multiple-objective criteria. The proposed algorithm first finds the best solution for each parameter, that is, makespan, cost, and load balance. Based on the best solutions obtained for each parameter, the algorithm finds the superbest solution for all parameters. The algorithm selects virtual machines (VMs) with the majority of occurrences in each solution to find the best solution. If no majority is found, the best solution for cost is selected. The proposed method applies GA for optimization in the selection of both the best and superbest solutions. The proposed algorithm is evaluated with benchmark datasets. The experimental results show that the proposed algorithm achieves better results as compared to the other methods. The rest of the article is organized as follows. Section “Related work” presents related work, followed by materials and methods in section “Materials and methods.” Section “Results and discussion” presents the results and discussion and section “Conclusion” concludes the article.

### Related work

Cloud computing is an emerging technique that needs more and more focus for outstanding results. Extensive research has taken place in cloud computing to achieve better performance. Workflow scheduling has been extensively studied. We present some proposed methods in this section. For further details, readers are referred to the reviews on scheduling workflows in cloud environments. Different task scheduling algorithms are used by different clouds in cloud computing such as first come first serve (FCFS), round robin (RR), generalized priority algorithm, match-making, and improved cost-based algorithm. The FCFS algorithm is considered as the energy efficient solution. The technique has been extensively used in many domains including scheduling. The technique suffers from the problem of non-preemptive behavior where the longest waiting time occurs for the shortest job that comes finally. The RR scheduling algorithm is efficient in load balancing; however, all tasks are concurrently active, which leads to high power consumption.

Scheduling algorithms in cloud computing may be task scheduling or workflow scheduling. Workflows may be simple or scientific workflows. Workflow scheduling is complicated as compared to task scheduling due to inter-dependencies among different tasks. The scheduler must consider these dependencies while scheduling workflows. A scheduling algorithm may use standard techniques or optimization techniques such as particle swarm optimization (PSO), genetic algorithm (GA), and ant colony optimization (ACO). This section presents a brief discussion on scheduling algorithms. First, we discuss heuristics-based approaches followed by meta-heuristics. These are closer approaches to our work and hence a comparison is included. First, we discuss heuristic approaches, followed by meta-heuristics based on GA.

Opportunistic load balancing (OLB) assigns each task to the next machine which is expected to be available. The algorithm assigns tasks to machines in an arbitrary order. The technique utilizes more and more computational resources of the machine by keeping them busy all the time as much as possible. This technique assigns tasks to the next available machine, but it ignores the expected execution time of the current task, which results in a poor makespan. OLB does not consider the expected execution time of the current task that leads to the worst makespan, while minimum execution time (MET) considers the mentioned problem. MET assigns tasks randomly to machines. This strategy leads to the best-expected execution time for the current task while the availability of the machine is unknown. MET gives better results by utilizing the best resources for the execution of the current task. The technique leads to load unbalancing. Minimum completion time
(MCT) combines the benefits of OLB and MET for better performance. The algorithm arbitrarily assigns tasks to a machine that has the minimum expected completion time for that task. MCT combines the best features of MET and OLB while avoiding the case where their performance is poor. MCT checks the completion time for all tasks and the task with the lowest expected completion time is assigned to the machine. Min-Min checks all tasks and selects tasks with an overall MCT from the set and assigns to the machine for execution. The selected tasks are removed from the task list. This process is repeated until all tasks are assigned to machines and the task list becomes empty. The algorithm computes the expected execution time for all tasks. Tasks with shortest expected execution time are assigned to machines while tasks with longest execution time wait for resources. All the tasks with the shortest execution time are finished first and tasks with longest expected execution time are executed when resources become available. This leads to performance degradation. The Max-Min algorithm is similar to Min-Min. It calculates the expected execution time for all the unmapped tasks from a list. The algorithm assigns the task with the maximum expected execution time to the best suitable machine while concurrently assigning other tasks with the minimum expected execution time to other machines. The execution of large and small tasks runs concurrently, which leads to better performance.

Alam et al. proposed two algorithms, namely the adaptive round robin load balancing algorithm (ARLBA) and the predictive round robin load balancing algorithm (PRLBA). The ARLBA calculates the maximum capacity and remaining capacity of the server machine which executes the scheduled tasks. When a request for scheduling task is received, the scheduler checks the maximum and remaining capacity of the server. If the request is less than the maximum and remaining capacity of the server, it is assigned to the server; otherwise, the next available server in the list is checked for requirements. The PRLBA uses machine learning techniques by utilizing support vector machine (SVM) to distribute the load on the available servers in the server matrix.

The traditional RR algorithm uses a fixed time slice (quantum) to schedule tasks. If the time quantum is kept large enough, the technique works like FCFS. If the time quantum is kept very small, then context switching between tasks creates overhead with no usage in turn. Selection of the quantum is a major question. Yasin et al. answered to this question. They proposed the idea of variable time quantum using RR and design an algorithm called priority fair round robin (PFRR). The algorithm assigns burst weight and priority weight to tasks and calculates the value of time quantum based on burst weight and priority weight. The time quantum changes according to the priority and burst weights of the task. Promising results are achieved as compared to the traditional RR algorithm. GA randomly initializes the population and evaluates each candidate solution based on the fitness value. In the next phase, the algorithm selects candidates for the next generations. In the third phase, crossover and mutation take place and generations are produced. The process is repeated until the termination criteria are met. In a cloud environment, tasks are considered as genes. The value of gene indicates the resource in the cloud.

