Yadav and Mishra
Journal of Cloud Computing (2023) 12:8
https://doi.org/10.1186/s13677-023-00392-z

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH **Open Access**

An enhanced ordinal optimization with lower scheduling overhead based novel approach for task scheduling in cloud computing environment

Monika Yadav* and Atul Mishra

Abstract
Efficient utilization of available computing resources in Cloud computing is one of the most challenging problems for cloud providers. This requires the design of an efficient and optimal task-scheduling strategy that can play a vital role in the functioning and overall performance of the cloud computing system. Optimal Schedules are specifically needed for scheduling virtual machines in fluctuating & unpredictable dynamic cloud scenario. Although there exist numerous approaches for enhancing task scheduling in the cloud environment, it is still an open issue. The paper focuses on an improved & enhanced ordinal optimization technique to reduce the large search space for optimal scheduling in the minimum time to achieve the goal of minimum makespan. To meet the current requirement of optimal schedule for minimum makespan, ordinal optimization that uses horse race conditions for selection rules is applied in an enhanced reiterative manner to achieve low overhead by smartly allocating the load to the most promising schedule. This proposed ordinal optimization technique and linear regression generate optimal schedules that help achieve minimum makespan. Furthermore, the proposed mathematical equation, derived using linear regression, predicts any future dynamic workload for a minimum makespan period target.

Keywords Cloud computing, Ordinal optimization, Makespan, CloudSim, Schedules

Introduction

Cloud computing has revolutionized the way computing resources and services are delivered dynamically in a virtualized manner over the Internet. Computing is delivered using a utility-based business model, wherein, on-demand delivery of computing power on a pay-as-you-go basis like traditional services such as electricity, water, gas, or telephony takes place. Cloud service providers, also known as hyper scalers, make accessing cloud services easier. Customers receive cloud services from cloud service providers vice Level Agreements (SLA).

Cloud computing is characterized by distinct features like multi-tenancy, scalability, elasticity, pooling of resources, and virtualization. Based on these, the cloud service provider deploys cloud services such as IaaS (Infrastructure as a service), SaaS (Software as a service,) and PaaS (Platform as a service). For the deployment of these service models and efficient utilization of cloud resources, the providers rely on the deployment of scheduling algorithms. These algorithms ensure that resources are easily available on demand, resources are efficiently utilized under high/low load conditions and the cost of using resources is reduced.

The virtualization technology theoretically enables an infinitude of resources for the cloud. But even if cloud

*Correspondence:
Monika Yadav
yadavmonika506@gmail.com
J.C. Bose University of Science and Technology, YMCA, Faridabad, India

service providers practically control an endless number of resources, consumers may experience issues when trying to access resources and services. The primary cause of this is an improper mapping of physical resources to virtual machines, which impedes the performance of cloud users. The issue of the uneven mapping of virtual resources to the physical infrastructure is solved through scheduling.

Task scheduling and allocating resources in the right order and with the least delay to improve system performance is difficult in a cloud environment. Due to the complexity of the cloud and real-time mapping of tasks with the virtual machines and then virtual machines mapping with the host machine, scheduling of tasks in cloud computing becomes an NP-Hard problem. Therefore, this study offers a solution to task scheduling problems with an advanced ordinal optimization technique. The ordinal optimization (OO) method extracts the best schedules from all candidate schedules currently available in a cloud environment. Furthermore, a mathematical equation is also suggested to schedule any task on the cloud with the shortest makespan period based on the regression technique designed for these selected ideal schedules.

The major contributions of this paper are summarized as follows:

1. A testbed for the candidate schedules was designed using CloudSim Simulator for applying and testing the proposed approach.
2. An enhanced Ordinal Optimization methodology with lower Scheduling overhead has been proposed which will give the optimal schedules from the currently available candidate schedules.
3. Linear regression technique is applied to predict the future scheduling of the cloudlets for obtaining the minimum possible makespan for a given set of available optimal schedules.
4. The proposed approach is experimentally investigated using the CloudSim simulator.

The remainder of the paper is organized as follows. The related studies that investigate task scheduling and resource allocation and associated works are addressed in the section "Related work". Section "Problem statement and formulation" discusses the problem statement and formulation behind this work. Section "Proposed approach" explains the proposed methodology along with the proposed Algorithm. Result discussions and comparison with the existing Blind Pick and Monte Carlo algorithms are covered in the section "Cloud simulation results and discussions". Finally, concluding remarks and future directions are presented in the section "Conclusion".

Related work

Several studies have set one's sight on resolving the task scheduling issue in the distributed environment in the past decade. Scheduling of large-scale workloads on distributed cloud platforms has been already explored by several researchers formerly [1–5]. In the last few years, several other crucial characteristics along with scheduling in the distributed cloud environment are also explored by imminent researchers. In some of the proposed research methods, features such as authentication, security, and load balancing are included besides scheduling [6–11]. To achieve a higher throughput storage architecture, Donghyun et al. [12] provide a storage data audit scheme for fog-assisted cloud storage with no need to modify the existing end-user IoT terminal devices. This section addresses the recently proposed algorithms for scheduling workloads in the cloud computing platform.

Dogan et al. [13] algorithm for scheduling applications with the ultimate goal of merest execution time, least completion time, and opportunistic load balance. The author claims that the proposed algorithm also diminishes the failure probability. Smith et al. [14] presented heuristic approaches including auction, min-min, and max-min. In this, the authors suggested two ways to implement a vigorous metric for heuristics Infrastructure allocation. Samrat Nath et al. [15] proposed a dynamic scheduling policy based on Deep Reinforcement Learning (DRL) with the Deep Deterministic Policy Gradient (DDPG) method to solve the problem of Mobile User task offloading in a Mobile edge computing server.

Buyya et al. [16] present a new model based on the budget, deadline, and Quality of Service with the ultimate goal to find an optimized solution for task scheduling-based problems. This genetic-based algorithm aims to solve deadline and cost-limitation-based optimization issues by addressing the heterogeneous and booking-based service-oriented distributed environment. Zhang et al. [17] based on multiple factors proposed a heuristic ant colony algorithm with a better VM fault-tolerant placement solution for scheduling service-providing virtual machines and conventional heuristic algorithms are used for scheduling the redundant virtual machines.

Benoit et al. [18] came forward with an approach that distributes the workloads based on the knowledge of resources available at any particular point in time. To schedule the cluster the traditional preemptive scheduling mechanism is used to map the distinguished virtual machines on a single host machine resulting in adding new approximations and heuristics to the algorithm. Gawali et al. [19] improve the performance of the system using a modified Cuckoo Optimization algorithm based on standard deviation. This two-phase Algorithm chooses the appropriate population sample in the first phase for optimal results and then applies the proposed algorithm to this sample population in the second phase.

Li and Buyya [20] designed a simulation-based model to schedule the workloads in a Grid computing environment. To calculate the accuracy of workload correlations the experiment is conducted in a model-driven simulation. Simulation results at the local and grid level indicate the decline in performance and indicate that the autocorrelation of loads in this model is not ideal. Lu and Zomaya [21] presented an integrated workload scheduling mechanism for executing tasks in the heterogeneous environment for grids to reduce the average response time for the workloads. Suitable for a wide network of computational grids, this policy is a tradeoff between the advantages of distributed networks such as workload balancing, fault-tolerant, and the pros of the centralized environment such as inherent efficiency.

Linear programming is one of the most common techniques used for optimization. This mechanism is used to obtain the most optimal solution for task scheduling with the given constraints on minimum makespan and maximum throughput. Bossche et al. [22] come up with a time and cost-limitation-based task scheduling algorithm for Infrastructure as a service platform.

