Cluster Computing (2019) 22:1379–1396
https://doi.org/10.1007/s10586-019-02915-3

TOPSIS–PSO inspired non-preemptive tasks scheduling algorithm in cloud environment

Neelam Panwar[1] · Sarita Negi[2] · Man Mohan Singh Rauthan[1] · Kunwar Singh Vaisla[3]

Received: 25 August 2018 / Revised: 23 January 2019 / Accepted: 31 January 2019 / Published online: 12 February 2019

Abstract
Cloud computing is an emerging paradigm that offers various services for both users and enterprisers. Scheduling of user tasks among data centers, host and virtual machines (VMs) becomes challenging issues in cloud due to involvement of vast number of users. To address such issues, a new multi-criteria approach i.e., technique of order precedence by similarity to ideal solution (TOPSIS) algorithm is introduced to perform task scheduling in cloud systems. The task scheduling is performed in two phases. In first phase, TOPSIS algorithm is applied to obtain the relative closeness of tasks with respect to selected scheduling criteria (i.e., execution time, transmission time and cost). In second phase the particle swarm optimization (PSO) begins with computing relative closeness of the given three criteria for all tasks in all VMs. A weighted sum of execution time, transmission time and cost used as an objective function by TOPSIS to solve the problem of multi-objective task scheduling in cloud environment. The simulation work has been done in CloudSim. The performance of proposed work has been compared with PSO, dynamic PSO (DPSO), ABC, IABC and FUGE algorithms on the basis of MakeSpan, transmission time, cost and resource utilization. Experimental results show approximate 75% improvement on average utilization of resources than PSO. Processing cost of TOPSIS–PSO reduced at approximate 23.93% and 55.49% than IABC and ABC respectively. The analysis also shows that TOPSIS–PSO algorithm reduces 3.1, 29.1 and 14.4% MakeSpan than FUGE, ant colony optimization (ACO) and multiple ACO respectively. Plotted graphs and calculated values show that the proposed work is very innovative and effective for task scheduling. This TOPSIS method to calculate relative closeness for PSO has been remarkable.

Keywords Cloud computing · Task scheduling · TOPSIS · PSO · Relative closeness

1 Introduction

Cloud computing is a promising technology, where numerous services are delivered to users over the web with instant response and scalability features. In cloud computing, virtualization works as key enabler which is defined as the process of segregating the resources of a physical machine (PM) to enable more than one execution environment. In virtualization, various concepts such as time-sharing, machine simulation and emulation are used to virtualize computing, storage, network and memory. It enables multi-tenancy concept in cloud computing where virtualized resources are pooled to serve multiple users by same PM as shown in Fig. 1. The virtualization consists of a hypervisor or a VM monitor which presents a virtual operating platform to a guest operating system. VMW are ESX/ESXi, Oracle VM Server, Citrix XenServer, Xen

✉ Neelam Panwar
neelam.panwar001@gmail.com

Sarita Negi
sarita.negi158@gmail.com

Man Mohan Singh Rauthan
mms_rauthan@rediffmail.com

Kunwar Singh Vaisla
vaislaks@rediffmail.com

[1] Computer Science and Engineering, HNBGU, Srinagar, Pauri, Uttarakhand, India

[2] Computer Science and Engineering, UTU, Dehra Dun, Uttarakhand, India

[3] Computer Science and Engineering, KEC, Dwarhat, Almora, Uttarakhand, India

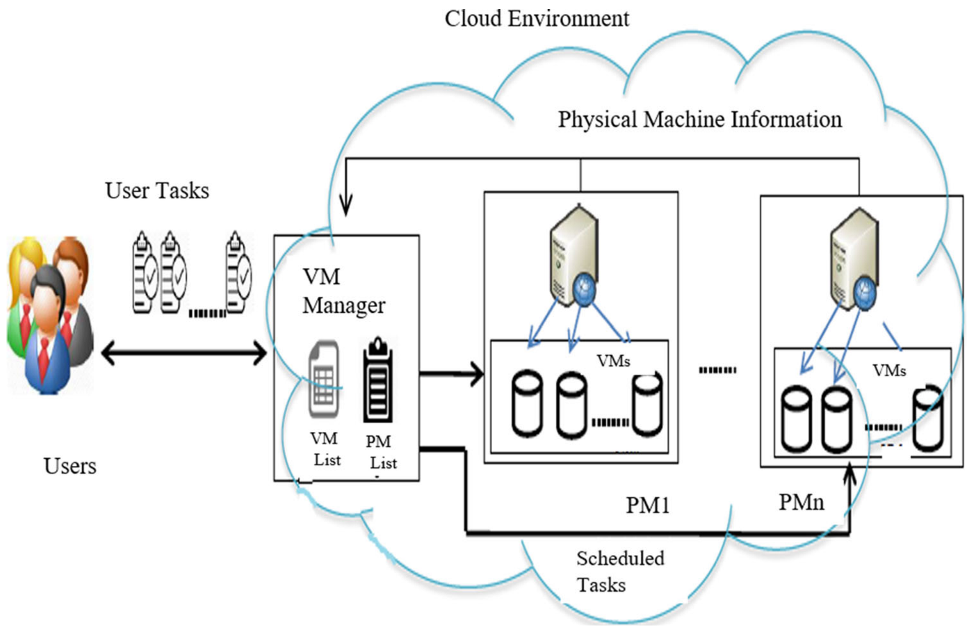

Fig. 1 Architecture of cloud environment

hypervisor and KVM are some popular hypervisors. Virtualization, load balancing, fault tolerance, security and scheduling are the major concerns to solve. There are two main functions of scheduling, i.e., allocation of CPUs to virtual machine (VM scheduler) and submission of task to VM (task scheduler). Scheduling of tasks is one of the major challenges in cloud technology to achieve highly efficient computations among the machines [1–3].

The procedure of searching the needed resources is same as the procedure of searching the various VMs, as the needed resources together form VMs. Users send requests to datacenter to execute their tasks. A task may involve entering and processing data, accessing software or some storage functions. Each task on cloud get executed by VMs of the cloud. The execution of these tasks depends on the mapping of task to VMs. The mapping of task to VMs is important to achieve best Quality of Service (QoS) of cloud [3].

Many researchers are interested in task scheduling practice to achieve load balancing, energy efficiency, resource utilization, migration of tasks and QoS [4]. Tasks can either be independent, dependent or the combination of both [5]. The nature of task scheduling is a NP-hard problem that comprises of various tasks and machines. These tasks must be processed with the machines (VMs) to achieve better utilization. Tasks are submitted to the task scheduler. It is the responsibility of task scheduler to submit task to appropriate VM. The problem of scheduling of tasks need to be properly handle by assigning suitable VMs. The mapping between task and required resources is performed successfully if cloud has achieved minimum MakeSpan, minimum execution time of tasks and VMs, full utilization of resources, etc. further, the task must be executed and a reply is sent to the user. The role of task scheduling is still in research to obtain the minimum execution time of VMs.

To solve the nature of NP-hard problem, an optimization technique such as, particle swarm optimization (PSO) is more suitable than the use of deterministic algorithms [6]. The parallel nature working of PSO resolves the optimization problems. Recently, intelligent meta-heuristic i.e., artificial neural network, fuzzy logic, genetic algorithms (GAs) and mathematical approaches such as distribution functions has been increased in use to achieve drastic solutions for scheduling. Researchers have built new concept of hybridization of these intelligent and mathematical approaches. One of the examples of this hybridization is FUGE which is a meta-heuristic method to enhance job scheduling [7]. Another hybridized scheduling PSO–GELs algorithm is introduced to solve the scheduling problems.

Recently, much attention has been received from researchers in evaluating the multi-objective based environment to solve the generic optimization problems [8–10]. A combination of multi-objective PSO (MOPSO) and technique of order preference by similarity to ideal solution (TOPSIS) is introduced to solve multi-objective inventory planning [8]. The idea of the work was so popular and reported better outcome for the inventory planning problems. The multi-objective methods or multi-criteria based decision making methods are developed to the solutions of real-world problems [9]. The nature of real based decision criteria removes all the barriers of linear problems. Nelson Jayakumar and Venkatesh [10] has proposed a hybrid method of GSO with TOPSIS (glowworm swarm optimization with technique for order preference similar to an ideal solution) for economic dispatch problem. A self adaptive learning PSO (SLPSO) algorithm works on multi objective task scheduling algorithm to map tasks to VMs in order to advance throughput of the datacenter [13]. Authors of [13] have proposed a self-motivated task scheduling. Scheduling of independent tasks over the cloud achieved through dynamic PSO (DAPSO) and Cuckoo search algorithm named MDAPSO [14].With shorten average operation time of tasks a proper resource allocation to user task assigned efficiently in the environment with increases utilization ratio of resources [15]. Live migration and non-live migration approach for scheduling is much efficient to perform migrated tasks on appropriate VMs [16]. This leads to achieve load-balancing in heavy cloud servers.

