Cluster Computing (2024) 27:341–376
https://doi.org/10.1007/s10586-022-03819-5

Security prioritized multiple workflow allocation model under precedence constraints in cloud computing environment

Mahfooz Alam[1] · Mohammad Shahid[2] · Suhel Mustajab[1]

Received: 23 December 2021 / Revised: 3 November 2022 / Accepted: 6 November 2022 / Published online: 3 January 2023

Abstract
In the last decade, cloud computing has become an effective and efficient service delivery platform to offer resources, innovation, and economies of scale on telecommunication networks. An allocation scheme without security constraints consideration may assign the high security-sensitive tasks onto the lower trustworthy machines. This would lead the performance deterioration. To address this issue, in this paper, a security-prioritized multiple workflow allocation (SPMWA) model is proposed by integrating the security-prioritized mapping scheme for Infrastructure-as-a-Service cloud computing environment. It is expected that incorporating a security-prioritized allocation scheme under precedence constraints will enhance the performance of workflow processing in risk-prone environments. In this model, more priority is given to tasks with high-security demand to get allocated onto the more trustworthy virtual machines during allocation to minimize the failure probability of the cloud system. The failure probability can be minimized by assigning the tasks on to the trustable virtual machines exhibiting sufficient trust level to minimize the number of task failures. The number of task failure, failure probability, and makespan have been computed for the comparative evaluation of the SPMWA. For performance comparison, the SPMWA model has been compared with state-of-the-art models from the literature having the same environment and objective. The experimental evaluation confirms the superior performance of the proposed model on account of the considered objective among its peers.

Keywords Cloud computing · Multiple workflow allocation · Security · Precedence constraints · Number of task failure · Failure probability

1 Introduction

Cloud computing is an internet-based service infrastructure that facilitates on-demand access on a pay-as-you-go basis for users of configuration resources such as stockpiling, server, network, administrations, and other applications to allow them pervasive services which can be rapidly discharged and provisioned with insignificant administrative effort [1, 2]. Day-by-day, cloud computing is growing in heterogeneous distributed processing, automatic computing, utility, and matrix computing. The heterogeneous distributed computing facilitates any type of administration for worldview like social networking, registering applications of computational assets, broadcast communication, and web administration. From these benefits of cloud computing, cloud users can make use of processing assets, and an apparition of IT resources used on-demand. These assets can be used for managing the different types of services such as Workflow as a Service (WaaS), Infrastructure as a Service (IaaS), Network as a Service (NaaS), Platform as a Service (PaaS), Storage as a Service (StaaS) Software as a Service (SaaS), and Data as a Service (DaaS) [3–6]. Cloud computing also allows anyone to provision services, virtual hardware, and runtime environments with a credit card. However, with newly developed technology,

✉ Mohammad Shahid
mdshahid.cs@gmail.com

Mahfooz Alam
mahfoozalam.amu@gmail.com; malam.cs19@gmail.com

Suhel Mustajab
suhelmustajab@gmail.com

[1] Department of Computer Science, Aligarh Muslim University, Aligarh, India

[2] Department of Commerce, Aligarh Muslim University, Aligarh, India

several issues have to be addressed in cloud computing such as dynamic resources provisioning, virtualization technologies and large computing infrastructure for cloud service providers (CSPs), availability and storage for large data processing, protection, and confidentiality of resources, legal issues arise in different countries and losing of data [2, 6, 7]. Some researchers and scientists utilize the CSP for computation-intensive applications and running large-scale data by workflow applications.