Bertot et al. [23] reviewed the Monte Carlo Method for cloud simulation enhancement. Their work shows that by generating random schedules one can obtain the optimal schedules with high precision. For fluctuating scheduling periods this method gives high system throughput and at the same time shows a decline in memory demand. But for the fluctuation in tasks with a large scheduling period Monte Carlo result in a decline in the overall system performance.

Blind-Pick can be applied to a diminished search space that can evolve with the rapid fluctuation in workload having mediocre overhead. This approach with moderate precision and not-so-ideal set selection result in degradation in overall system performance [24, 25]. A preemption-based divide and conquer methodology is used by Gawali et al. [26] for the virtual machine status which allocates resources in the cloud with improved turnaround time and response time.

In genetic-based approaches, the researcher usually aims to enhance the system throughput without being concerned about the overall system performance. As the focus is on the appropriate execution of the task with proper usage of infrastructure in the defined mode of execution. Genetic Algorithms follow an appropriate procedure for candidate selection, fitness evaluation, mutation, and variations. Gu et al. [27] described an infrastructure scheduling strategy for a cloud computing platform based on a genetic algorithm. Zhao et al. [28] proposed a scheduler based on genetics with the main goal to reduce makespan by using chromosome-based coding schemes and implementing them on a numerically based simulator. Shaoxing Zhu et al. [29] came forward with a more stability-based evolutionary scheduling algorithm. Infrastructure as a Service model is benefited from this multi-objective genetic algorithm.

Fast variation in workload gives rise to the need for an algorithm that schedules tasks with high throughput and degrades memory demand scheduling workloads in a multitasking environment. That can adapt to variation in the working environment with low overhead to obtain the optimal schedules [30, 31].

Having presented various studies carried out related to task scheduling in cloud computing environments, it is observed that the Monte Carlo simulations may require time in months to produce optimal schedules with the burden of high fluctuations resulting in degrading the overall performance. On the other hand, Blind Pick simulations give better results with a small search space, but on increasing the size of the search space, the performance degrades. Reducing these scheduling overheads is necessary for real-time cloud computing. A novel Task scheduling method based on ordinal optimization (OO) is presented in this paper. This new approach outperforms the Monte Carlo and Blind-Pick methods to yield higher performance.

Problem statement and formulation

As stated earlier, efficient utilization of available computing resources, in a highly dynamic cloud computing environment, is one of the most challenging problems for cloud service providers. Proper scheduling of tasks to available computing resources and allocation of resources in a manner that no node in the cloud is overloaded and all the available resources in the cloud do not undergo any kind of wastage is the key to it. This requires the design of an efficient and optimal task-scheduling strategy that can play a vital role in the functioning and overall performance of the cloud computing system. The following problem statement and its formulation, presented in this paper, derive from this stated fact.

Problem statement

To design a low overhead scheduling algorithm for mapping tasks to available computing resources in a cloud, by extracting the best schedules from all candidate schedules in a given search space, in the minimum time, to achieve the overall goal of minimum makespan. Furthermore, the solution may be able to predict the minimum makespan for a given load condition.

Problem formulation

The resource allocation and task scheduling in a cloud computing environment is an NP-Hard problem, as it requires more than polynomial time to reach a solution. (i.e., harder than hardest problem). To elaborate, let us consider n the number of tasks to be assigned to r number of resources. The number of computations required for assigning the 'n' number of tasks to the r number of resources is calculated using the formula $^{n}C_{r}$. For example, if 20 tasks are assigned to 6 resources then the number of computations is $^{20}C_6 = 38{,}760$. If we add only one additional task to this set, then the number of computations would grow many folds i.e., $^{21}C_6 = 54{,}264$. For a large number of tasks, it requires more than polynomial time to allocate resources to tasks. Task scheduling and resource allocation, in the cloud, takes place at two distinct layers. Tasks are mapped to Virtual machines based on their configuration and availability and thereafter virtual machines are mapped to physical hosts. This two-layer mapping in a cloud environment increases the complexity and size of the possible search space for finding the optimal schedules designed to achieve the overall goal of minimum makespan. Thus, more than polynomial time is required to allocate the available resources to the tasks in real-time resulting in making scheduling an NP-Hard problem in cloud computing. So instead of finding the time-consuming ideal solution, it's better to find the "good enough" optimal solution in the shortest possible time.

Several approaches have been proposed for solving this prominent issue of scheduling in the cloud but still, all the problems are not fully addressed. There are still a few challenges faced by the existing approaches. The main challenge is to reduce the scheduling overhead. However, in a true cloud platform, resource profiling and stage-based simulations are often run with thousands or millions of possible schedules when the best solution is needed. Generating an optimal schedule in the cloud can take weeks. To create an optimal schedule using Monte Carlo simulations, too long a simulation time of weeks may be required. Reducing this scheduling overhead is essential in real-time cloud computing.

The work proposed in this paper tries to handle the above-discussed issues and provide a more promising result. This paper proposes a new workload planning method to schedule cloudlets in the cloud by reducing the search space significantly to lower the scheduling overhead.

Proposed approach

The real-world problem needs a real-world solution. One can say that the ideal solution is just a theoretical concept as it is unattainable and is not cost-effective also. Finding the accurate solution for a problem seems to be an unrealistic & time-consuming task. Lack of structure, a wide range of uncertainties, and the presence of a vast search space in a cloud environment give rise to the need to quickly narrow down the available good enough scheduling approach rather than sticking to the time-consuming more accurate scheduling approach. This leads us to the concept of comparing orders first and then estimating their value second. In the other words, it's the ordinal optimization before cardinal optimization.

Ordinal Optimizations focus on influencing the strategic change of goals. Figure 1 illustrates the basic concept of Ordinal Optimization. The two basic principles of OO are:

1. Decisive Order is more elementary than value. In layman's terms, one can say that it's much easier to determine which stone that you are holding in both of your hands is heavy rather than telling the difference in their weight.
2. Goal Softening eliminates the computational burden of finding the optimal solution. Instead of asking for the "best for sure" one can settle for the "good enough with high probability". A given problem is much easier to solve by softening the goal of optimization.

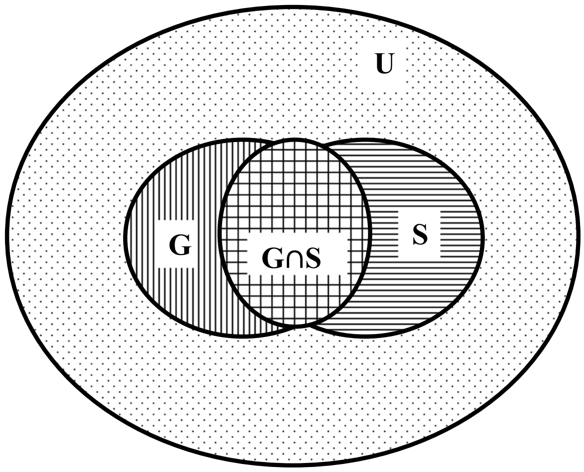


Fig. 1 The generalized concept for Optimization

In this work Horse Race condition (HR) is used with Ordinal optimization to narrow down the search space for selecting the Optimal Schedule. The HR can be pictured as having all Schedules in the search space competing with each other at the same time, similar to *N* horses running a race [32]. During the analysis process, some of the schedules might be leading at a particular point in time and the same schedules might be lagging at another instant in time. The positions of the Schedules are determined by the estimated time taken to complete the given task. Just like a race is stopped for all the horses concurrently. Similarly, for subset selection for Ordinal Optimization, all the schedules are stopped simultaneously and the performance of each schedule at the stopping time is analyzed.