A scheduling algorithm based on GA is presented in Verma and Kaushal. The algorithm considers QoS and budget constraints specified by users. Tasks are prioritized as bottom-level and top-level tasks. The initial population is created based on these priorities. The algorithm tests the fitness of each candidate and constructs the initial population. After the crossover and mutation parameters are applied, each candidate is validated according to the fitness function. The authors present comparative results with standard GA to validate the algorithm.

A bi-objective workflow scheduling algorithm for cloud resources is presented in Aryan and Delavar. In the first phase, the algorithm uses bi-directional task prioritization to create an initial population. Before the population is created, all tasks are sorted based on task priority. In the second phase, the algorithm searches for an appropriate solution to optimize the makespan. Crossover and mutation are performed before the termination of the algorithms. Comparative results with specialized schedulers are presented to evaluate the performance of the algorithm.

Another algorithm based on GA is presented in Chen et al. This algorithm targets costs such as computational cost, data transmission cost, and data storage cost while scheduling workflows in the cloud. Tasks are categorized as upward and downward to set priorities of scheduling tasks and merged to construct the initial population. Later, crossover and mutation take place to further optimize the process. Comparative results with standard GA are presented to validate the performance of the algorithm.

Barrett et al. present an algorithm for workflow scheduling in a cloud computing environment. The authors have focused on constraints such as cost and makespan. The algorithm uses GA and Markov process decision to generate optimal schedules and selects schedules according to the environment. Simulations are performed to test the performance of the algorithm.

A dynamic objective GA-based resource scheduling algorithm is presented in Chen et al. The authors have focused on tight deadlines and cost optimization. If GA fails to find a feasible solution, the dynamic objective strategy is used to allow GA to optimize the execution time and achieve user deadlines. Comparative results
with the standard GA are presented to evaluate the algorithm.

A multi-objective workflow scheduling algorithm for a cloud computing environment is presented in Zhu et al.\textsuperscript{16} The authors have presented novel solutions for initialization, fitness evaluation, and other genetic operations. Comparative results with some specialized schedulers on benchmark datasets are presented to evaluate the proposed algorithm.

Rodriguez and Buyya\textsuperscript{9} proposed a hybrid GA to optimize the workflow scheduling problem in heterogeneous systems. The authors have focused on makespan and load balance parameters. A heuristic is presented that produces a solution which is seeded into the initial population to obtain the optimal solution. A rigorous and twofold method is used to find the best solution in less time. Comparative results with evolutionary algorithms are presented to evaluate the performance of the proposed algorithm.

Liu et al.\textsuperscript{13} proposed a co-evolutionary GA for scheduling scientific workflows in cloud computing. The authors proposed an adaptive penalty function to strict constraints. In order to prevent premature, crossover and mutation probabilities are adjusted through the co-evolution process. Comparative results are shown to evaluate the performance of the proposed algorithm.

Casas et al.\textsuperscript{31} proposed a GA-based algorithm for scheduling scientific workflows in a cloud environment. The algorithm exploits the relationship between workflows and the required data. The algorithm is flexible and acts as an interface between the users and service providers in receiving, analyzing, and distributing jobs among available resources. The authors have focused on execution time and monetary cost. Experimental results are shown to validate the performance of the algorithm.

Another algorithm based on GA for task scheduling in cloud environments is proposed in Leena et al.\textsuperscript{32} The authors use a bi-objective optimization strategy to optimize execution time and cost. Simulation results are shown; however, no comparative analysis is presented. A hybrid GA-PSO algorithm for scheduling workflows in cloud computing is proposed in Manasrah and Ba Ali.\textsuperscript{33} The algorithm targets makespan, cost and load balance in heterogeneous cloud environments. The algorithm generates the initial population with GA and then uses PSO on the generated population. The algorithm is evaluated with the experimental results to validate its performance. Choudhary et al.\textsuperscript{34} present a bi-objective hybrid algorithm for workflow scheduling. The algorithm uses the gravitational search algorithm (GSA) and heterogenous finish early time (HFET) techniques to schedule workflows in a cloud computing environment. The method considers the cost ratio and the schedule length ratio as the objective parameters. The results are compared with different existing methods and validated with statistical tests. Another algorithm based on fuzzy dominance sort based HFET is proposed in Xiumin et al.\textsuperscript{35} The algorithm targets cost and makespan as the objective parameters. The integration of fuzzy dominance sort and HFET helps in better handling of the tradeoff between makespan and cost. Comparative results are presented to validate the proposed method. Another technique that targets cost and makespan in a cloud computing environment is proposed in Junlong et al.\textsuperscript{36} The proposed method consists of two phases. In the first phase, the algorithm works as single-objective workflow scheduling that targets cost. In the next phase, the algorithm works as bi-objective workflow scheduling that targets makespan and cost. Comparative results are presented to validate the performance of the method. An algorithm based on the joint meta-heuristic approach for job scheduling in cloud computing is presented in Shojaifar and Javanmardi.\textsuperscript{37} The algorithm uses the fuzzy theory and GA. The algorithm tunes the standard GA with the fuzzy theory to achieve better results. The algorithm focuses on better utilization of VM resources according to the capacity. The experimental results are compared with other methods to validate the performance of the proposed algorithm. Sardaraz and Tahir\textsuperscript{38} present a hybrid algorithm for scheduling workflows in a cloud computing environment. The algorithm uses PSO with the pre-processing phase to achieve better scheduling. In the pre-processing phase, workflow tasks are allocated to available resources according to the capacity of the resources and the decencies between workflow tasks. The algorithm targets makespan, cost and load balance as a multi-objective optimization scheduling problem. The algorithm is validated with the experimental results. Tables 1 and 2 show the experimental environment and targeted parameters of each technique, respectively.