The above discussed works improve task scheduling however, task scheduling is still major concern in cloud environment. Thus it challenges to design novel algorithms for task scheduling in cloud environment. This paper presents the combination of optimization algorithm PSO with a mathematical multi-criteria decision based approach called TOPSIS (TOPSIS–PSO) to solve the task scheduling problems. PSO is the simplest optimization technique which works well for global optimization problem. Because of its less efficiency of finding optimal solutions for local optimum, TOPSIS is used. Our proposed approach, TOPSIS–PSO, uses TOPSIS for the calculation of optimized fitness value (FV). Hence the role of TOPSIS is a fitness evaluation tool. In this work, three major criteria of a cloud task are used i.e., execution time, transmission time and cost. The evaluated FV of each task is input to the PSO for further optimization of particles. Results of TOPSIS–PSO are compared with other leading approaches that are implemented in real application based environment. The combination of PSO and TOPSIS improves the PSO's quality of finding optimum solutions and gives solutions to nonlinear bi-level programming problems [11]. The proposed idea is novel and innovative in the cloud computing field. The use of TOPSIS–PSO in cloud can improve the MakeSpan, resource utilization, processing cost. It can be applied to a broad variety of optimization problems.

The key contributions of this paper are summarized as,

- Efficient task scheduling method is presented in order to schedule tasks to VMs in cloud environment.
- Multi-criteria based method called TOPSIS algorithm is implemented with PSO to get efficient optimal solutions to perform task scheduling. These objectives include the enhancement on cloud metrics i.e., MakeSpan, execution time, transmission time and processing cost.

The performance of work is compared with pre-existing algorithms to check its reliability. The ultimate objective of the work is to enhance the QoS of the cloud performance.

The rest of the paper is organized as follows: Sect. 2 discusses previous related work on cloud based task scheduling. Problem formulation of the work is discussed in Sects. 3 and 4 highlights the system model of the proposed work. Section 5 elaborates proposed task scheduling algorithm with implementation while Sect. 6 deals with experimental evaluation of proposed work. In Sect. 7, we conclude our contributions with future scopes to the work.

2 Related work

Numerous researchers worked in task scheduling to get better, effective and reliable work on cloud computing. These researches differ from each other in their methods to schedule tasks among cloud nodes. These methods fall under different categories: static, dynamic, batch and online which endeavor to assign tasks to cloud nodes in optimal way. The work of [8, 10, 11] motivates to perform in cloud computing technology because combination of some optimization technique and multi-objective methods give relevant solutions for decision making of criteria in cloud environment. Through this section various related historic work are discussed in detail.

In Pooranian et al. [6], meta-heuristic algorithm introduced in grid computing area. The PSO with gravitational emulation local search (PSO–GELs) is a hybrid task scheduling algorithm that focused the problem of independent task scheduling problems. The hybrid algorithm reduces MakeSpan and decreases deadline miss rate of tasks.

Shojafar et al. [7] attempts a hybrid approach FUGE which is the combination of meta-heuristic method called fuzzy theory and GA to cloud job scheduling. They have mathematically proven optimization problem which is convex with Karush–Kuhn–Tucker condition. Length of job, Processing speed of VM, bandwidth of VM and memory of VM have been considered for assigning the jobs to the resources. This approach obtains a proper efficiency over degree of imbalance, execution time and execution cost.

Tsou [8], gives the idea of hybridization of MOPSO and TOPSIS for solving multi-objective optimization problems in inventory planning. MOPSO was used to generate the non-dominated solutions of a reorder point where as TOPSIS performed a compromise solution for different decision makers.

The method in Nelson Jayakumar and Venkatesh [10] uses GSO algorithm to find the optimal solution for multiple objective environmental economic dispatches problem. Although the work is not related to the technology field but it gives the motivational idea to use the concept of TOPSIS for solving multi-criteria based problems. Here TOPSIS is used as an overall fitness ranking tool to evaluate multiple objectives.

In Jia et al. [11], a solution of non-linear bi-level programming problems has presented with the combination of TOPSIS and PSO. The literature has thoroughly explained the non-linear bi-level programming problems which occurs in various optimization techniques. This work has motivated and inspired to understand the optimization problem techniques and to opt such solutions in cloud task scheduling approach.

Wang et al. [12], presented a framework which allows IaaS provider to outsource its tasks to outside cloud when it does not have sufficient resources to fulfill task requirements. The task scheduling is carried out by SLPSO. In SLPSO four update approaches are used to flexibly update the velocity of particles (tasks). At each iteration in PSO, update approach should be found based on execution probability. To obtain the best sequence of execution, a particle is updated by four approaches and best approach is preferred for other particles. From PSO algorithm the best particle is assigned with its best VM. In this algorithm, frequent selection of update strategy leads to computational complexity and also increases the scheduling time.

Zhang and Zhou [13], proposed a self-motivated task scheduling with two-stage strategy to maximize scheduling performance. Initially the user tasks are kept in task queue. In the first stage of the algorithm, Bayes classifier classifies the jobs on the basis of historical data stored in historical database. In second stage, dynamic scheduling algorithm is used to map tasks to suitable VMs. After selecting suitable VM the tasks are sent to ready queue, if the suitable VM is not idle then the task is moved to waiting queue. When the suitable VM becomes idle the task needs to move into ready queue and the task is scheduled to suitable VM. Multiple queues and databases used in this method increase the space complexity.

Awad et al. [17], proposed load balancing mutation, a PSO method for task scheduling using reliability, make span, execution time, transmission time and round trip time. Three mathematical models which have individual objective functions are proposed for task scheduling. In first model expected execution time (EET) for each task on each VM is calculated. In second model expected transmission time (ETT) for each task is computed and in third model expected round trip time (ERRT) is computed using EET and ETT. The process of scheduling tasks to VMs is carried out by PSO algorithm. In PSO algorithm, the velocity of particle is updated for each iteration using ERRT. This method considers MakeSpan but ignores other important parameters such as processing cost and resource utilization.

Lakraa and Yadav [18], proposed multi objective task scheduling algorithm for mapping tasks to VM in order to improve throughput of the datacenter. In this method QoS requirement for all tasks are found and the high QoS value is given to the task which requires low QoS by cloud broker. Hence the task with lower QoS value gets high priority for scheduling. Cloud broker also collects the list of VMs with their MIPS values and the VMs are sorted according to MIPS value. Then the first VM in VMs list is assigned to first task in task list. Once the allocation reached at last VM the next task will be submitted to the first VM and the process of allocation is repeated for all tasks. In this method the VMs are ranked based on MIPS value only and other parameters such as memory and bandwidth are not considered.

Cho et al. [19], combined ACO and PSO algorithms to build ant colony optimization with PSO (ACOPS) for task scheduling. The new workload is predicted by ACOPS using historical information. To reduce scheduling time pre-reject module is proposed in the work. When task arrives initially the algorithm checks for remaining memory of each server and find maximum remaining memory. If the memory requirement of task is larger than remaining memory then the task is rejected. The suitable tasks enter into ACOPS scheduling algorithm and the initial process is executed using ACO algorithm. In ACO, the search module gives the solution for all ants (tasks) and the pheromone function of tasks is computed using memory, CPU utilization and disk utilization. PSO operator is applied after search module in ACO to improve the search results. The pheromone is updated by an ant selected based on PSO operator i.e., global best value is selected. This algorithm considers memory, CPU utilization and disk utilization but does not consider MakeSpan, bandwidth, execution time, etc.

Jena [20], introduced task scheduling algorithm using a multi-objective nested PSO (MOPSO) to minimize energy consumption and MakeSpan. Some concepts of the evolutionary algorithms (EAs) and the multi-objective EAs have been collaborated to introduce the MOPSO. Ghanbari and Othman [21], proposed a new priority based job scheduling algorithm which was based on multiple criteria decision making (MCDM) model using analytical hierarchy process. MCDM is a practical model which is functionally correlated with problems of discrete alternatives.

Lawrance and Silas [22], introduced potentially all pairwise rankings of all possible alternatives (PAPRIKA) based scheduling algorithm which is also a MCDM model. Shih et al. [23] introduced MCDM based approach for solving real world decision making problems. The role of decision matrix (DM) is to provide different course of action to finite numbers such as select, prioritize, and rank.