Let's mathematically formalize the proposed approach. Table 1 depicts the basic notations used in the proposed work. Suppose we are having the search space as a set of candidate schedules (U), where θ is an individual schedule such that (θ∈ U). The top-g Schedules selected using HR_ne out of the available candidate schedules (U) are termed as the "good enough" schedules of subset G using the preemption methodology. g denotes the Size of the subset G. With approx. Similar cardinality and HR_e pick another subset S called as "selected subset". i.e., $|S| \cong |G|$. The selection criteria of subset S directly affect the probability of finding the optimal schedule. Truly good schedules inside S are termed as k(≤g) such that u> > g> > s> > k. In other words, k is the number of schedules of subset S that are also the member of subset G. Probability of finding the schedules with a variation in noise as i given by $P(|G\cap| \geq k : \sigma^2, N)$. This alignment probability can be made more accurate by increasing the size of G and S.

Table 1 Basic notations

Notation	Definition
OO	Ordinal Optimization
U	A set of all possible schedules in search space is termed a Candidate Set.
N	Total number of available Schedules
θ	A Schedule that is an element of the Candidate Set U.
u	The cardinality of Set U, i.e., $|U| = u$.
G	A Subset of U that has good enough schedules, is termed an Acceptance Set.
g	The cardinality of Set G, i.e., $|G| = g$.
S	A Subset of U has the most promising Schedule termed a Selection Set.
s	The cardinality of Set G, i.e., $|S| = s$.
M	Time taken to execute all the tasks, termed as Makespan
k	The cardinality of Set G ∩ S, i.e., $|G \cap S| = k$.
HR	Horse Race condition
HR_ne	Horse Race condition with no elimination
HR_e	Horse Race condition with elimination
OPC	Ordered Performance Curve
σ	variation in noise

The conceptual flow chart for the proposed work is depicted in Fig. 2. It searches for good enough schedules and insists on aiming for the best schedules. Ordinal Optimization is a tradeoff between accurate and good enough with high probability. This enhanced Ordinal Optimization approach is applied to the real-time cloud environment to obtain the optimal schedules in a minimum Makespan. The complete approach can be explained with the help of pseudocode as in Algorithms 1 and 2.

```
1.  Input: Cloudlets    // tasks to be executed in the cloud environment
2.  Output:  Optimal Schedules
3.  For datacenter =1 to n
    // creation of data center
4.  for VM =1 to n
    // creation of Virtual Machine
    4.1 machine configuration (MIPS), for each Virtual Machine, is random within the range
        (25-1000)
5.  for cloudlet = 1 to n
    // Creation of cloudlets
6.  for Schedule[θ] =1 to n do
    // Designing of candidate schedules
    6.1 fordata center =1 to n do
        6.1.1    Datacenter [i] = VM[ rand()% n - 1 ] +1
                 //Randomly schedule VMs to the data center using Time shared scheduler
    6.2 for VM =1 to n do
        6.2.1     VM [i] = cloudlet[ rand()% n -1 ] + 1
                  //Randomly schedule cloudlet to VMs using Space Shared Scheduler
    6.3  Calculate Makespan[θ] for each schedule
        // Execute task and calculate the Execution Time
7.  end for
8.  A non-decreasing Ordered performance curve is plotted to depict the performance value
    against the design.
9.  Average Makespan = slope(OPC)
10. Applying Subset Selection rules for OO
    10.1    Selection of Subset G using preemption methodology
        //Selecting Good Enough Schedules (G) by applying HR_ne
        10.1.1  Input: OPC
        10.1.2  for Schedule[θ] =1 to n do
            10.1.2.1          if  Makespan[θ] <Average Makespan then
            10.1.2.2           Subset G[ ] = Schedule[θ]
            10.1.2.3          end for
    10.2    Selection of Subset S
        // Selecting Acceptance Schedules (S) by applying HR_e with global comparison
        10.2.1   for Schedule[θ] =1 to n do
        10.2.2   if  Makespan[θ] <Makespan[θ +1] then
        10.2.3  Subset S[ ] = Schedule[θ]
        10.2.4  end if
        10.2.5  else
        10.2.6  Subset S[ ] = Schedule[θ+1]
        10.2.7  end else
        10.2.8  θ = θ +2
        10.2.9  end for
    10.3    Selection of Optimal Schedule k
        10.3.1  Optimal Schedules = G∩S
                // Each Optimal schedule will be having a different configuration
```

Algorithm 1. Enhanced Ordinal Optimization Algorithm

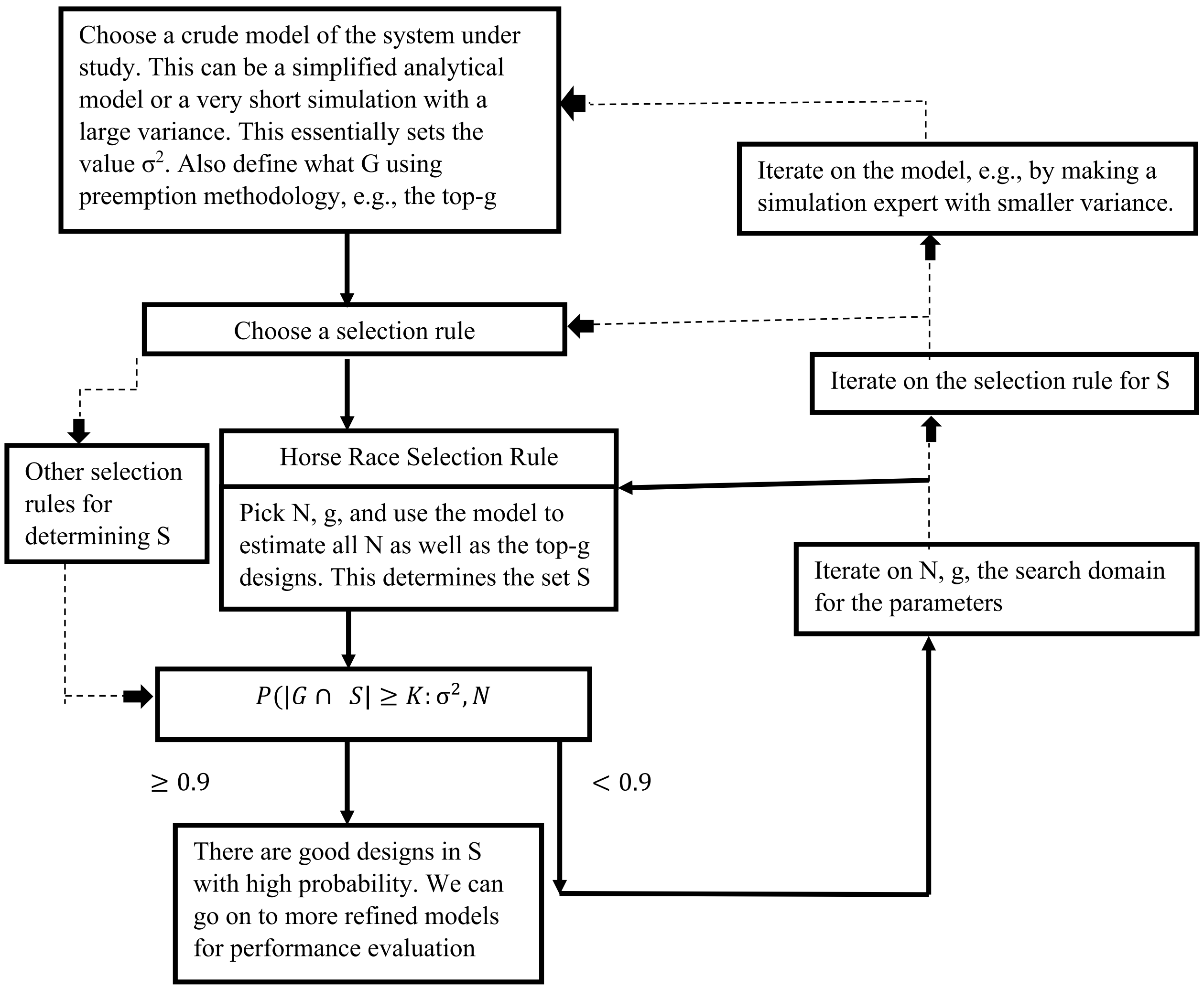

Choose a crude model of the system under study. This can be a simplified analytical model or a very short simulation with a large variance. This essentially sets the value σ^2. Also define what G using preemption methodology, e.g., the top-g
Iterate on the model, e.g., by making a simulation expert with smaller variance.
Choose a selection rule
Iterate on the selection rule for S
Other selection rules for determining S
Horse Race Selection Rule
Pick N, g, and use the model to estimate all N as well as the top-g designs. This determines the set S
Iterate on N, g, the search domain for the parameters
$P(|G \cap S| \geq K: \sigma^2, N$
≥ 0.9
< 0.9
There are good designs in S with high probability. We can go on to more refined models for performance evaluation