Many solutions proposed to address the problem of workflow scheduling have focused on only one parameter or some researchers have considered the problem as bi-objective optimization. The parameters are correlated with each other. Targeting one parameter influences the other parameters. The case further gets complicated with multi-objective optimization. Over time, more applications are shifting toward the cloud computing environment that leads the resources to be shared among a large number of users. The designed solutions need to address the problem by considering multiple objectives to achieve better resource utilization at low cost and also satisfy other QoS requirements. In addition, the techniques should also be dynamic to handle resources. The proposed algorithm exploits this correlation and achieves better results in terms of resource utilization, execution time, and monetary cost. The proposed technique has the ability to handle
dependencies between tasks with the scheduling ability in the presence of multiple constraints.

Materials and methods

This section presents the details of the proposed algorithm. The proposed algorithm is based on three parameters, that is, makespan, monetary cost, and load balance. The scheduler works as an intermediary between workflow applications and cloud resources. The purpose is to optimize the three parameters. The algorithm assumes that the estimated execution time of each workflow is known in advance. The scheduler also knows the dependencies between different tasks in the workflow. All VMs in the cloud have fixed bandwidth, processing power, memory, cost per quantum of time, and storage. Table 3 shows the symbols used in the proposed algorithm.

Workflows and the resource model

Consider a cloud environment consisting of a pool of resources \((VM_1, VM_2, VM_3, \ldots, VM_n)\) and workflow applications \((W_1, W_2, W_3, \ldots, W_m)\). Each workflow task has execution time measured in MIPS. Tasks also have dependencies, that is, parent and child tasks and the data to be transferred between different tasks. Resources are defined as VMs having CPU, memory, storage, and bandwidth. The goal of the scheduler is to assign resource \(R_i\) to workflow \(W_j\) such that the execution time and cost are minimized. The constraint on the scheduler is to maintain the load balance between all available resources. Makespan is sometimes also referred to as the execution time which is the completion time of the last task in a workflow. Monetary cost is the cost of execution of a particular workflow and the cost of transferring data between different tasks. Load balance measures the load among various nodes in a cloud environment. Load balance is measured as the standard deviation of the load of all nodes. A smaller value means that the system is well load balanced. The objective of this research is to reduce makespan and cost, and maintain load balance among resources. Each resource has processing capacity.

Table 1. Experimental environment used for different scheduling schemes.

| Reference                        | Type of workflow | Simulator | Type |
|----------------------------------|------------------|-----------|------|
| Rodriguez and Buyya\(^9\)       | Scientific       | Dynamic   |      |
| Liu et al.\(^{13}\)             | Scientific       | WorkflowSim | Dynamic |
| Zhu et al.\(^{16}\)             | Scientific       | CloudSim  | Dynamic |
| Priyadarsini and Arockiam\(^{22}\) | Tasks       | CloudSim  | Dynamic |
| Verma and Kaushal\(^{26}\)      | Synthetic        | Java      | Static |
| Aryan and Delavar\(^{27}\)      | Tasks            | Static    |      |
| Chen et al.\(^{28}\)            | Simple           | CloudSim  | Static |
| Barrett et al.\(^{29}\)         | Tasks            | CloudSim  | Static |
| Chen et al.\(^{30}\)            | Simple           | Static    |      |
| Casas et al.\(^{31}\)           | Scientific       | vSphere   | Dynamic |
| Manasrah and Ba Ali\(^{33}\)    | Scientific       | WorkflowSim | Dynamic |
| Choudhary et al.\(^{34}\)       | Scientific       | C++       | Dynamic |
| Xiumin et al.\(^{35}\)          | Scientific       | Java      | Dynamic |
| Junlong et al.\(^{36}\)         | Scientific       | Dynamic   |      |
| Shojafar and Javanmardi\(^{37}\) | Scientific       | CloudSim  | Dynamic |
| Sardaraz and Tahir\(^{38}\)     | Scientific       | CloudSim  | Dynamic |

Blank cells mean that the desired parameter is not shown in the article.

Table 2. Various objectives targeted by different scheduling schemes.

| Reference                        | MS   | Cost | LB   | Deadline | SLA  | Privacy |
|----------------------------------|------|------|------|----------|------|---------|
| Rodriguez and Buyya\(^9\)       | Yes  | –    | Yes  | –        | –    | –       |
| Liu et al.\(^{13}\)             | Yes  | Yes | –    | –        | –    | Yes     |
| Zhu et al.\(^{16}\)             | Yes  | Yes | –    | –        | –    | –       |
| Priyadarsini and Arockiam\(^{22}\) | Yes | Yes | Yes  | Yes      | Yes  | –       |
| Verma and Kaushal\(^{26}\)      | –    | Yes | Yes  | –        | –    | –       |
| Aryan and Delavar\(^{27}\)      | Yes  | –   | –    | –        | –    | –       |
| Chen et al.\(^{28}\)            | –    | Yes | –    | –        | –    | Yes     |
| Barrett et al.\(^{29}\)         | Yes  | Yes | –    | –        | –    | –       |
| Chen et al.\(^{30}\)            | Yes  | Yes | –    | Yes      | –    | –       |
| Casas et al.\(^{31}\)           | Yes  | Yes | –    | –        | –    | –       |
| Manasrah and Ba Ali\(^{33}\)    | Yes  | Yes | Yes  | –        | –    | –       |
| Choudhary et al.\(^{34}\)       | Yes  | Yes | –    | –        | –    | –       |
| Xiumin et al.\(^{35}\)          | Yes  | Yes | –    | Yes      | –    | –       |
| Junlong et al.\(^{36}\)         | Yes  | Yes | Yes  | –        | –    | –       |
| Shojafar and Javanmardi\(^{37}\) | Yes  | Yes | Yes  | –        | –    | –       |
| Sardaraz and Tahir\(^{38}\)     | Yes  | Yes | Yes  | –        | –    | –       |