Workflow is a specific model used for scientific application in various domains having dependent and communicating components [8]. Generally, a workflow application is modeled by using Directed Acyclic Graph (DAG), having a node-set representing tasks and an edge set with the dependencies among the tasks using the directed edges between them. Workflow allocation faces several challenges due to its dynamic nature and heterogeneity as well as searching for suitable resources of geographical distribution to construct the allocation decision while meeting optimization of the Quality of Service (QoS) parameters in a cloud computing environment. The workflows are used in several applications from different domains like genomics, earthquake analysis, gravitational waves detection, astronomy, healthcare, project planning, chemical reaction, and supply chain management. Since a huge amount of data is processed in cloud computing daily so the task allocation mechanism is more important and should be computed efficiently. Workflow's task allocation is one of the most common application models, particularly in designing, business plans, and logical fields which has been directed towards workflow task allocation. Therefore, many researchers address the workflow tasks allocation problem for many aspects such as computational time minimization [9–15], minimum budget assurance [16, 17], least energy consumption [18, 19], improving the throughput and server performance [20, 21] and employing security measures [22–32] in heterogeneous and other efficient computing [33, 34] for single workflow. Further, the work has been also reported for multiple workflows allocation to improve the turnaround time, response time and flowtime [35–43], budget [43, 44], and energy [45, 46], by aggregating the batch of workflows prior to allocation. In workflow allocation, precedence constraints (execution order) management is one of the challenging issues. Precedence constraints in workflow tasks are preserved by various methods such as ranking methods [11], and level attribute methods [10, 38, 39]. In recent times, we can see that many researchers have been working to ensure the security constraints in the IaaS cloud and the trust of cloud users so that their information and applications are protected and managed effectively. However, the creation of ad-hoc security solutions, targeting a very small part of the whole problem makes a fair and sound evaluation of the state of the art. At this point, we start from the view that the cloud computing paradigm can be fully exploited only if the contributions of cloud users and CSPs for security constraints are made wider by improving their trust. As a result, software security assurance mechanisms improve the confidence level of cloud users and cloud transparency so that the CSPs behave as expected [47]. The standard software security assurance is defined with the line and the assurance of cloud security is defined to gain reasonable confidence that applications and/or infrastructure will reliably demonstrate more than one security constraint, and process performs as expected despite attacks and failures [48]. Since assurance is an extensive notion in a cloud environment than security for any type of workflow applications, it consists of methodologies for collecting and validating proof supporting security constraints. Moreover, various challenging works have been proposed for secure workflow allocation only for a single workflow [22–32] but lagging for multiple workflows. So, emergent work requires exploiting the process of security services for multiple workflow applications to defend security-critical applications from threats in the IaaS cloud where batch mode processing is necessary. In addition to security constraints in the cloud user request will increase the security overhead (in terms of time), which turns the turnaround time, flowtime, and operational cost of use increments [49] but reduces the risk or failure probability [24, 28].

In this paper, a security-prioritized multiple workflow allocation (SPMWA) model is proposed by integrating the security-prioritized mapping scheme for Infrastructure-as-a-Service cloud computing environment. It is expected that incorporating a security-prioritized allocation scheme under precedence constraints will enhance the performance of workflow processing in risk-prone environments. SPMWA gives more priority to the tasks having high-security demands to get allocated onto the more trustworthy virtual machines during allocation to minimize the failure probability of the system. The contributions of the proposed model are given below as follows:

- A Security Prioritized Multiple Workflows Allocation (SPMWA) model is proposed by incorporating a security prioritization scheme in workflow task allocation to minimize the failure probability of the system for the cloud computing environment.
- A security model is introduced to estimate the failure probability of the system. This model also calculates the total number of task failure to assess how often tasks with a given security requirement end up being allocated on the VMs exhibiting insufficient trust.
- Workflows are partitioned in accordance to depth level to maintain precedence constraints. Communication

requirements among the workflow tasks are estimated by considering the edge weights and machine distances.
- An idle gap reduction policy is employed to the utilization of the idle gaps generated during the allocation process by accommodating the successor tasks from the next partitions.
- We expect that integrating such security measures would aid in designing a more robust workflow task allocation model for networked and high security-sensitive applications.
- The experimental results of SPMWA are compared with state-of-the-art workflow models from the literature. For this, we have taken two multiple workflow allocation strategies namely sequential-based strategy and merge-based strategy [35]. These strategies and security prioritized allocation are incorporated in HEFT [11] to develop two versions of Security Prioritized Multiple workflow allocation (SPM) models, namely SPM1 (Merge-Based) and SPM2 (Sequential-Based) respectively. For performance comparison, LBSIR [39] and HEFT [11] have also been included.
- The performance evaluation of the proposed model has been carried out on random multiple workflows and real-world scientific application graphs namely Montage [50], CyberShake [50], and LIGO [51].
- The proposed model can be used in many emerging security-sensitive domains such as batch mode transaction processing, chemical reaction, supply chain management, project management, and Pegasus project for some other scientific workflow applications.

The basic structure of the paper is as follows: Sect. 2 presents the standard and current literature review related to the work. Section 3 describes the system model for the proposed methodology. Section 4 describes the proposed model design with an algorithmic template, a simple illustration, and a time complexity analysis. The performance evaluation for the proposed work with experimental results is discussed in Sect. 5. The conclusion and future work are presented in Sect. 6.