Fig. 2 Flow Chart for enhanced Ordinal Optimization

The proposed approach is simulated on a cloud computing environment that provides a real-time cloud computing scenario. The configuration details of the data centers, Virtual Machines, and cloudlets used in the customized simulation setup are given in Table 2 and consist of general information on the data centers, such as the number of data centers, the number of Virtual Machines, the number of cloudlets, etc. Algorithm 1, gives a detailed description regarding the creation of a testbed for applying the proposed approach. Initially, 5 different data centers are created then 25 random Virtual machines with different configurations are created. Two hundred fifty varying cloudlets are then created. Virtual machines are scheduled to data centers using time shared scheduler and cloudlets are scheduled to the virtual machine using space shared scheduler for the designing of 30 candidate schedules. Horse Race condition (HR) is used with Ordinal optimization to narrow down the search space for selecting the Optimal Schedule. The positions of the Schedules are determined by the estimated time taken to complete the given task. The top-g Schedules selected using Horse race without elimination out of the available candidate schedules (U) are termed as the "good enough" schedules of subset G

Table 2 Specifications for designing all the candidate schedules

No. of Datacentres created	5
No. of cloudlets	250
No. of VMs	25
Cloudlet Scheduler	Space Share
VM Scheduler	Time Share
M/C configuration (MIPS)	random (250–1000)

using the preemption methodology. Horse Race with elimination picks another subset S called as "selected subset. Then using ordinal optimization most promising schedules are selected.

```
1. Input: Varying Workload on Optimal Schedules
2. Output: Mathematical Equation to schedule cloudlets with minimum possible Makespan
3. for Optimal Schedule (θ) =1 to | G∩S | do
        3.1 Workload[4] = rand()% [1000 -200 ] + 1
        3.2 for Workload[i] do
        3.3 Makespan (θ) = execution time(Optimal Schedule (θ))
            //Execute task and calculate the Execution Time
4. end for
5. Makespan Vs Load graph is plotted
6. Linear regression is applied to the graph plotted in step 5.
        6.1 Calculate the Mean of varying workloads in the cloud environment($M_x$)
        6.2 Calculate the standard deviation of varying workloads in the cloud
            environment($S_x$)
        6.3 Calculate the Mean of the average Makespan for each optimal schedule ($M_y$)
        6.4 Calculate the standard deviation of the average Makespan for each optimal
            schedule ($S_y$)
        6.5 Calculate the correlation (r) between X and Y
7. Getting the Best fitted line for optimal schedule
        7.1 Calculate slope (b) for the best-fitted line
            $b = \frac{rSy}{Sx}$                                          (1)
        7.2 Calculate intercept (A)
            $A = M_y - bM_x$                                              (2)
        7.3 Determining the best-fitted Line
            $Y' = bX + A$                                                 (3)
8. The mathematical equation 3 for the best-fitted line will schedule cloudlets in the
   minimum possible Makespan by using the available optimal schedules.
```

Algorithm 2. Mathematical Equation to schedule cloudlets with minimum possible Makespan

Algorithm 2, derives a mathematical equation, that can be used to predict the possible minimum makespan for cloudlets that are coming in the future and scheduled on the optimal schedule obtained from Algorithm 1. Through Ordinal Optimization 10 optimal schedules are selected. Each schedule has a different configuration. Four different types of loads are applied on each optimal schedule. Four different workloads of 250, 300, 350, and 400 cloudlets are applied. Makespan corresponding to them is recorded. Graph corresponding to these cloudlets and makespan is plotted and linear regression is then applied. The slope and intercept of this graph are calculated and finally, Eq. 3 gives the mathematical equation for scheduling future cloudlets on these optimal schedules in the minimum possible makespan.

Designing of candidate schedules (U) for applying ordinal optimization

CloudSim is a simulation tool that provides a platform for developing a cloud architecture model that supports services and infrastructure provided by the cloud. Researchers can experiment with their work on this tool as it looks and feels like a cloud platform with all the variation and fluctuation required to implement the work [33]. In our earlier work, CloudSim version 3.0 is used to design the search space of candidate schedules [34]. The same Candidate set (U) is used in this work. Each schedule is denoted by θ.

The candidate schedules set consists of 30 schedules' = {θ1, θ_2, θ_3…… θ_{30}}.

A set of 30 schedules were designed using the following parameters on CloudSim 3.0.

a) Number of Datacenters
b) The varying number of virtual machines in a Datacenter
c) Machine configuration of each virtual machine in the data center
d) Number of cloudlets executing in a particular datacenter
e) Type of scheduling policy.

Figure 3 shows that cloudlets are assigned to Virtual Machines by space-shared scheduling and virtual machines are assigned to hosts in the data center by Time Shared Scheduling. These 30 schedules will act as a testbed for applying the proposed enhanced ordinal optimization.

In Fig. 4, the values of the makespan are the actual Performance distributed cloud environment of applying the Candidate Schedule schedules. The graph depicts the performance of each schedule based on makespan.

Ordered performance curve

Based on the Makespan, Schedules are plotted from the smallest to the largest to form a nondecreasing curve, which is named an ordered performance curve (OPC) in OO [35]. In Fig. 5 OPC is the plot of performance value against the designs i.e., Makespan vs. Candidate Schedules. By using OPC Average Makespan is coming as 948.853 which depicts the average performance of the schedules.

Subset selection rules for OO

Ordinal optimization uses selection rules for selecting the Subset G & S. But before choosing the appropriate selection rules it must go through the below questions:

1. Set S is selected by ordering all top designs using cardinal value assessment and comparing them either pair-wise or globally.
2. Initial Computing budget is assigned to the design either by iterating the initial design with elimination or without elimination.

After reviewing the above two questions, appropriate approaches are used in ordinal optimization. The horse Race condition is used when the initial estimate is made for the performance of each schedule using the crude model to select the top s schedules. HR with

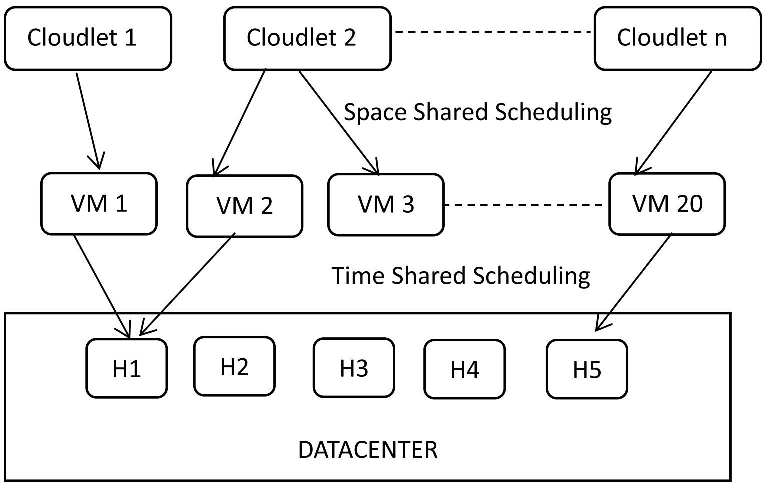


Fig. 3 Scheduling architecture diagram

no elimination (HR_ne) is used when the proposed model compares the mean values of all the candidate schedules.