MS: makespan; LB: load balance; SLA: service-level agreement.
Blank cells mean that the desired parameter is not targeted in the article.
storage capacity, memory, and bandwidth. The processing capacity $C_i$ of $VM_i$ can be calculated with equation (1)

$$C_i = (PE_i \times MIPSi)$$  

(1)

where $PE_i$ is the number of processing elements of $VM_i$. The capacity of all VMs is calculated as shown in equation (2)

$$C = \sum_{i=1}^{n} C_i$$  

(2)

where $n$ is the number of VMs. The load of a VM can be calculated using equation (3)

$$L_{vmi} = \frac{TL}{C_i}$$  

(3)

where $TL$ is the total length of tasks allocated to $vm_i$ and $C_i$ is the capacity of $vm_i$. The load of all VMs can be calculated using equation (4). Equations (1)–(4) are taken from Chunmao et al.40

$$L = \sum_{i=1}^{n} L_{vmi}$$  

(4)

Load balance can be measured as the standard deviation as shown in equation (5).39 Smaller values mean better load management

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (L_{vmi} - L)^2}$$  

(5)

where $L_{vmi}$ refers to the load of $VM_i$, $L$ is the average load of all VMs, and $n$ is the number of VMs

$$MS = FT_{i-1}^{\text{task, time}}$$  

(6)

where $MS$ is the makespan and $FT$ refers to the completion time of a task.38 Monetary cost is calculated using equation (7)

$$MC = TEC + TTC$$  

(7)

where $MC$ refers to the monetary cost of a workflow and $TEC$ and $TTC$ are, respectively, calculated using equations (8) and (9)38

$$TEC_{wi} = \sum_{i=1}^{n} vm_{i}^{\text{time}} \times vm_{i}^{\text{cost}}$$  

(8)

$$TTC_{wi} = \sum_{i=1}^{n} \text{size}_{i} \times \beta_{\text{cost}}$$  

(9)

where $TTC$ refers to the total transfer cost of workflow $wi$, $n$ is the number of tasks in the workflow, $vm_{i}^{\text{time}}$ refers to the time a VM has executed a particular task, and $vm_{i}^{\text{cost}}$ is the cost of the VM for executing a task.

The fitness of a single objective can be calculated using equation (10)39

$$F_i = \min_j$$  

(10)

where $F_i$ is the fitness value of the $i$th parameter and $\min_j$ is the minimum value of the $j$th solution. The fitness of multi-objectives is calculated using equation (11) based on the maximum and minimum values of makespan, cost, and load balance. Sardaraz and Tahir38 calculate the fitness on the basis of makespan and cost including a parameter $w$ for the user to set a weightage to either the makespan or the cost. In the evaluation of the proposed algorithm, an equal weightage was used for all parameters

$$F = w \left( \frac{\max ms - ms}{\max ms - \min ms} \right) + w \left( \frac{\max mc - mc}{\max mc - \min mc} \right) + w \left( \frac{\max \sigma - \sigma}{\max \sigma - \min \sigma} \right)$$  

where $\sigma$ refers to the load balance, which can be calculated using equation (5). In case if the algorithm does not find any better values, the previously selected values are used.

**System model**

Figure 1 shows the architecture of cloud computing used in this article. User tasks are submitted to the task manager. The task manager reads the task information, that is, execution time, dependencies, and data size, and forwards the information to the scheduler. The scheduler is responsible for the allocation of resources according to the proposed method. The global resource manager takes information from the local resource manager and updates the scheduler accordingly.
Multi-objective optimization based on GA

In cloud, scheduling tasks are considered as genes and the value of a gene indicates the resources in the cloud. In the first phase, the GA randomly initializes the whole population and checks the fitness of each candidate solution. In the next phase, the algorithm moves to the selection of candidates for the next generation. In the third phase, the survived chromosomes go through crossover and mutation phases in order to produce the next generations. The process is repeated a specified number of times. Crossover takes place between two chromosomes according to a predefined probability. Mutation randomly changes genes from a chromosome according to a predefined probability. Selection of chromosomes for the next generation depends upon the fitness function used.

The proposed algorithm first finds the best solution for each parameter, that is, makespan, cost, and load balance. GA is used to find the best solution for each parameter. Solutions are selected on the basis of the fitness function as shown in equation (10). After selecting the best solutions for individual parameters, the algorithm selects the superbest solution for all parameters. The superbest solution is selected on the basis of the best solutions identified in the previous step. To find the superbest solution, the algorithm builds a consensus on VMs in the best solution for each parameter. The VMs with majority in the best solutions are selected for the superbest solution. If no majority is found, the best solution for cost is selected.

Algorithm 1. Steps of the proposed algorithm.

**Input:** workflow $w$, task list
**Output:** scheduled plan
1. Generate the initial population (Algorithm 2)
2. $P = \text{fittest chromosomes}$
3. for each parameter do
4. Selection
5. Evaluation
6. Crossover
7. Mutation
8. Select the best solution for each parameter
9. Construct the superbest solution
10. Selection
11. Evaluation
12. Crossover
13. Mutation
14. end
15. return the superbest solution

Algorithm 2. Generate the initial population.