Table 1 Comparison of task scheduling algorithms

References	Work highlights	Methodology	Environment	Benefits	Tribulations
[13]	Multi-objective tasks scheduling algorithm	Uses non-dominated sorting to solve multi-objective problem	Dynamic	Less execution time and high throughput	Starvation may occur for the tasks having low priority. Bandwidth not considered
[13]	Two-stage strategy for dynamic cloud task scheduling	A two-stage strategy is proposed for dynamic cloud scheduling to maximize scheduling performance	Dynamic	Failure rate is reduced. work guarantees high priority ratios of ordinary tasks than its peers do	Multiple queues and databases used in this method which increase the space complexity
[17]	Enhanced particle swarm optimization for task scheduling	A mathematical model is introduced using load balancing mutation (balancing) a particle swarm optimization (LBMPSO). Balances task and VM	Dynamic	The work enhances reliability, execution time, transmission time and cost, MakeSpan, round trip time	Algorithm is unsuitable for dependent and heterogeneous tasks. MakeSpan increases due to large number of iteration in PSO
[18]	Enhanced load-balancing min–min for static meta-task scheduling	ELBMM is based on min–min technique and use task rescheduling to achieve resource utilization. Task with minimum completion time assigns to suitable resource	Static	Improves resource utilization	Two scheduling processes lead to increase MakeSpan time
[20]	Multi objective tasks scheduling algorithm (*TSPSO*)	Uses multi-objective optimization model called multi objective *PSO* based framework	Dynamic	Less energy consumption and MakeSpan. Reduces failed tasks	Resource utilization need to be increase. Not suitable for independent task
[24]	Improved cost-based algorithm	A job grouping algorithm that takes the priority levels (high, medium and low) of tasks	Static	Less processing cost	Not suitable for dynamic cloud environment and dependent tasks
[25]	Bandwidth aware divisible task scheduling (*BATS*)	Divisible task scheduling problem using non-linear programming model	Dynamic	Reduces MakeSpan, improves utilization ratio	Increases cost and time to solve programming model
[26]	Parallel task scheduling based on fuzzy clustering	Dynamic scheduling of tasks conducts the clustering solution of concurrent jobs	Dynamic	Performs dependent tasks and obtains higher efficiency	Dependency of tasks may require the fault tolerance schemes

These actions make convenient to DM attributes for making decisions owing to an explicitly represented procedure. TOPSIS performed the concept of distance measures of the alternatives from the positive ideal solution (PIS) and the negative ideal solution (NIS).It is the most straight-forward technique in MCDM. TOPSIS has been an important branch of DM.

Table 1 shows the comparative review of [13, 13, 17, 18, 20, 24–26] which shows that most of the previous researches have focused on single objective only. Therefore, to deal with these gaps, a MOPSO algorithm using TOPSIS (TOPSIS–PSO) is introduced to optimize MakeSpan, execution time, transmission time and to reduce cost.

3 Problem formulation

The main objective of task scheduling algorithms is to bind set of user tasks to set of distributed resources in order to achieve several goals including minimizing MakeSpan, execution time, transmission time, cost and maximizing resource utilization. Single objective scheduling algorithms face some problems that include increased execution time and decreased throughput in priority based algorithms. Similarly, shortest job first and first come first serve algorithms perform very well in best case scenario but performance is degraded in worst case. So an efficient scheduling algorithm is required which focuses on multi objectives. Using a proper scheduling algorithm implementation in broker improves the datacenter's performance [13]. In this paper following objective problems are considered for developing efficient scheduling algorithm.

Notations	Description
VM	Virtual machine
PM	Physical machine
DC	Data center
T_i	ith number of task
CT_j	Completion time of jth VM
PC_T	The cost of using processing in current resource
$Size_i$	Size of T_i
BW_j	Bandwidth of jth VM
CT_j	Completion time of jth VM
AVG_U	Average utilization of CPU
L_i	Length of T_i
MIPS	Million instructions per second (processing speed)
RC	Relative closeness
FV	Fitness value
W_j,	Weight value for criteria
EX_T, $Trans_T$, AVG_U	Execution time, transmission time, CPU utilization

3.1 Execution time (EX_T)

The time required to execute a task on particular VM is called execution time. The execution time of task T_i on VM V_j formulated as follows [26]:

$$EX_T = \frac{L_i}{MIPS_j}. \tag{1}$$

EX_T is computed by considering L_i in terms of number of million instructions and computing capacity of jth VM as MIPS. Task i is assigned to VM which minimizes EX_T.

3.2 Transmission time (delay) ($Trans_T$)

The time taken to transfer the task on a particular VM is called transmission time and is computed as [16]:

$$Trans_T = \frac{Size_i}{BW_j}. \tag{2}$$

$Trans_T$ for a task on particular VM is determined by task $Size_i$ and BW_j.

3.3 Processing cost (PC_T)

The cost of processing a task on a resource is calculated by:

$$PC_T = \text{Cost of processing per unit time} * EX_T. \tag{3}$$

3.4 MakeSpan

MakeSpan can be defined as overall time taken to perform scheduling process. As MakeSpan reduces, it improves the efficiency of the algorithm. The MakeSpan of task scheduling is calculated as [16]:

$$\text{MakeSpan} = \max(CT_j), \tag{4}$$

where $CT_j = start(VM_j) + \tau_j$, τ_j denotes the time when all the tasks assigned to jth VM finish their execution and start (VM_j) denotes the time instance when jth VM starts its execution. The maximum completion time is taken as MakeSpan.

3.5 Average resource utilization (AVG_U)

The major challenge in cloud computing is the proper utilization of resources. The AVG_U of proposed work is calculated as [16]:

$$AVG_U = \sum_{j \in VM} \frac{CT_j}{\text{MakeSpan} * \text{number of VMs}}. \tag{5}$$

AVG_U can be defined as the ratio between completion time of VM and the time duration for which CPU performs useful work.

3.6 Formulation of fitness value for PSO

In the proposed work, the combination of minimization objective criteria given in Eqs. (1)–(3) can be optimized. The RC of each task is computed by TOPSIS algorithm. Involvement of TOPSIS algorithm supports multiple objectives in order to maximize efficiency of task scheduling. The obtained RC value is used as FV of tasks for PSO.

FV_{T1}	$= RC_{T1}$
FV_{T2}	$= RC_{T2}$
–	–
–	–
FV_{Ti}	$= RC_{Ti}$

where RC of task computed by TOPSIS is $RC_T = (RC_{T1}, RC_{T2},\ldots,RC_{Ti})$ that are corresponding to the FV of task $FV_T = (FV_{T1}, FV_{T2},\ldots,FV_{Ti})$ respectively. The detailed explanation of TOPSIS formulation for RC is discussed in next section.

4 System model and proposed work

CloudSim (Cloud Simulator) is an extensible modeling and simulation tool of cloud computing systems and application environment. It provides excellent framework to both system and behavior modeling of cloud. CloudSim first introduced by GRIDS Laboratory Melbourne, Australia [25]. It is the most popular open source tool that encompasses various java based packages. Most of the cloud computing terminologies are designed in CloudSim to enable better research environment. The various cloud components have their specific role to perform cloud operations. CloudSim toolkit comprises of various components including Cloud Information Service (CIS), datacenter, host, VMs and tasks as shown in Fig. 2. Each individual VM processes on its own resources in parallel or independently [27]. The simulation work has been performed under CloudSim toolkit.

- *CIS* CIS is the uppermost entity which is created automatically when CloudSim is initialized. It provides services to register, index and discover resources. Different entities interact with each other through this entity.
- *SimEntity* different entities which handle and send events to other entities created by SimEntity.
- *DC* DC can be defined as pool of homogeneous or heterogeneous resources which are virtualized and provided to VMs when required. It handles queries which are related to VM instead of task.
- *Datacenter Broker* (*DCb*) DCb represents a broker which acts on behalf of user. It handles VM management, as creation and destruction of VMs and controls the order in which tasks are submitted to VMs.
- *PM* it executes actions related to management of VMs (e.g., creation and destruction) and defines policies for provision of memory, bandwidth and allocation of processors to VMs.
- *VM* it acts as an emulation of a PM running inside a host and provides all the functionalities of an actual system. Tasks are assigned to VMs through cloudlet scheduler for their execution.
- *VM scheduler* it represents the policy used by the VM monitor (VMM) to share processing capacity among VMs running in a host. It defines the way in which processors will be used by VMs i.e., processors will be shared by VMs or not.
- *Processing element* (*PE*) PE represents processor unit defined in terms of MIPS rating, i.e., the number of million instructions a processor can execute per second.
- *Utilization model* this parameter allows the tasks to make control over resource utilization. If utilization model is set to full, then a task utilizes all the available MIPS of the processor and if set to stochastic, then the task generates random utilization of processor every time span.