Selection of subset G (good enough schedules)

- ❖ Ordered performance curve
- ❖ HR(horse race) with no elimination (HR_ne)

HR with no elimination (HR_ne) is used for the selection of Subset G, this approach compares the mean values of all the candidate schedules using preemption methodology. The Schedules having a Makespan less than the Average value of OPC are selected and termed as the top best schedules. From Fig. 5 these top schedules form the set G of good enough schedules.

$$G = \{\theta_3, \theta_{16}, \theta_{19}, \theta_{25}, \theta_{11}, \theta_{30}, \theta_{18}, \theta_6, \theta_{26}, \theta_{24}, \theta_{23}, \theta_7, \theta_9, \theta_{17}, \theta_{13}, \theta_2, \theta_{10}\}$$

them. In the end, a list of Sorted schedules is obtained in descending order.

This technique compares two schedules and eliminates the one which has a larger Makespan. The whole candidate set U is reduced to 15 schedules. From U = {θ_1, θ_2, θ_3…….. θ_{30}} below schedules are selected, and set S is formed.

$$S = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_8, \theta_{11}, \theta_{13}, \theta_{16}, \theta_{18}, \theta_{19}, \theta_{24}, \theta_{26}, \theta_{29}, \theta_{30}\}$$

Finding GΠS

In the Ordinal Optimization approach, the set (G∩S) results in k optimal schedules which are good enough schedules obtained from Set S & G.

Number of Candidate schedules, U = 30
Number of Good enough subset, G = 17
Number of accepted schedules, S = 15
GΠS = 10

$$G = \{\theta_3, \theta_{16}, \theta_{19}, \theta_{25}, \theta_{11}, \theta_{30}, \theta_{18}, \theta_6, \theta_{26}, \theta_{24}, \theta_{23}, \theta_7, \theta_9, \theta_{17}, \theta_{13}, \theta_2, \theta_{10}\}$$
$$S = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_8, \theta_{11}, \theta_{13}, \theta_{16}, \theta_{18}, \theta_{19}, \theta_{24}, \theta_{26}, \theta_{29}, \theta_{30}\}$$
$$G\Pi S = \{\theta_2, \theta_3, \theta_{11}, \theta_{13}, \theta_{16}, \theta_{18}, \theta_{19}, \theta_{24}, \theta_{26}, \theta_{30}\}$$

Selection of subset S (acceptance schedule)

Acceptance schedule is selected by Horse Race methodology with global comparison i.e., HR_e. In this mechanism, the best schedule of each comparison round receives one makespan value, and then that champion schedule is compared with other schedules based on the makespan value. The winner of each round is kept in every successive round of comparison whereas the other schedules are simply eliminated by dumping

Cloud simulation results and discussions

Hereafter, it presents how these optimum schedules work with different loads in the cloud computing environment.

Experiment conditions

Through Ordinal Optimization 10 optimum schedules are selected.

So $G\Pi S = \{\theta_2, \theta_3, \theta_{11}, \theta_{13}, \theta_{16}, \theta_{18}, \theta_{19}, \theta_{24}, \theta_{26}, \theta_{30}\}$.

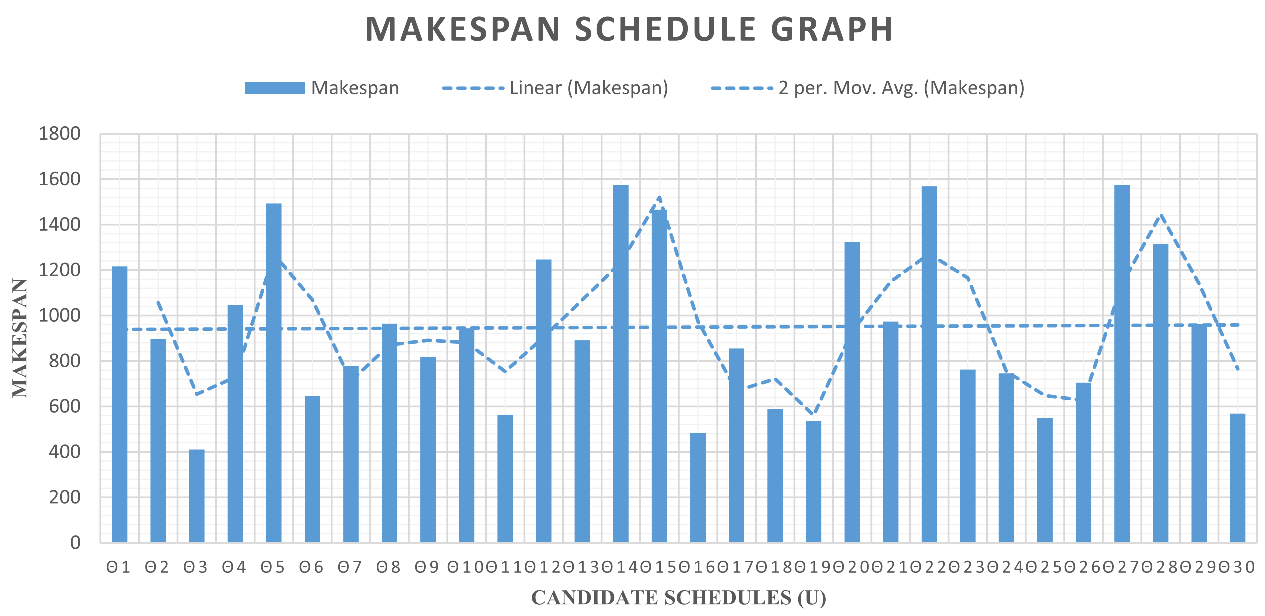


Fig. 4 Makespan vs. Schedule graph

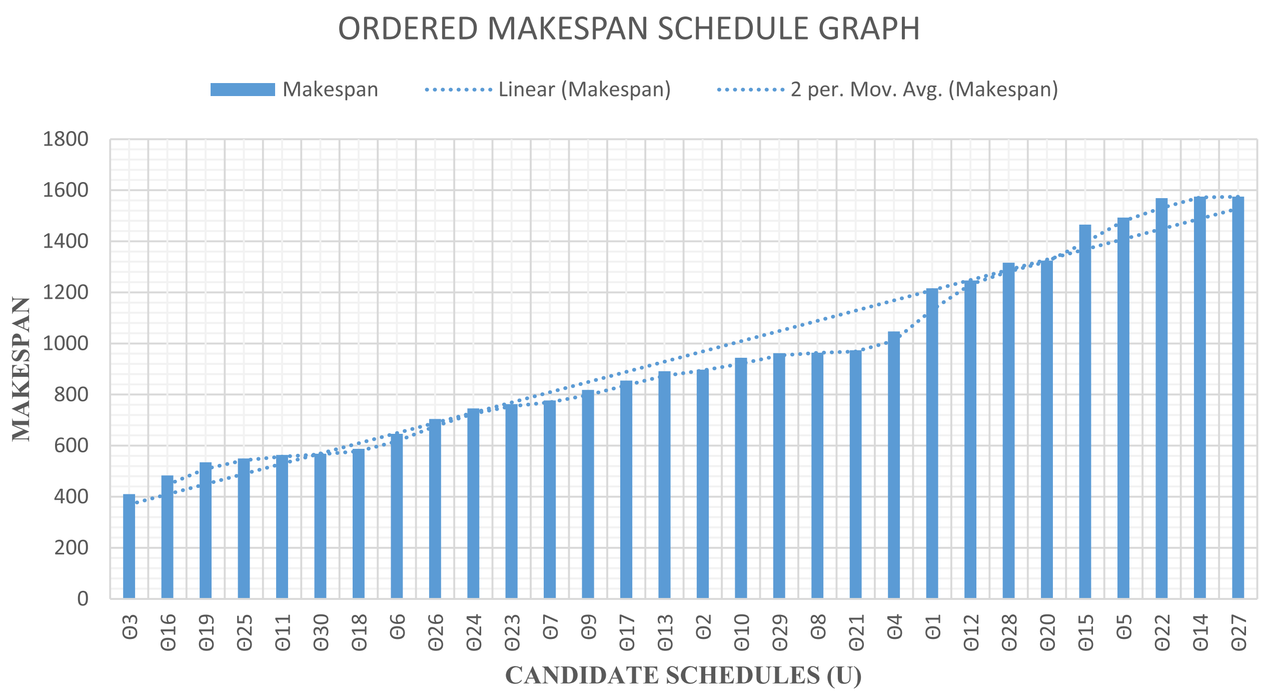


Fig. 5 Ordered performance curve

Each schedule has a different configuration. Four different types of loads are applied on each schedule. Four different workloads of 250, 300, 350, and 400 cloudlets are applied.