**Input:** workflow $w$
**Output:** initial population
1. genes = number of tasks in $w$
2. $l = \text{largest level in } w$
3. $n = \text{number of tasks in } l$
4. $P = \text{size of the population}$
5. for $i = 1$ to $l$
6. for $j = 1$ to $n$
7. \quad $\text{gene}_j = \text{random value from } [1 \text{ to } n]$
8. \quad $\text{chromosome}_i = \text{gene}_j$
9. end
10. $P = \text{chromosome}_i$
11. end
12. return $P$

GA is used to optimize the problem of assignment of tasks to VMs. The proposed algorithm uses GA in a parallel fashion, that is, GA finds the best solution for each parameter in parallel, selects the fittest chromosomes for each parameter, and goes through crossover and mutation phases. The superbest solution is selected on the basis of the best solutions and crossover and mutation are performed.

The proposed algorithm randomly generates the initial population. First, the number of levels and number of parallel tasks in the workflow are identified. The algorithm randomly assigns VMs to tasks. The size of VMs is maintained according to the number of parallel tasks in the workflow as in Casas et al.\textsuperscript{31} The process is repeated to produce the initial population. Figure 2 shows the working of the proposed algorithm, whereas Algorithm 1 shows the steps involved.

The proposed algorithm uses the roulette wheel selection procedure to select chromosomes. In this process, the selection is based on spinning a partitioned
Partitions are created on the basis of the fitness value of a chromosome. The clustered crossover operator is used for crossover. Clustered crossover is a better option for crossover in workflow scheduling. In this process, clusters of genes are created according to workflow levels. Breaking points in each chromosome are selected and both chromosomes are mixed to produce offspring. Cluster crossover is useful in maintaining the structure of workflows. The swap mutation operator is used in the proposed algorithm. In swap mutation, a pair of genes is selected in a single chromosome and the values of genes are swapped to produce new chromosomes.

### Results and discussion

The proposed algorithm was evaluated using scientific workflows with different tasks and dependency levels. These datasets have been used to validate many scheduling algorithms in a cloud environment. The datasets consist of different real workflow applications with different numbers of tasks, the size of data transferred between different tasks, and the execution time of each task. Five workflow applications are used in the simulation of the proposed work. The Montage dataset is created by NASA/IPAC by closing together multiple images to form custom mosaics of the sky. The Sipht application is used in searching small RNAs in bacterial replication in the NCBI database. The workflow is part of a bioinformatics project at Harvard. The CyberShake application is used to distinguish earthquake threatening regions. The workflow is used by the Southern California Earthquake Centre. The LIGO workflow is used to create and analyze gravitational data of coalescing compact binary systems. The Epigenomics application is used to automate the operations in genome sequence processing. The workflows are parsed to get the desired parameters, that is, execution time of each task, size of data to be transferred, and dependencies between different tasks. The details of workflows are shown in Table 4. Figure 3 shows the structure of the workflows used for the simulation. To evaluate the performance of the proposed algorithm, the algorithm was compared with standard GA, PSO, and specialized schedulers in the same field. Standard GA and PSO are used to make a baseline for comparison. The other algorithm is selected based on the targeted parameters, datasets used, and so on. Simulations were performed on a computer with an Intel Core i3 processor, equipped with 16 GB memory and running the Ubuntu 16.04 operating system. Algorithms were evaluated in terms of makespan, load balance, and monetary cost. CloudSim was used to simulate the algorithms. The number of VMs was set according to datasets, that is, number of parallel tasks in a workflow. VMs with different specifications were selected for the simulation. Each VM was allocated 1000 MB memory and MIPS from 1500 to 3000. Processing, memory, storage, and transfer costs were set to 0.017, 0.05, 0.01, and 0.01, respectively. The

| Dataset       | No. of nodes | w levels | Parallel tasks | Average file size (MB) | Average execution time (s) |
|---------------|--------------|----------|----------------|------------------------|---------------------------|
| Montage       | 100          | 9        | 62             | 18.05                  | 10.19                     |
| Sipht         | 100          | 7        | 51             | 22.34                  | 173.34                    |
| LIGO          | 100          | 8        | 24             | 28.8                   | 209.78                    |
| CyberShake    | 100          | 5        | 48             | 999.41                 | 21.82                     |
| Epigenomics   | 100          | 8        | 24             | 4033.59                | 1277.21                   |

Figure 2. Working of the proposed algorithm.
standard version of GA was simulated with roulette wheel selection, multipoint crossover, and single-point mutation with crossover and mutation probabilities of 0.6 and 0.2, respectively. PSO was executed with recommended values from the literature, that is, the inertia weight factor was set to 1.2 and the learning factor was set to 2. The fitness function and other parameters used to simulate the proposed method were also used for standard GA and PSO. GA-PSO was simulated with parameters and functions discussed in the respective article.

Each algorithm was executed 20 times and the average values are selected for comparison. Each algorithm was executed until it reaches its termination criterion, that is, no improvement in cost or makespan. Table 5 shows the genetic operators and other parameters used in the evaluation of the proposed algorithm.

Table 5. Genetic operators and other parameters used in the evaluation of the proposed algorithm.

| Parameter          | Value  |
|--------------------|--------|
| Crossover rate     | 0.8    |
| Mutation rate      | 0.2    |
| Initial population | 500    |
| Makespan weight factor | 0.33 |
| Cost weight factor  | 0.33   |
| Load balance weight factor | 0.33 |

Table 6 shows the comparative results of different algorithms included in the experiments. The results are shown in terms of makespan, monetary cost, and load balance. In most cases, the proposed algorithm performs better compared to the other algorithms in terms of all the parameters. Figures 4–6 show the percent improvement of the proposed algorithm over the compared algorithm in terms of cost, makespan, and load balance, respectively.

For the Montage dataset, the proposed algorithm achieves 36.92, 32.75, and 10.23 percent improvement gain in makespan over PSO, GA, and GA-PSO, respectively. Percent improvement in the cost of the proposed algorithm over the three algorithms is 34.05, 28.34, and 10.78. In the case of load balance, the proposed algorithm achieved 25.35, 29.41, and 10.85 percent improvement over PSO, GA, and GA-PSO, respectively.