4.1 The proposed TOPSIS–PSO based task scheduling algorithm

Here PSO algorithm is combined with TOPSIS algorithm in order to find optimal solution by considering multiple criteria. In PSO algorithm, the fitness function is formulated by TOPSIS algorithm based on multiple criteria. PSO algorithm is better in optimal solution determination with

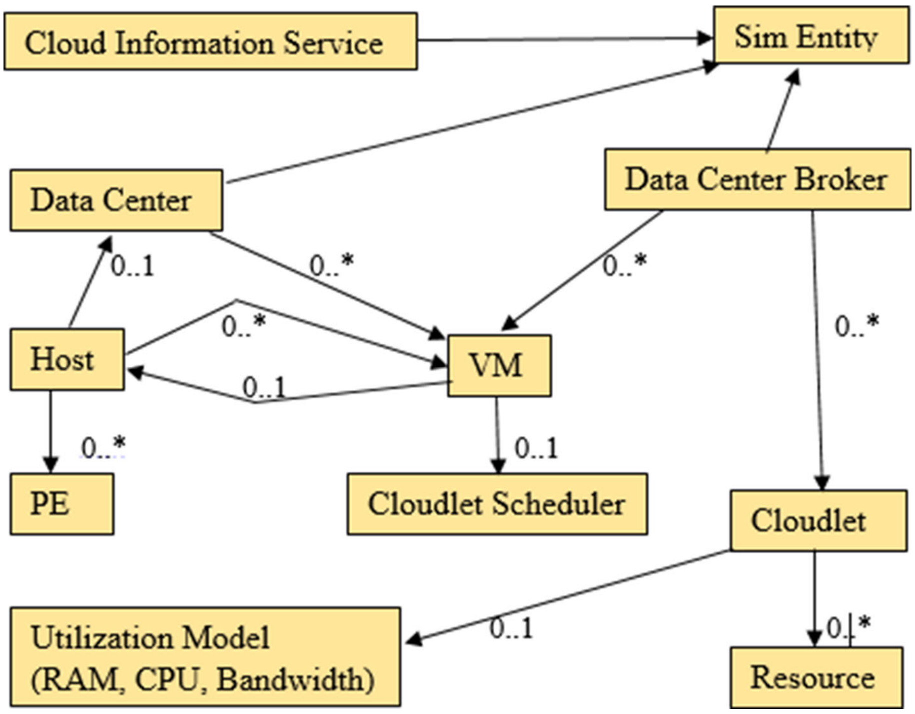

Fig. 2 CloudSim component overview

minimum time consumption. In order to include multiple objectives in PSO, we have used TOPSIS algorithm in PSO. TOPSIS algorithm is also relatively faster than other multi criteria decision making algorithms. Thus proposed hybrid PSO–TOPSIS algorithm helps to achieve optimal solution based on multiple criteria without increasing time consumption. The system model comprises of set of physical machines PM = (PM_1, PM_2,…,PM_M) where each PM holds some VMs = (VM_1, VM_2,…,VM_j). Number of tasks is assigned to each VM respectively to perform the execution of tasks. Each VM runs on its own resources in parallel and independently. Figure 3 depicts the system architectural model of the proposed work. For efficient task scheduling without loss of resource utilization TOPSIS–PSO method performs optimized task scheduling. The task scheduling is carried out by two phases. In first phase, TOPSIS algorithm is applied to obtain RC of the selected cloud criteria (i.e., execution time, transmission time and cost) of a task. In second phase the PSO begins with computed RC of the given three criteria for all tasks in all VM. The working of TOPSIS and PSO is explained in the following section.

4.2 TOPSIS approach

PSO and various heuristic techniques have been used to optimize single criteria based solutions. These heuristic solutions are not up to the mark for multiple criteria based problems. In cloud technology to obtain better optimized solution for multiple criteria, a decision making method can result better ideal solution. This work has grabbed attention on a multi-criteria decision analysis based method called TOPSIS. It was first developed by Hwang and Yoon in 1981. TOPSIS method effects generously on real world decision making problems and work positively for many applications. Task scheduling process is carried out by TOPSIS–PSO algorithm which maps user tasks to VM by considering multiple substantial factors. FV for PSO is computed by TOPSIS algorithm. Involvement of TOPSIS algorithm supports multiple objective functions in order to maximize efficiency of task scheduling. The overall procedure of TOPSIS algorithm is depicted in *Algorithm 1* and steps are as follows:

Step 1 evaluate the DM of size m × n, where m represents number of alternatives and n denotes number of criteria. Alternatives are represented by VMs and criteria are denoted by objective functions as shown in Table 2.

Step 2 this step transforms the dimensional attributes into non-dimensional attributes by comparing against each criterion. The obtained matrix from *Step 1* is standardized using Eq. (6),

$$D_{ij} = \frac{X_{ij}}{\sqrt{\sum X_{ij}^2}}, \tag{6}$$

where j = {1, 2,…,m}, j = {1, 2,…,n} and X_{ij} represents an element in DM corresponding to ith alternative and jth criteria.

Step 3 weight values to each criterion are provided by a function called decision-maker and these weights are provided according to the relevance of the criteria in scheduling process. Each element in normalized DM is multiplied by weight values corresponding to each criterion to generate weighted normalized DM $[W_E,\ W_{trans},\ W_{PC}]$ as,

$$EX_T = EX_{TQ} \times W_E, \tag{7}$$

$$Trans_T = Trans_{TQ} \times W_{trans}, \tag{8}$$

$$PC_T = E_{TQ} \times W_{PC}. \tag{9}$$

Subject to:

$$0 < W_E,\ W_{trans},\ W_{PC} \leq 1\ W_E + W_{trans} + W_{PC}.$$

Step 4 determination of PIS and NIS. PIS (S^+) associated with that value of criteria which has positive impact and NIS (S^-) associated with that value of criteria which has negative impact on the solution are given by,

$$S^+ = \left(V_1^+,\ V_2^+, \ldots, V_i^+\right), \tag{10}$$

$$S^- = \left(V_1^-,\ V_2^-, \ldots, V_i^-\right). \tag{11}$$

Here, EX_T, PC_T and $Trans_T$ are considered as criteria which have positive impact and need to be minimized.

Step 5 calculation of separation measures. In this step, separation of each alternative from PIS and NIS are measured as,

$$S^* = \sqrt{\sum_{j=1}^{3}\left(V_{ij} - V_j^+\right)^2}, \tag{12}$$

$$S' = \sqrt{\sum_{j=1}^{3}\left(V_{ij} - V_j^-\right)^2}. \tag{13}$$

Above two equations are used to measure separation from PIS and NIS respectively, where j denotes the number of criteria.

Step 6 calculation of RC. The RC of Qth task with respect to S^+ is defined as,

$$FV_Q = RC_Q = \frac{S'}{S' + S^*}. \tag{14}$$

RC is taken as FV and updated in PSO algorithm. At each iteration particle move towards best solution in population and update their velocity based on FV computed from TOPSIS algorithm.

```
ALGORITHM 1: TOPSIS to obtain Fitness Value
Input:   m alternatives, n attributes/criteria and weight associated with every criterion.

Output:  Relative closeness or closeness coefficient associated with every alternative.

1. Calculate Decision Matrix.
   DM[ExecutionTime] ←LengthOfTask / MipsOfVM;

   DM[TransmissionTime] ←FileSizeOfTask / BandwidthOfVm;

   DM[ProcessingCost] ←ProcessingCostPerSecond * ExecutionTimeOfTask;

2. Determine Normalized DM.
   for i in alternative

       for j in criteria

           sumPowSqrt[j]← $(\Sigma(DM[i][j])^2)^{1/2}$;

           NDM[j][i] ← DM[j][i]/sumPowSqrt[i];

       end for

   end for

3. Calculate Weighted NormalizedDM.
   for i in alternative

       for j in criteria

           WNDM[i][j] ← NDM[i][j]* WeightCriteria[j];

       end for

   end for

4. Calculate PIS and NIS.
   PIS ← set of benefit attributes i.e. more is better.

   NIS ← set of negative attributes i.e. less is better.

5. Determine the separation measures from PIS and NIS for each alternative.
   for i in alternative

       for j in criteria

           SMPIS[i] ← $(\Sigma_j(PIS[j]- WNDM[i][j])^2)^{1/2}$;

           SMNIS[i] ← $(\Sigma_j(NIS[j]- WNDM[i][j])^2)^{1/2}$;

       end for

   end for

6. Calculate RC.
   for i in alternative

       ClosenessCoefficient[i] ← SMNIS[i]/( SMNIS[i] + SMPIS[i]);

   end for
```