Table 3 shows the Makespan corresponding to each schedule and load. A graph plotted as Makespan vs. Load for analyzing these optimal schedules. Now different types of loads are applied to these GΠS schedules and plot a graph between Load and Makespan for GΠSis shown in Fig. 6.

Numerical analysis of the proposed approach

Forecasting the outcome of one parameter based on the result of another parameter is termed linear regression. The criterion variable (Y) is the variable for which the

Table 3 Makespan of schedule vs. load

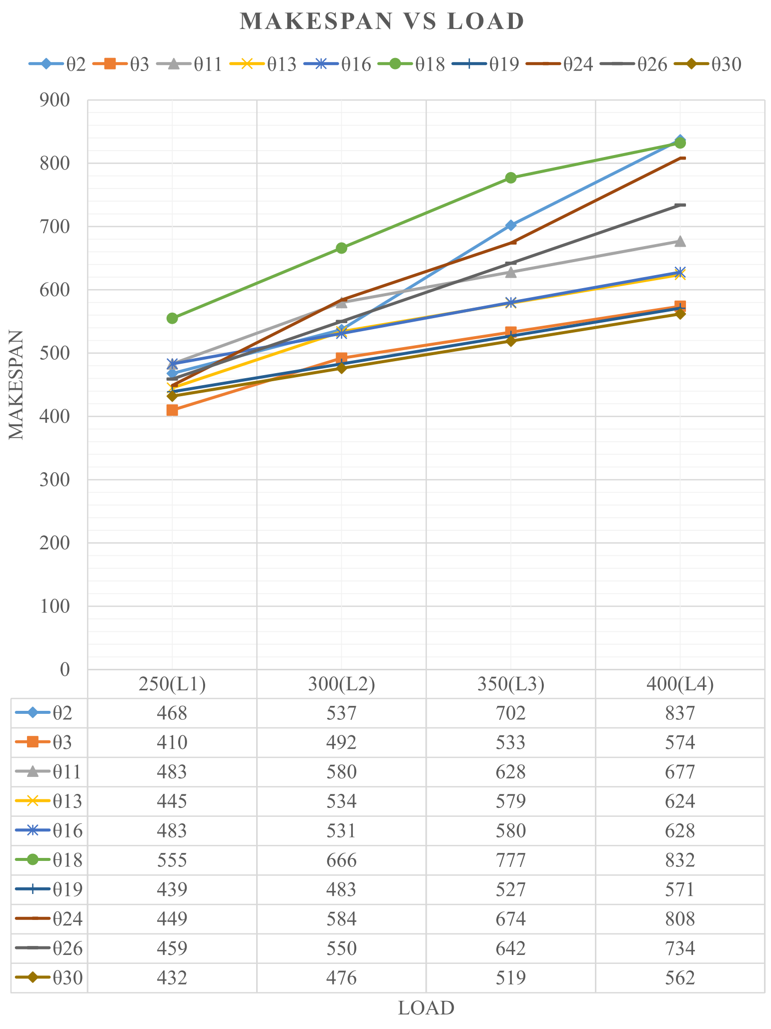
Schedules➔ ↓Load	θ2	θ3	θ11	θ13	θ16	θ18	θ19	θ24	θ26	θ30
250(L1)	468	410	483	445	483	555	439	449	459	432
300(L2)	537	492	580	534	531	666	483	584	550	476
350(L3)	702	533	628	579	580	777	527	674	642	519
400(L4)	837	574	677	624	628	832	571	808	734	562

value is being predicted. The predictor variable (X) is the variable based on which forecasting of the Criterion variable is done. Criterion variable X and Y for the varying workload is depicted in Table 4. In the case of simple regression, there is only one Predictor Variable (X). A straight line as a slope is obtained when the Criterion variable (Y) is plotted as a function of the Predictor Variable (X).

Linear regression aims at uncovering the best-fitting undeviating line through all the values of the graph. The tailor-made line is referred to as a regression line.

Computing the regression line

- ➢ The mean of X is denoted by $M_{x.}$
- ➢ The Mean of Y is denoted by $M_{y.}$



	250(L1)	300(L2)	350(L3)	400(L4)
θ2	468	537	702	837
θ3	410	492	533	574
θ11	483	580	628	677
θ13	445	534	579	624
θ16	483	531	580	628
θ18	555	666	777	832
θ19	439	483	527	571
θ24	449	584	674	808
θ26	459	550	642	734
θ30	432	476	519	562

Fig. 6 Makespan vs. Load

Table 4 Input table for linear regression

X	Y
250	462.3
300	543.3
350	616.1
400	684.7

Table 5 Calculus for computing the regression line

M_x	M_y	S_x	S_y	*r*
325	576.6	64.54972	95.60202	0.999284(high correlation)

- ➢ The standard deviation of X is denoted by $S_{x.}$
- ➢ The standard deviation of Y is denoted by $S_{y.}$
- ➢ correlation between X and Y is coined by r.s

Table 5 depicts the calculus for computing the regression line and the undeviating line of Fig. 7 depicts the slope i.e., the band it is derived as follows:

$$b = \frac{rSy}{Sx} \quad b = 1.4799 \quad b = 1.48 \tag{1}$$

A is the intercept and the given below formula can be used to calculate it

$$A = M_y - bM_x \quad \mathbf{A = 95.6} \tag{2}$$

The regression line is calculated by the below formula:

$$Y' = bX + A \tag{3}$$

$$\mathbf{Y' = (1.48)\ X + 95.6\ (best - fitted\ line)} \tag{4}$$

To calculate the minimum Makespan of the optimum schedule for a given load on the cloud by using the above Eq. 4.