In the case of the Sipht dataset, the proposed algorithm achieved a percent improvement of 35.44, 34.37, and 12.92 for makespan over the compared algorithms. For the same dataset, percent improvement in the cost of the proposed algorithm over PSO, GA, and GA-PSO is 32.91, 35.75, and 10.92, respectively. Percent improvement in terms of the load balance of the proposed algorithm over the compared algorithms is 32.74, 42.20, and 22.05. The proposed algorithm achieved a percent improvement of 33.46, 32.35, and 9.89 in makespan over PSO, GA, and GA-PSO, respectively.

In the case of the CyberShake dataset, the proposed algorithm achieved a percent improvement of 34.16, 32.99, and 9.33 in makespan over PSO, GA, and GA-PSO, respectively. Figures 4–6 show the percent improvement of the proposed algorithm over the compared algorithm in terms of cost, makespan, and load balance, respectively.

For the Montage dataset, the proposed algorithm achieves 36.92, 32.75, and 10.23 percent improvement gain in makespan over PSO, GA, and GA-PSO, respectively. Percent improvement in the cost of the proposed algorithm over the three algorithms is 34.05, 28.34, and 10.78. In the case of load balance, the proposed algorithm achieved 25.35, 29.41, and 10.85 percent improvement over PSO, GA, and GA-PSO, respectively.

In the case of the Sipht dataset, the proposed algorithm achieved a percent improvement of 35.44, 34.37, and 12.92 for makespan over the compared algorithms. For the same dataset, percent improvement in the cost of the proposed algorithm over PSO, GA, and GA-PSO is 32.91, 35.75, and 10.92, respectively. Percent improvement in terms of the load balance of the proposed algorithm over the compared algorithms is 32.74, 42.20, and 22.05. The proposed algorithm achieved a percent improvement of 33.46, 32.35, and 9.89 in makespan over PSO, GA, and GA-PSO, respectively.

In the case of the CyberShake dataset, the proposed algorithm achieved a percent improvement of 34.16, 32.99, and 9.33 in makespan over PSO, GA, and GA-PSO, respectively. Figures 4–6 show the percent improvement of the proposed algorithm over the compared algorithm in terms of cost, makespan, and load balance, respectively.

For the Montage dataset, the proposed algorithm achieves 36.92, 32.75, and 10.23 percent improvement gain in makespan over PSO, GA, and GA-PSO, respectively. Percent improvement in the cost of the proposed algorithm over the three algorithms is 34.05, 28.34, and 10.78. In the case of load balance, the proposed algorithm achieved 25.35, 29.41, and 10.85 percent improvement over PSO, GA, and GA-PSO, respectively.

In the case of the Sipht dataset, the proposed algorithm achieved a percent improvement of 35.44, 34.37, and 12.92 for makespan over the compared algorithms. For the same dataset, percent improvement in the cost of the proposed algorithm over PSO, GA, and GA-PSO is 32.91, 35.75, and 10.92, respectively. Percent improvement in terms of the load balance of the proposed algorithm over the compared algorithms is 32.74, 42.20, and 22.05. The proposed algorithm achieved a percent improvement of 33.46, 32.35, and 9.89 in makespan over PSO, GA, and GA-PSO, respectively.

In the case of the CyberShake dataset, the proposed algorithm achieved a percent improvement of 34.16, 32.99, and 9.33 in makespan over PSO, GA, and GA-PSO, respectively. Figures 4–6 show the percent improvement of the proposed algorithm over the compared algorithm in terms of cost, makespan, and load balance, respectively.

For the Montage dataset, the proposed algorithm achieves 36.92, 32.75, and 10.23 percent improvement gain in makespan over PSO, GA, and GA-PSO, respectively. Percent improvement in the cost of the proposed algorithm over the three algorithms is 34.05, 28.34, and 10.78. In the case of load balance, the proposed algorithm achieved 25.35, 29.41, and 10.85 percent improvement over PSO, GA, and GA-PSO, respectively.

In the case of the Sipht dataset, the proposed algorithm achieved a percent improvement of 35.44, 34.37, and 12.92 for makespan over the compared algorithms. For the same dataset, percent improvement in the cost of the proposed algorithm over PSO, GA, and GA-PSO is 32.91, 35.75, and 10.92, respectively. Percent improvement in terms of the load balance of the proposed algorithm over the compared algorithms is 32.74, 42.20, and 22.05. The proposed algorithm achieved a percent improvement of 33.46, 32.35, and 9.89 in makespan over PSO, GA, and GA-PSO, respectively.

In the case of the CyberShake dataset, the proposed algorithm achieved a percent improvement of 34.16, 32.99, and 9.33 in makespan over PSO, GA, and GA-PSO, respectively. Figures 4–6 show the percent improvement of the proposed algorithm over the compared algorithm in terms of cost, makespan, and load balance, respectively.
PSO, respectively. For the cost parameter, the percent improvement of the proposed algorithm over the compared algorithms is 33.14, 35.29, and 20.39. In the case of load balance, the percent improvement over PSO, GA and GA-PSO is 22.55, 27.49, and 13.74, respectively.

For the epigenomics dataset, the percent improvement attained by the proposed algorithm in makespan over PSO, GA, and GA-PSO is 56.4, 55.9, and 8.9, respectively. Percent improvement in cost over the compared algorithms is 32.2, 34.42, and 16.13. In the case of load balance, the percent improvement of the proposed algorithm over the compared algorithms is 28.51, 31.53, and 19.45.