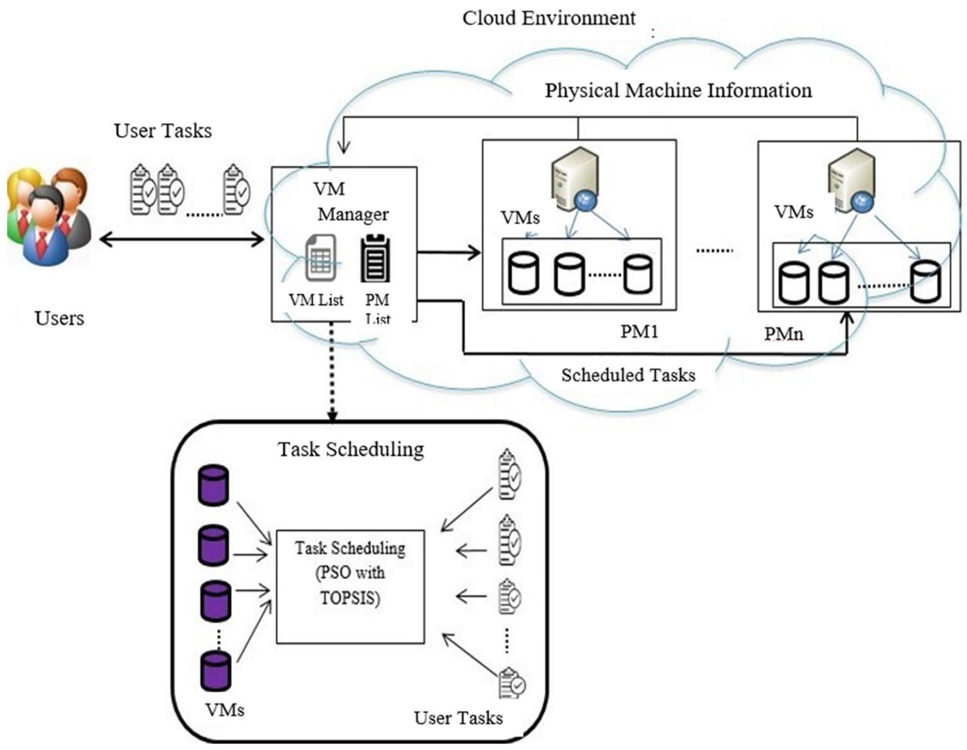

Fig. 3 Framework architecture of proposed method

Table 2 Decision matrix

	EX_T	$Trans_T$	PC
T_1	EX_{T11}	$Trans_{T11}$	PC_{11}
T_2	EX_{T21}	$Trans_{T2i}$	PC_{2i}
–	–	–	–
–	–	–	–
T_Q	EX_{TQ1}	$Trans_{TQ1}$	PC_{Qi}

4.3 PSO approach

Kennedy and Eberhart proposed a newest heuristic approach to perform optimization of solutions known as PSO. In this approach, a simulation is performed to get desired destination of either group of fish or flock of birds which is also known as swarm. Further, the concept of PSO came into the mathematical or computing concept to achieve best optimal solution on the basis of self-adaptive global search. The major part of PSO is initialization of particles. Each particle adjusts its velocity and position according to its best position and the position of best particle i.e., global best of the entire population in each generation. According to the PSO algorithm, each particle is represented using velocity and location which can be obtain using Eqs. (15) and (16).

$$V_P[k+1] = w * V_P[k] + C1 * rand_1 * (pbest - X_P[k]) + C2 * rand_2 * (gbest - X_P[k]), \tag{15}$$

$$X_P[k+1] = X_P[k] + V_P[k+1], \tag{16}$$

where $V_P[k+1]$and $V_P[k]$ represent current velocity and previous velocity of particle p. $X_P[k+1]$ and $X_P[k\ ]$ are the current and previous position of particle p. Two acceleration coefficients C1, C2 and random numbers (between 0 and 1) $rand_1$, $rand_2$ are used in velocity computation. Best position of particle p and position of best particle in the population are denoted by pbest and gbest and w represents inertia weight [25].

PSO is a swarm-based intelligence algorithm [26] inspired by the social behavior of animals such as flocking of birds searching for food. In PSO a particle is similar to a bird moving through a search (problem) space. The velocity is used to synchronize particle movement, which has both magnitude and direction. At any point of time position of each particle is influenced by its best position and the position of the pbest in the search space. FV is problem specific and used to measure the performance of a particle. The population represents the number of particles in the search space. Particles are initialized randomly. Each particle will have a FV which is obtained using TOPSIS. The pbest of a particle is the best result (i.e., FV) reached so far by the particle, whereas gbest is FV of best particle in the search space.

ALGORITHM 2: Particle Swam Optimization with TOPSIS

Input: User tasks list, TList= $\{T_1, T_2, T_3 \ldots\ldots T_n\}$ and list of VMList = $\{VM_1, VM_2, VM_3 \ldots\ldots VM_m\}$

Output: Optimal mapping between TList and VMList.

```
begin
    Set particle dimension ← number of tasks in TList
    Initialize particles randomly (velocity $V_i$, location $X_i$) with pbest and gbest
        for each T ∈ TList
            Construct DM
            t=0
            while (t <epoches)
                Calculate FV (RC) for each particle (using TOPSIS Algorithm 1) and update $X_i$
                    Check if current fitness value less than pbest (particle's local best).
                            Assign pbest ← $X_i$
                    else
                            Keep pbest
                    endif
                Compare all pBest
                Assign gbest ← highest pbest
                    Check if current gbest less than fitness value
                        Assign gbest ← $X_i$
                    else
                        Keep previous gbest
                    endif
                Assign optimal VM to particle with highest gbest
                Assign t to VM
                t++
            end while
        end for
end
```

This process is performed for each user task which is scheduled to VM. By assigning each task to VM, M_T and EX_T are reduced significantly. TOPSIS–PSO based task scheduling process minimizes MakeSpan of user tasks without loss of efficient resource utilization. The working nature of PSO is depicted in *Algorithm 2* and TOPSIS–PSO algorithm based task scheduling is shown in Fig. 4.

5 Implementation

The proposed TOPSIS–PSO algorithms have been implemented with three criterions for enhancing cloud QoS. The algorithm have been simulated in CloudSim tool and executed on 2.20 GHz Intel core Xeon processor with 16 GB RAM memory. For the purpose of comparison, 10 separate runs are made to obtained statistical results. The simulation

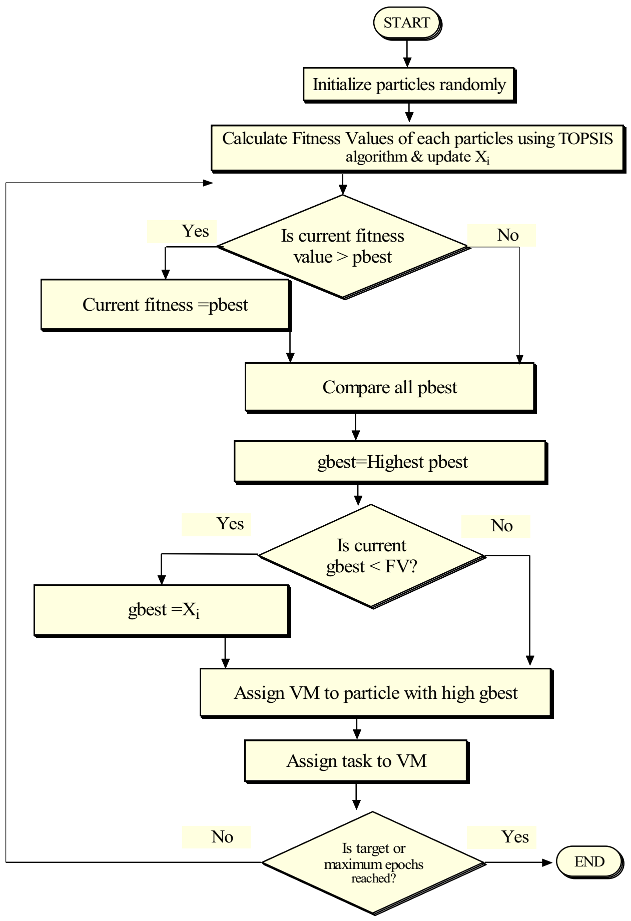
START
Initialize particles randomly
Calculate Fitness Values of each particles using TOPSIS algorithm & update X_i
Yes
Is current fitness value > pbest
No
Current fitness =pbest
Compare all pbest
gbest=Highest pbest
Yes
Is current gbest < FV?
No
gbest =X_i
Assign VM to particle with high gbest
Assign task to VM
No
Is target or maximum epochs reached?
Yes
END

Fig. 4 Flow diagram of TOPSIS–PSO algorithm

work has been performed under CloudSim tool which includes various cloud environment configurations shown in Table 3.

In Table 4, the results obtained for various priority ranges for tasks on VM_1 are discussed. Here T_3 has highest priority on VM_1 since it has minimum transmission time, execution time, and processing cost. Thus TOPSIS–PSO algorithm assigns T_3 to VM_1 for execution. After assigning T_3 to VM_1 same process is repeated for remaining tasks and VMs.

TOPSIS algorithm is used for FV calculation and initialized by estimating relative weight value for each criteria. The weight for all three criteria is represented by $[EX_T, Trans_T, PC] = [W_E, W_{trans}, W_U]$. To explain the abstract working of the TOPSIS we took only 10 tasks assigned to 5 VMs. The working procedure will be the same for higher number of tasks and VMs. Stepwise procedure of TOPSIS for finding RC for each criterion is shown below:

Step 1 evaluation of DM of alternatives (VMs) and criteria (execution time, transmission time and cost) using attributes of tasks under five VMs are shown in Table 5.

Table 3 Type 1 experiment using cloud setup configuration details

Parameters	Values
Tool configuration	No. of VM
System architecture	X86
Operating system	Linux
VMM	Xen
Host description	
RAM	40 GB
Storage	11 TB
Bandwidth	500–2024
No. of PE	5
Utilization model	Full utilization
VM description	
RAM	512
Size (amount of storage)	10,000 (MB)
MIPS	2400
No. of processing unit	4
Task scheduler	Time-shared
Task description	
Number of task	10–80
Length	100–2500
MIPS	300–4000

Step 2 standardization of obtained DM using Eq. (6) shown in Table 6.