The fallacy in forecasting cannot be eliminated. For any schedule the fallacy of forecasting is the value of schedule (Y) subtracted predicted value (Y′) i.e., the value on the best-fitted line. Table 6 shows the predicted values (Y′) and the errors of prediction (Y-Y′). Column (Y-Y′) ^{2} depicts the squared error of forecasting. The Sum of squared errors of

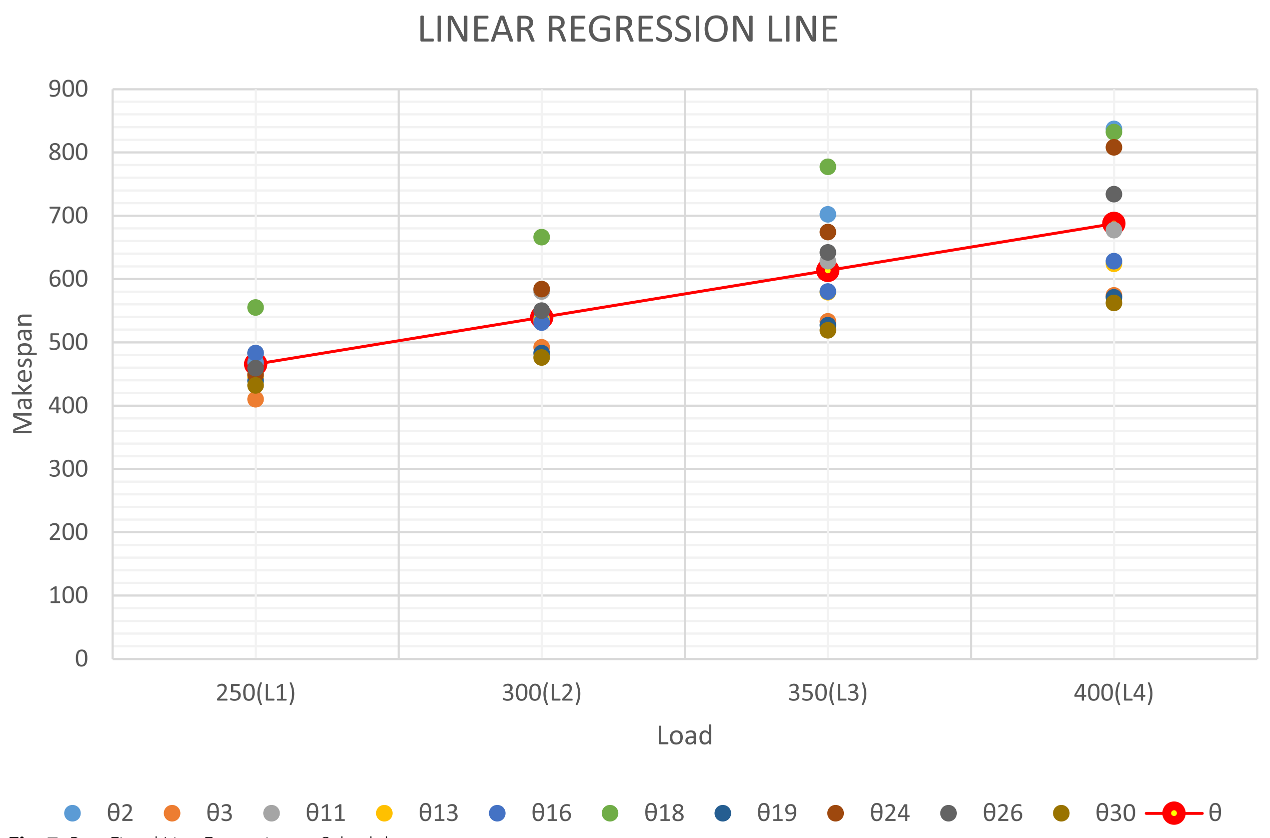


Fig. 7 Best Fitted Line For optimum Schedule

Table 6 Linear regression table

Linear regression table				
X	**Y**	**Y′**	**Y-Y′**	**(Y-Y′)^2**
250	462.3	465.6	−3.3	10.89
300	543.3	539.6	3.7	13.69
350	616.1	613.6	2.5	6.25
400	684.7	687.6	−2.9	8.41

Table 7 Load vs. minimum Makespan table

	Makespan According to the best-fitted line
250(L1)	465.6
300(L2)	539.6
350(L3)	613.6
400(L4)	687.6

forecasting is the benchmark for obtaining the best-fitted line. A regression line is given by the below Eq. 5:

$$Y' = bX + A \tag{5}$$

The predicted value (Y′) is the sum of the intercept of Yi.e.A and the bX where b is the slope of the regression line. Table 7 depicts the minimum Makespan corresponding to the workload as per the best-fitted Line.

The proposed method mainly focuses on the Makespan parameter. In the Future Other Factors like Security, efficiency, task priority, and energy consumption must be taken as well to enhance the overall performance in the cloud environment. This approach works within a min-max range of virtual machine configurations, cloudlets, and data centers. Any deviation from this range and workload above the threshold need to be explored in the future. Table 8 discussed the Comparison of the proposed approach with the other existing scheduling methods.

Conclusion

A cloud service provider's Platform has heterogeneous infrastructure from a variety of cloud users and through virtualization, a large number of cloudlets are scheduled on these limited number of resources in such a manner that each cloud user gets the minimum delay. A low-overhead-based scheduling scheme, based on the Ordinal Optimization modeling technique, is being proposed in this work.

A testbed for the candidate schedules was designed for applying and testing the proposed approach. This includes creating various data centers, cloudlets, and virtual machines along with the scheduling policies for the cloudlets and virtual machines so that a realistic cloud environment could be set up to schedule the tasks and analyze the results. The varying workloads are then mapped onto the optimal schedules, which were obtained after applying the Ordinal optimization modeling technique, to generate the desired makespan. Subsequently, the Linear regression technique is applied to these schedules to predict the future scheduling of the cloudlets for obtaining the minimum possible makespan for a given set of available optimal schedules. In the future, the proposed technique can be further implemented with other parameters like Security, efficiency, task priority, and energy consumption as well to enhance the overall performance in the cloud environment.

Table 8 Comparison of task scheduling methods

Scheduling Method	**Strength and Advantages**	**Disadvantages or Limitations**
Monte Carlo Simulation Method	High precision to get the best schedule. The Monte Carlo method reduces the memory requirements of the fixed short scheduling period, resulting in high system throughput.	High simulation work with exhaustive searches for optimization. This method does not make the adapt to sudden changes in workload. Longer planning horizons degrade performance.
Blind Pick Scheduling Method	With moderate overhead, this method applies a reduced search space and can somewhat adapt to rapid workload fluctuations.	It has moderate accuracy because it has less overhead. With a bad selection set, the performance drops in Monte Carlo.
Ordinal Optimization (Proposed) Method	With very little overhead, OO can adapt to fast workload fluctuations and run suboptimal schedules with high multitasking throughput and reduced memory footprint.	The suboptimal schedule generated at each period may not be as optimal as the schedule generated by the Monte Carlo method. A high noise level can degrade the schedule generated by OO.

Acknowledgments
The authors would like to thank the anonymous reviewers for their insightful comments and suggestions on improving this paper.

Authors' contributions
Monika Yadav reviewed the state of the art in the field, proposed the method, implemented algorithms, and analyzed the results. Atul Mishra supervised this research, lead and approved its scientific contribution, provided general input, reviewed the article, and issued his approval for the final version. The author(s) read and approved the final manuscript.

Funding
The authors declare that they have no funder.

Availability of data and materials
The data used during the current study are available from the corresponding author upon reasonable request.

Declarations

Ethics approval and consent to participate
The work is a novel work and has not been published elsewhere nor is it currently under review for publication elsewhere.

Consent for publication
Informed consent was obtained from all individual participants included in the study.

Competing interests
The authors declare that they have no competing interests.