Workflows in these datasets have different characteristics, that is, CPU intensive, data intensive, or memory intensive. By calculating the fittest solutions for each parameter, the proposed algorithm makes scheduling efficient according to the intensity of the workflow tasks. Taking the superbest solution from the previous best solutions results in better scheduling as multi-objective optimization. The problem is first

Table 6. Comparative results of the proposed algorithm with other scheduling algorithms.

| Parameters | PSO | GA | GA-PSO | Proposed algorithm |
|------------|-----|----|--------|---------------------|
| Montage    |     |    |        |                     |
| Makespan   | 306 | 287| 215    | 193                 |
| Cost       | 1.38| 1.27|1.02    | 0.91                |
| LB         | 209 | 221| 175    | 156                 |
| Sipht      |     |    |        |                     |
| Makespan   | 4821| 4742|3574    | 3112                |
| Cost       | 1.58| 1.65|1.19    | 1.06                |
| LB         | 226 | 263| 195    | 152                 |
| LIGO       |     |    |        |                     |
| Makespan   | 4871| 4791|3597    | 3241                |
| Cost       | 1.42| 1.49|1.21    | 1.02                |
| LB         | 231 | 248| 208    | 159                 |
| CyberShake |     |    |        |                     |
| Makespan   | 6241| 6132|4532    | 4109                |
| Cost       | 1.81| 1.87|1.52    | 1.21                |
| LB         | 235 | 251| 211    | 182                 |
| Epigenomics|     |    |        |                     |
| Makespan   | 73,642| 72,916|35,247 | 32,104              |
| Cost       | 6.21 | 6.42| 5.02   | 4.21                |
| LB         | 249 | 260| 221    | 178                 |

PSO: particle swarm optimization; GA: genetic algorithm; LB: load balance.

Makespan is shown in seconds, and cost is shown in dollars. LB is calculated using equation (5). The average results of 20 runs are shown.

Figure 4. Percent improvement gain in the cost of the proposed algorithm over other algorithms.
considered as single-objective optimization that yields the best solutions for each parameter. These solutions are used as the population for multi-objective optimization in the next phase that results in finding the best solutions. Instead of testing the fitness function for infeasible solutions, the algorithm tests the fittest solutions from the previous phase.

Conclusion

This article presents a novel scheduling algorithm for scheduling scientific workflows in a cloud computing environment. The proposed algorithm targets execution time and monetary cost while considering load balance among computing nodes. The algorithm first considers the problem as single-objective optimization and then converts it into multi-objective optimization. The proposed algorithm is validated with experiments on benchmark datasets that show better performance in comparison to other state-of-the-art algorithms. The proposed algorithm achieves a percent improvement gain of up to 37% in makespan, 33% in cost, and 11% in load balance for the Montage dataset. Similar improvements for other datasets are also observed. The results of the proposed method show that targeting multiple parameters can improve the cloud performance with better resource utilization. In the present state, the proposed algorithm focused on three parameters, that is, makespan, cost, and load balance. There is a need to correlate other parameters, for example, QoS, security, and privacy, to achieve better improvement.
scheduling. The dynamic allocation and scheduling of cloud resources need to be addressed to achieve better performance and QoS.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iD
Muhammad Tahir https://orcid.org/0000-0002-7750-8959