Step 3 each element in standardized DM is multiplied by weight value of each criterion to generate weighted standardized DM $[W_E = 0.2, W_{trans} = 0.3, W_{PC} = 0.5]$ using Eqs. (7), (8) and (9) respectively. The weight values are provided by decision maker shown in Table 7.

Step 4 determination of PIS and NIS solution using Eqs. (10) and (11) shown in Table 8.

Step 5 calculation of positive and negative separation measures using Eqs. (12) and (13) respectively is shown in Tables 9 and 10.

Step 6 the positive RC is calculated using Eq. (14). The final RC of each tasks are obtained shown in Table 11.

Step 7 Initialization of particles using Eqs. (15) and (16). Parameters that are used for PSO are shown in Table 12. The steps in the PSO algorithm are listed in *Algorithm 2*. The algorithm starts from initializing the particle dimensions from task list. Velocity and position of particles are initialized randomly using Eqs. (15) and (16). The values allocated to the dimensions of each particle indicate the computing resources that assigned to VM. Thus a particle represents mapping between user tasks and available resources of VMs. In the above given case, each particle has 10 dimensions for 10 numbers of tasks. The criteria are the tasks that are further assigned to available

Table 4 Priority ranges for tasks

Task	Metrics			Priority
	EX_T (s)	$Trans_T$ (s)	PC_T (₹)	
T_1	2	5	0.4	Medium
T_2	5	6	1.0	Low
T_3	1	3	0.2	High

Table 5 Decision matrix

Alternative	Criteria		
	EX_T	$Trans_T$	Cost
T_1	0.06	0.3	0.012
T_2	0.075	0.4	0.015
T_3	0.09	0.5	0.018
T_4	0.105	0.6	0.021
T_5	0.12	0.7	0.024
T_6	0.135	0.8	0.027
T_7	0.155	0.85	0.031
T_8	0.175	0.9	0.035
T_9	0.195	0.95	0.039
T_{10}	0.215	1.00	0.043
D_{ij}	1.325	7.00	0.265

Table 6 Standardize decision matrix

Alternative	Criteria		
	EX_T	$Trans_T$	Cost
T_1	0.1343	0.1288	0.1359
T_2	0.1678	0.1717	0.1699
T_3	0.2014	0.2147	0.2039
T_4	0.235	0.2576	0.2378
T_5	0.2685	0.3005	0.2718
T_6	0.3021	0.3435	0.3058
T_7	0.3468	0.3649	0.3511
T_8	0.3916	0.3864	0.3964
T_9	0.4363	0.4079	0.4417
T_{10}	0.4811	0.4293	0.487

VMs. Table 13 represents the assignment of tasks to the VMs such as, task T_2, T_8 and T_9 are assigned to VM_1, task T_5 and T_7 are assigned to VM_2 and so on. As the number of tasks and VMs are increased the assignment matrix will show the mapping accordingly.

Table 7 Weighted decision matrix

Alternative	Criteria		
	EX_T	$Trans_T$	Cost
T_1	0.0403	0.0515	0.0408
T_2	0.0503	0.0687	0.051
T_3	0.0604	0.0859	0.0612
T_4	0.0705	0.103	0.0713
T_5	0.0806	0.1202	0.0815
T_6	0.0906	0.1374	0.0917
T_7	0.104	0.146	0.1053
T_8	0.1175	0.1546	0.1189
T_9	0.1309	0.1632	0.1325
T_{10}	0.1443	0.1717	0.1461

Table 8 Determining positive and negative ideal solution

Alternative	Criteria		
	EX_T	$Trans_T$	Cost
T_1	0.0403	0.0515	0.0408
T_2	0.0503	0.0687	0.051
T_3	0.0604	0.0859	0.0612
T_4	0.0705	0.103	0.0713
T_5	0.0806	0.1202	0.0815
T_6	0.0906	0.1374	0.0917
T_7	0.104	0.146	0.1053
T_8	0.1175	0.1546	0.1189
T_9	0.1309	0.1632	0.1325
T_{10}	0.1443	0.1717	0.1461
Positive S^+	0.0403	0.0515	0.0408
Negative S^-	0.1443	0.1717	0.1461

In our proposed TOPSIS–PSO algorithm, PSO has with complexity of O(ntlogn) in which n is number of populations and t is number of epochs [28]. In addition, TOPSIS algorithm has complexity of $O(n^2 + n + 1)$ for FV computation. $O(n^2)$ is complexity of attribute normalization and weighting, O(n) is required for finding PIS–NIS and O(1) is required for ranking the attributes [29].

6 Results and discussion

In the result section, paper illustrates the outcome to study the performance of proposed TOPSIS–PSO algorithm by comparing with various existing algorithms with respect to MakeSpan, transmission time, cost and average resource

Table 9 Separation from positive ideal solution

Alternative	Criteria			
	EX_T	$Trans_T$	Cost	S*
T_1	0.0000	0.0000	0.0000	0.0000
T_2	0.0001	0.0003	0.0001	0.0224
T_3	0.0004	0.0012	0.0004	0.0448
T_4	0.0009	0.0027	0.0009	0.0670
T_5	0.0016	0.0047	0.0017	0.0894
T_6	0.0025	0.0074	0.0026	0.1118
T_7	0.0041	0.0089	0.0042	0.1310
T_8	0.0060	0.0106	0.0061	0.1506
T_9	0.0082	0.0125	0.0084	0.1706
T_{10}	0.0108	0.0144	0.0111	0.1907

Table 10 Separation from negative ideal solution

Alternative	Criteria			
	EX_T	$Trans_T$	Cost	S′
T_1	0.0108	0.0144	0.0111	0.1907
T_2	0.0088	0.0106	0.0090	0.1688
T_3	0.0070	0.0074	0.0072	0.1470
T_4	0.0054	0.0047	0.0056	0.1255
T_5	0.0041	0.0027	0.0042	0.1043
T_6	0.0029	0.0012	0.0030	0.0838
T_7	0.0016	0.0007	0.0017	0.0628
T_8	0.0007	0.0003	0.0007	0.0418
T_9	0.0002	0.0001	0.0002	0.0209
T_{10}	0.0000	0.0000	0.0000	0.0000

Table 11 Relative closeness (FV)

Alternative	S*	S′	S* + S′	FV = S′/(S* + S′)
T_1	0.000	0.1905	0.1905	1.0000
T_2	0.0224	0.1685	0.1909	0.8827
T_3	0.0447	0.147	0.1917	0.7668
T_4	0.0671	0.1253	0.1924	0.6512
T_5	0.0894	0.1049	0.1943	0.5399
T_6	0.1118	0.0843	0.1961	0.4299
T_7	0.1311	0.0632	0.1943	0.3253
T_8	0.1507	0.0412	0.1919	0.2147
T_9	0.1706	0.0224	0.193	0.1161
T_{10}	0.1905	0.0000	0.1905	0.0000

Table 12 PSO parameters

Parameters	Values
Swarm-size	10
No. of iterations	150
Self-consciousness study factor C_1	1.49445
Swarm consciousness study factor C_2	1.49445
V_{min}	0
V_{max}	1

Table 13 Task assignment matrix to VMs

Criteria	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
VM_1	0	1	0	0	0	0	0	1	1	0
VM_2	0	0	0	0	1	0	1	0	0	0
VM_3	0	0	1	1	0	0	0	0	0	0
VM_4	1	0	0	0	0	1	0	0	0	0
VM_5	0	0	0	0	0	0	0	0	0	1

utilization. The limitation of [24] is, these are single objective based task scheduling algorithms where TOPSIS–PSO overcomes the limitations by choosing MCDM. The following section describes the analysis of the achieved result with cloud metrics.

6.1 Type 1: (TOPSIS–PSO- vs. -PSO- vs. -DAPSO and TOPSIS–PSO- vs. -ABC- vs. -IABC)

For the first Type 1 experiment, we compared our approach with PSO, dynamic PSO (DAPSO) [14] in terms of MakeSpan, transmission time and resource utilization. The same experiment is performed with ABC and IABC [24] in terms of processing cost. The parameter setting is given in Table 3.