Received: 19 April 2022 Accepted: 8 January 2023
Published online: 17 January 2023

References

1. Delias P, Doulamis AD, Doulamis ND, Matsatsinis N (2011) Optimizing resource conflicts in workflow management systems. IEEE Trans Knowl Data Eng 23(3):417–432. https://doi.org/10.1109/TKDE.2010.113
2. Hanani A, Rahmani AM, Sahafi A (2017) A multi-parameter scheduling method of dynamic workloads for big data calculation in cloud computing. J Supercomput 73(11):4796–4822. https://doi.org/10.1007/s11227-017-2050-6
3. Tziritas N, Xu CZ, Loukopoulos T, Khan SU, Yu Z (2013) Application-aware workload consolidation to minimize both energy consumption and network load in cloud environments. In: Proceedings of the international conference on parallel processing, pp 449–457. https://doi.org/10.1109/ICPP.2013.54
4. Yadav M, Poongodi T (2020) 5. Federated cloud service management and IoT. In: Internet of things, 1st edn. De Gruyter, p 101. https://doi.org/10.1515/9783110628517-005
5. Sandhu AK (2022) Big data with cloud computing: discussions and challenges. Big Data Min Anal 5(1). https://doi.org/10.26599/BDMA.2021.9020016
6. Deelman E, Singh G, Livny M, Berriman B, Good J (2008) The cost of doing science on the cloud: the montage example. In: 2008 SC - international conference for high performance computing, networking, storage and analysis, SC 2008. https://doi.org/10.1109/SC.2008.5217932
7. Hoffa C et al (2008) On the use of cloud computing for scientific workflows. In: Proceedings - 4th IEEE international conference on eScience, eScience 2008, pp 640–645. https://doi.org/10.1109/eScience.2008.167
8. Yadav M, Breja M (2021) Genre-based recommendation on community cloud using Apriori algorithm. In: Prateek M, Singh TP, Choudhury T, Pandey HM, Gia Nhu N (eds) Proceedings of international conference on machine intelligence and data science applications: MIDAS 2020. Springer Singapore, Singapore, pp 139–151. https://doi.org/10.1007/978-981-33-4087-9
9. Malik SUR, Khan SU, Srinivasan SK (2013) Modeling and analysis of state-of-the-art VM-based cloud management platforms. IEEE Trans Cloud Comput 1(1):50–63. https://doi.org/10.1109/TCC.2013.3
10. Somasundaram TS, Govindarajan K (2014) CLOUDRB: a framework for scheduling and managing High-Performance Computing (HPC) applications in science cloud. Futur Gener Comput Syst 34:47–65. https://doi.org/10.1016/j.future.2013.12.024
11. Yadav M, Breja M (2020) Secure DNA and Morse code based profile access control models for cloud computing environment. Procedia Comput Sci 167(2019):2590–2598. https://doi.org/10.1016/j.procs.2020.03.317
12. Kim D, Son J, Seo D, Kim Y, Kim H, Seo JT (2020) A novel transparent and auditable fog-assisted cloud storage with compensation mechanism. Tsinghua Sci Technol 25(1):28–43. https://doi.org/10.26599/TST.2019.9010025
13. Doǧan A, Özgüner F (2005) Biobjective scheduling algorithms for execution time-reliability trade-off in heterogeneous computing systems. Comput J 48(3):300–314. https://doi.org/10.1093/comjnl/bxh086
14. Smith J, Siegel HJ, Maciejewski AA (2008) A stochastic model for robust resource allocation in heterogeneous parallel and distributed computing systems. In: IPDPS Miami 2008 - proceedings of the 22nd IEEE international parallel and distributed processing symposium, program and CD-ROM. https://doi.org/10.1109/IPDPS.2008.4536431
15. Nath S, Wu J (2020) Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems. Intell Converged Netw 1(2):181–198. https://doi.org/10.23919/icn.2020.0014
16. Yu J, Buyya R (2006) A budget constrained scheduling of workflow applications on utility grids using genetic algorithms. In: 2006 workshop on workflows in support of large-scale science, WORKS'06, vol 14, pp 217–230. https://doi.org/10.1109/WORKS.2006.5282330
17. Zhang W, Chen X, Jiang J (2021) A multi-objective optimization method of initial virtual machine fault-tolerant placement for star topological data centers of cloud systems. Tsinghua Sci Technol 26(1):95–111. https://doi.org/10.26599/TST.2019.9010044
18. Benoit A, Marchal L, Pineau JF, Robert Y, Vivien F (2009) Resource-aware allocation strategies for divisible loads on large-scale systems. In: IPDPS 2009 - proceedings of the 2009 IEEE international parallel and distributed processing symposium, pp 2–5. https://doi.org/10.1109/IPDPS.2009.5160912
19. Gawali MB, Shinde SK (2017) Standard deviation based modified cuckoo optimization algorithm for task scheduling to efficient resource allocation in cloud computing. J Adv Inf Technol 8(4):210–218. https://doi.org/10.12720/jait.8.4.210-218
20. Buyya R, Ranjan R, Calheiros RN (2010) InterCloud: utility-oriented federation of cloud computing environments for scaling of application services. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), vol 6081 LNCS, no. PART 1, pp 13–31. https://doi.org/10.1007/978-3-642-13119-6_2
21. Lu K, Zomaya AY (2007) A hybrid policy for job scheduling and load balancing in heterogeneous computational grids. In: Sixth international symposium on parallel and distributed computing, ISPDC 2007. https://doi.org/10.1109/ISPDC.2007.4
22. Van Den Bossche R, Vanmechelen K, Broeckhove J (2010) Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads. In: Proceedings - 2010 IEEE 3rd international conference on cloud computing, CLOUD 2010, pp 228–235. https://doi.org/10.1109/CLOUD.2010.58
23. Bertot L, Genaud S, Gossa J (2018) An overview of cloud simulation enhancement using the Monte-Carlo method. In: Proceedings - 18th IEEE/ACM international symposium on cluster, cloud and grid computing, CCGRID 2018, pp 386–387. https://doi.org/10.1109/CCGRID.2018.00064
24. Zhang F, Cao J, Hwang K, Li K, Khan SU (2015) Adaptive workflow scheduling on cloud computing platforms with iterativeordinal optimization. IEEE Trans Cloud Comput 3(2):156–168. https://doi.org/10.1109/TCC.2014.2350490
25. Zhang F, Cao J, Tan W, Khan SU, Li K, Zomaya AY (2014) Evolutionary scheduling of dynamic multitasking workloads for big-data analytics in elastic cloud. IEEE Trans Emerg Top Comput 2(3):338–351. https://doi.org/10.1109/TETC.2014.2348196

26. Gawali MB, Shinde SK (2018) Task scheduling and resource allocation in cloud computing using a heuristic approach. J Cloud Comput 7(1). https://doi.org/10.1186/s13677-018-0105-8
27. Gu J, Hu J, Zhao T, Sun G (2012) A new resource scheduling strategy based on genetic algorithm in cloud computing environment. J Comput 7(1):42–52. https://doi.org/10.4304/jcp.7.1.42-52
28. Topcuoglu H, Hariri S, Wu MY (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst 13(3):260–274. https://doi.org/10.1109/71.993206
29. Zhu Z, Zhang G, Li M, Liu X (2016) Evolutionary multi-objective workflow scheduling in cloud. IEEE Trans Parallel Distrib Syst 27(5):1344–1357. https://doi.org/10.1109/TPDS.2015.2446459
30. Ho YC (1999) An explanation of ordinal optimization: soft computing for hard problems. Inf Sci 113(3–4):169–192. https://doi.org/10.1016/S0020-0255(98)10056-7
31. Malawski M, Figiela K, Bubak M, Deelman E, Nabrzyski J (2015) Scheduling multilevel deadline-constrained scientific workflows on clouds based on cost optimization. Sci Program 2015. https://doi.org/10.1155/2015/680271
32. Edward Lau TW, Ho YC (1997) Universal alignment probabilities and subset selection for ordinal optimization. J Optim Theory Appl 93(3):455–489. https://doi.org/10.1023/a:1022614327007
33. Goyal T, Singh A, Agrawa A (2012) Cloudsim: simulator for cloud computing infrastructure and modeling. Procedia Eng 38:3566–3572. https://doi.org/10.1016/j.proeng.2012.06.412
34. Yadav M, Mishra A, Balusamy B (2020) Design of candidate schedules for applying iterative ordinal optimisation for scheduling technique on cloud computing platform. Int J Internet Manuf Serv 7(1–2):5–19. https://doi.org/10.1504/IJIMS.2020.105027
35. Hu Y et al (2000) Screening of optimal structure among large-scale multi-state weighted k-out-of-n systems considering reliability evaluation. Ann Oper Res 206(1–4):107268. https://doi.org/10.1016/j.ress.2020.107268

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.