References
1. Hao Y, Liu G and Lu J. Three levels load balancing on CloudSim. Int J Grid Distrib Comput 2014; 7(3): 71–88.
2. Gao Y, Guan H, Qi Z, et al. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J Comput Syst Sci 2013; 79(8): 1230–1242.
3. Khola M, Sardaraz M and Tahir M. Evolutionary algorithms in cloud computing from the perspective of energy consumption: a review. In: 2018 14th international conference on emerging technologies (ICET), Islamabad, Pakistan, 21–22 November 2018, pp.1–6. New York: IEEE.
4. Foster I, Zhao Y, Raiciu I, et al. Cloud computing and grid computing 360-degree compared. arXiv preprint arXiv:0901.0131, 2008.
5. Zhang Q, Cheng L and Boutaba R. Cloud computing: state of-the-art and research challenges. J Internet Serv Appl 2010; 1(1): 7–18.
6. Masdari M, Nabavi SS and Ahmadi V. An overview of virtual machine placement schemes in cloud computing. J Netw Comput Appl 2016; 66: 106–127.
7. Pandey S, Wu L, Guru SM, et al. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: 2010 24th IEEE international conference on advanced information networking and applications, Perth, WA, Australia, 20–23 April 2010, pp.400–407. New York: IEEE.
8. Masdari M, Salehi F, Jalali M, et al. A survey of PSO-based scheduling algorithms in cloud computing. J Netw Syst Manag 2017; 25(1): 122–158.
9. Rodriguez MA and Buyya R. Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. IEEE T Cloud Comput 2014; 2(2): 222–235.
10. Mei L, Chan WK and Tse T. A tale of clouds: paradigm comparisons and some thoughts on research issues. In: 2008 IEEE Asia-Pacific services computing conference, Yilan, Taiwan, 9–12 December 2008, pp.464–469. New York: IEEE.
11. Ahmad SG, Liew CS, Munir EU, et al. A hybrid genetic algorithm for optimization of scheduling workflow applications in heterogeneous computing systems. J Parallel Distr Com 2016; 87: 80–90.
12. Juve G, Chervenak A, Deelman E, et al. Characterizing and profiling scientific workflows. Future Gen Comp Syst 2013; 29(3): 682–692.
13. Liu L, Zhang M, Buyya R, et al. Deadline-constrained coevolutionary genetic algorithm for scientific workflow scheduling in cloud computing. Concurr Comp: Pract E 2017; 29(5): e3942.
14. Masdari M, ValiKardan S, Shahi Z, et al. Towards workflow scheduling in cloud computing: a comprehensive analysis. J Netw Comput Appl 2016; 66: 64–82.
15. Rodriguez MA and Buyya R. A taxonomy and survey on scheduling algorithms for scientific workflows in IAAS cloud computing environments. Concurr Comp: Pract E 2017; 29(8): e4041.
16. Zhu Z, Zhang G, Li M, et al. Evolutionary multi-objective workflow scheduling in cloud. IEEE T Parall Dist 2015; 27(5): 1344–1357.
17. Mittal S and Katal A. An optimized task scheduling algorithm in cloud computing. In: 2016 IEEE 6th international conference on advanced computing (IACC), Bhimavaram, India, 27–28 February 2016, pp.197–202. New York: IEEE.
18. Kong X, Lin C, Jiang Y, et al. Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction. J Netw Comput Appl 2011; 34(4): 1068–1077.
19. Braun TD, Siegel HJ, Beck N, et al. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J Parallel Distrib Com 2001; 61(6): 810–837.
20. Freund RF, Gherrity M, Ambrosius S, et al. Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet. In: Proceedings seventh heterogeneous computing workshop (HCW98), Orlando, FL, 30 March 1998, pp.184–199. New York: IEEE.
21. Armstrong R, Hensgen D and Kidd T. The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions. In: Proceedings seventh heterogeneous computing workshop (HCW98), Orlando, FL, 30 March 1998, pp.79–87. New York: IEEE.
22. Priyadarsini RJ and Arockiam L. Performance evaluation of Min-Min and Max-Min algorithms for job scheduling in federated cloud. Int J Comp Appl 2014; 99(18): 47–54.
23. Ibarra OH and Kim CE. Heuristic algorithms for scheduling independent tasks on nonidentical processors. JACM 1977; 24(2): 280–289.
24. Alam F, Thayananthan V and Katib I. Analysis of round-robin load-balancing algorithm with adaptive and predictive approaches. In: 2016 UKACC 11th international conference on control (CONTROL), Belfast, 31 August–2 September 2016, pp.1–7. New York: IEEE.
25. Yasin A, Faraz A and Rehman S. Prioritized fair round robin algorithm with variable time quantum. In: 2015 13th international conference on Frontiers of Information Technology (FIT), Islamabad, Pakistan, 14–16 December 2015, pp.314–319. New York: IEEE.
26. Verma A and Kaushal S. Budget constrained priority based genetic algorithm for workflow scheduling in cloud. In: Fifth international conference on advances in recent
technologies in communication and computing, Bangalore, India, 20–21 September 2013, pp.314–319. London: IET.

27. Aryan Y and Delavar AG. A bi-objective workflow application scheduling in cloud computing systems. *Int J Integr Technol Educ* 2014; 3(2): 51–62.

28. Chen C, Liu J, Wen Y, et al. A hybrid genetic algorithm for privacy and cost aware scheduling of data intensive workflow in cloud. In: *International conference on algorithms and architectures for parallel processing*, Zhangjiajie, China, 18–20 November 2015, pp.578–591. Cham: Springer.

29. Barrett E, Howley E and Duggan J. A learning architecture for scheduling workflow applications in the cloud. In: *2011 IEEE ninth European conference on web services*, Lugano, 14–16 September 2011, pp.83–90. New York: IEEE.

30. Chen ZG, Zhan ZH, Li HH, et al. Deadline constrained cloud computing resources scheduling through an ant colony system approach. In: *2015 international conference on cloud computing research and innovation (ICCCRI)*, Singapore, 26–27 October 2015, pp.112–119. New York: IEEE.

31. Casas I, Taheri J, Ranjan R, et al. GA-ETI: an enhanced genetic algorithm for the scheduling of scientific workflows in cloud environments. *J Comput Sci* 2018; 26: 318–331.

32. Leena V, Ajeena Beegom AS and Rajasree M. Genetic algorithm based bi-objective task scheduling in hybrid cloud platform. *Int J Comput Theor Eng* 2016; 8(1): 7.

33. Manasrah AM and Ba Ali H. Workflow scheduling using hybrid GA-PSO algorithm in cloud computing. *Wirel Commun Mob Com* 2018; 2018: 1934784.

34. Choudhary A, Gupta I, Singh V, et al. A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing. *Future Gener Comp Sy* 2018; 83: 14–26.

35. Xiumin Z, Gongxuan Z, Jin S, et al. Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT. *Future Gener Comp Sy* 2019; 93: 278–289.

36. Junlong Z, Tian W, Peijin C, et al. Cost and makespan-aware workflow scheduling in hybrid clouds. *J Syst Architect* 2019; 100: 101631.

37. Shojafar M and Javanmardi S. FUGE: a joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. *Cluster Comput* 2015; 18(2): 829–844.

38. Sardaraz M and Tahir M. A hybrid algorithm for scheduling scientific workflows in cloud computing. *IEEE Access* 2019; 7(1): 186137–186146.

39. Wu X, Deng M, Zhang R, et al. A task scheduling algorithm based on QoS-driven in cloud computing. *Procedia Comput Sci* 2013; 17: 1162–1169.

40. Chunmao J, Junwei W and Zhicong L. Adaptive thresholds determination for saving cloud energy using three-way decisions. *Cluster Comput* 2019; 22(4): 8475–8482.

41. Bharathi S, Chervenak A, Deelman E, et al. Characterization of scientific workflows. In: *2008 third workshop on workflows in support of large-scale science*, Austin, TX, 17 November 2008, pp.1–10. New York: IEEE.