6.1.1 Analysis of MakeSpan

The MakeSpan metric gives the CT of tasks in VMs. The main objective of the work is to minimize MakeSpan for fast execution of tasks. The MakeSpan of TOPSIS–PSO is compared to DAPSO and PSO algorithm. MakeSpan is individually observed with respect to number of tasks (10 to 40) that are having 5 and 10 number of VMs respectively. The MakeSpan is calculated using Eq. (4). Tables 14 and 15 depict the calculated values of MakeSpan for 5 and 10 VMs. Figures 5 and 6 depict the total MakeSpan for PSO, DAPSO and TOPSIS–PSO. The

Table 14 MakeSpan with five VM

Tasks	PSO	DAPSO	TOPSIS–PSO
10	3.200	2.620	1.490
20	5.627	4.520	3.710
30	8.896	8.250	6.680
40	13.311	10.250	10.150

Table 15 MakeSpan with 10 VM

Tasks	PSO	DAPSO	TOPSIS–PSO
10	2.25	1.74	0.77
20	3.89	2.98	1.84
30	7.24	5.72	3.46
40	8.59	5.3	5.13

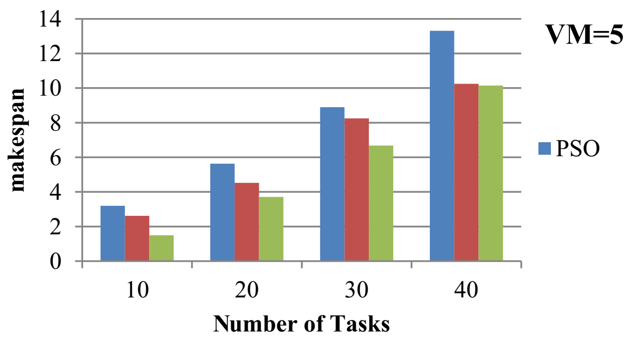

Fig. 5 Comparative analysis on MakeSpan with five VMs

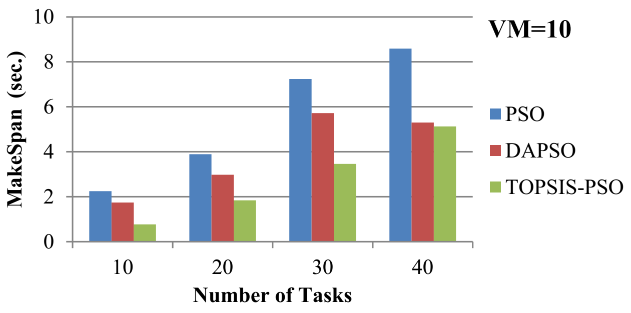

Fig. 6 Comparative analysis on MakeSpan with 10 VMs

analysis shows that proposed TOPSIS–PSO algorithm provides 16.97% and 7.57% less MakeSpan than PSO and DAPSO for five VMs. Similarly, it shows 16.85% and 32.40% less MakeSpan for 10 VMs respectively.

Table 16 Transmission time

Tasks	PSO	TOPSIS–PSO
10	0.711	0.664
20	1.148	1.104
30	1.590	1.524
40	2.100	2.000

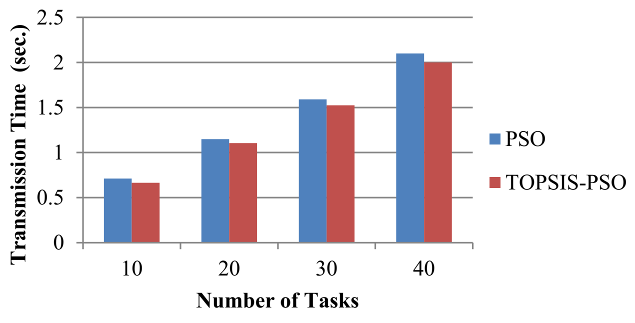

Fig. 7 Comparative analysis on transmission time

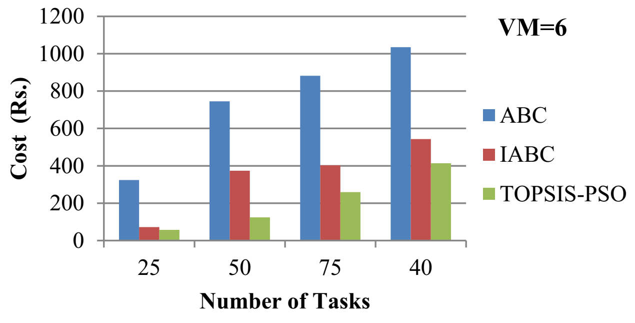

Fig. 8 Comparative analysis on cost

6.1.2 Analysis of transmission time

The $Trans_T$ metric gives the time taken by task to reach in VMs. The $Trans_T$ of TOPSIS–PSO and PSO algorithms are compared. $Trans_T$ is calculated using Eq. (2) and calculated outcomes are shown in Table 16 with respect to number of task (10 to 40). Figure 7 shows the graph representation of $Trans_T$ for PSO and TOPSIS–PSO algorithm. The obtained results show that proposed algorithm achieves 2.37% less $Trans_T$ than PSO which shows better result.

6.1.3 Analysis of processing cost

The PC metric gives the overall processing cost spent on processing tasks on VMs. Analysis of PC is compared with algorithm ABC and IABC. In cloud computing environment PC is the most effective metric of concern. Minimization of PC leads to better QoS to the users. Figure 8 shows the comparative analysis for PC with respect to

Table 17 Processing cost

Tasks	ABC	IABC	TOPSIS–PSO
25	324.21	72.34	57.35
50	745.02	374.01	124.05
75	881.45	402.61	259.10
40	1034.41	543.32	414.01

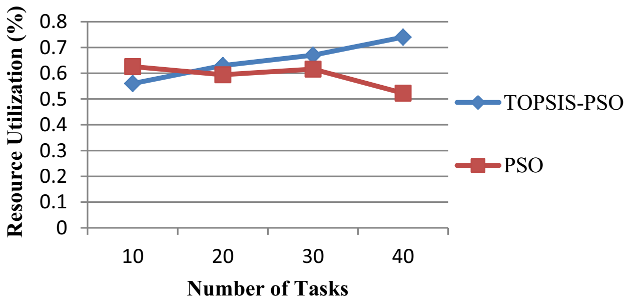

Fig. 9 Comparative analysis on $\mathrm{AVG_U}$

Table 18 Resource utilization

Tasks	PSO	TOPSIS–PSO
10	0.63	0.56
20	0.59	0.63
30	0.62	0.67
40	0.52	0.74

number of executed tasks. The PC is calculated using Eq. (3) and the obtained values of proposed algorithm are shown in Table 17. TOPSIS–PSO is compared to ABC and IABC algorithms that show the TOPSIS–PSO has reduced

Table 19 Cloud parameter settings Type 2 experiment

Parameters	Values
Length of task	1000–20,000
Total number of tasks	100–500
Total number of VMs	50
VM frequency	500–2000
Population size	10
VM memory (RAM)	256–2048
VM bandwidth	500–1000
Number of PEs requirements	1–4
Number of DCs	10
Number of PMs	2–6

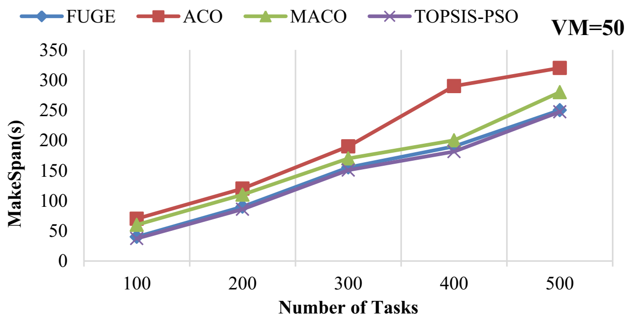

Fig. 10 Comparative analysis of average MakeSpan

the PC 23.93% and 55.49% than IABC and ABC respectively.

6.1.4 Analysis of resources utilization

Better utilization of resources shows the low wastage of resources. It also shows the efficiency of the system to the resources (RAM, CPU). Figure 9 depicts the $\mathrm{AVG_U}$ by the proposed algorithm with respect to executed tasks. Here we compare $\mathrm{AVG_U}$ performance of proposed algorithm with PSO that shows the TOPSIS–PSO algorithm achieves higher $\mathrm{AVG_U}$ as number of tasks significantly increases. $\mathrm{AVG_U}$ is calculated using Eq. (5) and values of PSO and TOPSIS–PSO shown in Table 18. The analysis shows that proposed TOPSIS–PSO algorithm achieves approximate 75% of utilization for six numbers of VMs which is better in results than PSO for greater number of tasks.

6.2 Type 2: (TOPSIS–PSO- vs. -FUGE- vs. -ACO- vs. -MACO)

For the second experiment, we compared our approach with the ACO, multiple ACO (MACO) and FUGE algorithms [7] in terms of MakeSpan. The parameter setting for Type 2 experiment is given in Table 19.

Figure 10 depicts the MakeSpan obtained by the proposed algorithm with respect to executed tasks. Here MakeSpan performance of the proposed algorithm compared with existing algorithms such as: FUGE, ACO and

Table 20 Average MakeSpan

Tasks	FUGE	ACO	MACO	TOPSIS–PSO
100	40	70	60	37.14
200	90	120	110	85.79
300	155	190	170	150.59
400	190	290	200	181.45
500	250	320	280	247.34

MACO. The results show that the performance of TOPSIS–PSO algorithm have reduces MakeSpan for the given configuration. MakeSpan values for FUGE, ACO, MACO and TOPSIS–PSO are shown in Table 20. The analysis shows that proposed TOPSIS–PSO algorithm reduces 3.1, 29.1 and 14.4% total MakeSpan than FUGE, ACO and MACO respectively. The obtained results conclude that TOPSIS–PSO has the better performance for the given parameters.

7 Conclusion

This paper, introduced novel TOPSIS based task scheduling algorithm to achieve better outcome of cloud metrics. The novel idea is to combine TOPSIS and PSO algorithm. The TOPSIS–PSO algorithm works on two stages. TOPSIS method calculates the RC of VMs with respect to each task. The proposed algorithm works on the multi-criteria based approach where three criteria (execution time, transmission time and cost) are taken to improve the scheduling process. PSO algorithm receives the calculated RC of each task which acts as FV of tasks (particles). Tasks are arranged according to TOPSIS method and assigned to VMs depending on their FV over PSO. The work has resulted in improvement of execution time, MakeSpan, resource utilization, processing cost, and transmission time as compared to the dynamic scheduling algorithms. The TOPSIS method to calculate RC for PSO has been remarkable for dynamic environment. TOPSIS–PSO algorithm performs better than FUGE algorithm for given parameters, but for more number of tasks the algorithm underperforms as it lacks load balancing mechanism. In future we intend to extend our proposed task scheduling work with the consideration of load balancing for large number of tasks and power consumption in order to improve energy efficiency in cloud environment.

References

1. Li, Q., Hao, Q., Xiao, L., Li, Z.: Adaptive management of virtualized resources in cloud computing using feedback control. In: First International Conference on Information Science and Engineering, Nanjing, China, pp. 99–102. IEEE (2009)
2. Parikh, K., Hawanna, N., Haleema, P.K., Jayasubalakshmi, R., Iyengar, N.: Virtual machine allocation policy in cloud computing using CloudSim in Java. Int. J. Grid Distrib. Comput. **8**(1), 145–158 (2015)
3. Tawfeek, M., El-Sisi, A., Keshk, A., Torkey, F.: Cloud task scheduling based on ant colony optimization. Int. Arab J. Inf. Technol. **12**(2), 129–137 (2015)
4. Zhan, Z., Liu, X., Gong, Y., Zhang, J.: Cloud computing resource scheduling and a survey of its evolutionary approaches. ACM Comput. Surv. (2015). https://doi.org/10.1145/2788397
5. Panwar, N., Rauthan, M.S.: Analysis of various task scheduling algorithms in cloud environment: review. In: 7th International Conference on Cloud Computing, Data Science and Engineering—Confluence, pp. 255–261. IEEE (2017)
6. Pooranian, Z., Shojafar, M., Abawajy, J.H., Abraham, A.: An efficient meta-heuristic algorithm for grid computing. J. Comb. Optim. **30**(3), 413–434 (2015)
7. Shojafar, M., Javanmardi, S., Abolfazli, Saeid., Cordeschi, Nicola.: FUGE: a joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. Clust. Comput. **18**(2), 829–844 (2015)
8. Tsou, C.: Multi-objective inventory planning using MOPSO and TOPSIS. Expert Syst. Appl. **35**, 136–142 (2008)
9. Behzadian, M., Khanmohammadi Otaghsara, S., Yazdani, M., Ignatius, J.: A state-of the-art survey of TOPSIS applications. Expert Syst. Appl. **39**(17), 13051–13069 (2012)
10. Nelson Jayakumar, D., Venkatesh, P.: Glowworm swarm optimization algorithm with TOPSIS for solving multiple objective environmental economic dispatch problem. Appl. Soft Comput. **23**, 375–386 (2014)
11. Jia, L., Zou, G., Fan, L.: Combining TOPSIS and particle swarm optimization for a class of nonlinear bilevel programming problems. In: 10th International Conference on Computational Intelligence and Security. IEEE (2014). https://doi.org/10.1109/cis.2014.52
12. Wang, Y., Li, B., Weise, T., Wang, J., Yuan, B., Tian, Q.: Self-adaptive learning based particle swarm optimization. Inf. Sci. **181**(20), 4515–4538 (2011)
13. Zhang, P., Zhou, M.: Dynamic cloud task scheduling based on a two-stage strategy. IEEE Trans. Autom. Sci. Eng. **99**, 1–12 (2017)
14. Al-maamari, A., Omara, F.A.: Task scheduling using PSO algorithm in cloud computing environments. Int. J. Grid Distrib. Comput. **8**(5), 245–256 (2015)
15. Zhan, S., Huo, H.: Improved PSO-based task scheduling algorithm in cloud computing. J. Inf. Comput. Sci. **9**(13), 3821–3829 (2012)
16. Panwar, N., Negi, S., Rauthan, M.S.: Non-live task migration approach for scheduling in cloud based applications. In: NGCT 2017, CCIS, vol. 828, pp. 124–137 (2018)
17. Awad, A.I., El-Hefnawy, N.A., Abdel_kader, H.M.: Enhanced particle swarm optimization for task scheduling in cloud computing environments. Procedia Comput. Sci. **65**, 920–929 (2015)
18. Lakraa, A.V., Yadav, D.K.: Multi-objective tasks scheduling algorithm for cloud computing throughput optimization. Procedia Comput. Sci. **48**, 107–113 (2015)
19. Cho, K.M., Tsai, P.W., Tsai, C.W., Yang, C.S.: A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing. Neural Comput. Appl. **26**(6), 1297–1309 (2015)
20. Jena, R.K.: Multi objective task scheduling in cloud environment using nested PSO framework. Procedia Comput. Sci. **57**, 1219–1227 (2015)
21. Ghanbari, S., Othman, M.: A priority based job scheduling algorithm in cloud computing. In: International Conference on Advances Science and Contemporary Engineering (ICASCE 2012), vol. 50, pp. 778–785 (2012)
22. Lawrance, H., Silas, S.: Efficient QoS based resource scheduling using PAPRIKA method for cloud computing. Int. J. Eng. Sci. Technol. **5**(03), 638–643 (2013)
23. Shih, H.S., Shyur, H.J., Lee, E.S.: An extension of TOPSIS for group decision making. Math. Comput. Model. **45**, 801–813 (2007)
24. Selvarani, S., Sadhasivam, G.S.: Improved cost-based algorithm for task scheduling in cloud computing. In: IEEE International Conference on Computational Intelligence and Computing

Research (ICCIC), pp. 1–5 (2010). https://doi.org/10.1109/ICCIC.2010.5705847
25. Lin, W., Liang, C., Wang, J.Z., Buyya, R.: Bandwidth-aware divisible task scheduling for cloud computing. Softw. Pract. Exp. **44**, 163–174 (2012)
26. Zhang, Q., Liang, H., Xing, Y.: A parallel task scheduling algorithm based on fuzzy clustering in cloud computing environment. Int. J. Mach. Learn. Comput. **4**(5), 437–444 (2014)
27. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp. **41**, 23–50 (2011)
28. Ruhela, D.S.: A study of computational complexity of algorithms for numerical methods. PhD Thesis, University of Rajasthan, Rajasthan, India (2014)
29. Hamdani, H., Wardoyo, R.: The complexity calculation for group decision making using TOPSIS algorithm. AIP Conf. Proc. (2016). https://doi.org/10.1063/1.4958502

Neelam Panwar is a Research Scholar in the Department of Computer Science and Engineering at the HNB Garhwal University Uttarakhand, India. She completed B.Sc. Degree in PCM (Physics, Chemistry, and Mathematics) from HNB Garhwal University, Srinagar Garhwal, Uttarakhand, India. She received MCA Degree from Uttarakhand Technical University, Dehradun Uttarakhand, India. Her research interests focus on cloud computing.

Sarita Negi did her B.Tech. Degree in Computer Science and Engineering from Central Institute of Technology, Gauhati University, India in 2013. She received her Master Degree in Computer Science and Engineering from the Department of Computer Science and Engineering, Govind Ballabh Pant Engineering College of Uttarakhand, India in 2016. She is currently pursuing her Ph.D. from Uttarakhand Technical University, Dehradun. Her research interests include wireless sensor network, VANET and cloud computing.

Man Mohan Singh Rauthan is working as H.O.D., Computer Science and Engineering, HNBGU, Srinagar Garhwal, and Uttarakhand, India. He completed M.Sc. (Physics) from I.I.T., Delhi. He received MTech Degree in Computer Science from Roorkee University. He did his Ph.D. in Information Technology from Kumaon University, Uttarakhand, India. He worked at Govt. Computer Center, Bhopal as a Programmer. He also worked at National Informatics Centre, Delhi as Sr. Systems Analyst for more than 6 years. His research interest includes information retrieval and cloud computing.

Kunwar Singh Vaisla received the graduation in Science (B.Sc.) and Master (MCA) degree in Computer Applications from University of Rajasthan, Jaipur in 1994 and 1998, respectively. He completed Ph.D. in Computer Science from Kumaon University, Nainital in 2011. Presently working as Associate Professor (Computer Science and Engineering) in Kumaon Engineering College (A Govt. Autonomous College), Dwarahat (Almora), Uttarakhand. Interested field of research are ICT impact on G2C of e-Governance, data warehouse and mining, complex/compound object mining, IBIR.