2 Related work

Workflow allocations have been a common research topic in the recent computing environment for decades and have been built together with changes in technology. In the last few years, most of the research has been engrossed in workflow allocation problems using DAG heuristics [8, 11]. The mapping of workflow allocation problems is well-established NP-complete [52]. Consequently, various heuristic algorithms have been developed to achieve sub-optimal solutions for resource allocation in the cloud environment. Here, some reported works deal with independent task allocation considering VM placement [53], cost-effective task allocation [54], security-aware task allocation [55–57], etc. On other hand, many other models have presented the mechanism for addressing the allocation of dependent communication workflow tasks. In this section, the various models are further categorized into single workflow allocation [9–19], security-aware workflow allocation [22–32], and multiple workflow allocation models [35–46].

2.1 Single workflow allocation

In single workflow allocation, only one DAG is represented by task mapping onto parallel resources. The most common DAG-based scheduling is designed for a single workflow on heterogeneous distributed systems such as [9–19]. Dynamic-Level Scheduling (DLS) [9] is one of the first algorithms that find the availability of resources and thus allow the task to be scheduled onto the current busy resource. DLS does not guarantee the minimum processing time for a task. Furthermore, it also does not attempt to idle time gaps between two tasks on the same resource in contrast to other more current algorithms. Levelized Minimum Time (LMT) algorithm [10] is very simple and based on the precedence level of tasks onto the resource having the lowest processing time for a heterogeneous distributed system. The CPOP [11] algorithm attains better allocations than LMT and is similar to DLS with lower time complexity. The main characteristic of CPOP is the workflow of all the tasks that belong to the critical path assigned to a single resource. The HEFT [11] is one of the best DAG list scheduling algorithms, as it has quadratic time complexity. This algorithm aims to reduce the time complexity and schedule length. The modified HEFT has also been proposed to reduce processing time in a cloud computing environment [12]. Other versions of HEFT have also been proposed, such as Predict Earliest Finish Time (PEFT) [13], Communication aware Earliest Finish Time (CEFT) [14], and Dependency-ratio Bundling Earliest Finish Time (DBEFT) [15] to reduce the schedule length. PEFT algorithm defines the task priority based on the optimistic cost table. PEFT takes the assurance of task execution in minimum processing time. CEFT algorithm is based on the task duplication heuristic using communication ratio having task execution order as according to upward rank. DBEFT is also list-based task duplication scheduling to reduce communication costs. DBEFT improves the scheduling length ratio over the PEFT, CEFT, and HEFT. Some of the work has been proposed for multi-objective workflow allocation reducing the makespan and economic cost, [16, 17], and optimizing time and energy [18, 19].

2.2 Security-aware workflow allocation

Security plays an essential role in e-commerce and digital transaction processing systems. In the field of distributed information-sharing networks, significant work considering security constraints [22–32] has been reported in the heterogeneous distributed environment like grid/cloud computing to share and process the resources with trustworthy contributing peers. The authors presented a task allocation strategy with security constraints and the deadline for parallel applications [22] with the objectives of optimizing security parameters and processing time. However, it is implemented on homogeneous clusters. In [23], the authors present the trust-based allocation model for the scientific workflow to improve the stability of the schedule. In [24], authors have developed the security-driven scheduling (SDS) model for heterogeneous distributed systems to optimize the makespan, speedup, and risk probability. SDS introduces task priority allocation on the suitable processor using estimated security overhead. The strategy presented in [25] namely Cloud-DLS incorporates dynamic trust-based task allocation in the DLS algorithm in the cloud environment. The objective of Cloud-DLS is to assure the execution of the task and minimize the processing time. In [26], the authors have proposed a trust service-oriented workflow allocation (TMOWS) model to minimize the execution time and cost simultaneously in a cloud environment using fuzzy member functions. TMOWS meets the security demands of the users or other constraints with balance factors. In [27], the authors model maintains the reliability of services. It avoids discrete events, and workflow application failure between the direct and recommended trust. It also found the best solution and concurrently satisfied deadline conditions. The work in [28] presents a novel security-sensitive workflow allocation with a task duplication (SOLID) scheme. SOLID has been developed having three features, firstly, the selection of duplicated predecessor tasks which is useful to avoid the data encryption and transmission time by delaying the start time of the task, further, defines the latest finish time of workflow's task, and lastly, it also assures these tasks should be finished on the cheapest resources with aim of minimizes the makespan and monetary cost. In [29], trust-based stochastic workflow scheduling (TSS) is proposed to minimize the makespan with increased speedup using the TSS trust model for security estimation including both direct trust and reputation relationships. The strategy presented in [30] for security-aware workflow allocation (SAWA) is to reduce the number of failed tasks. SAWA selects the task allocation as per depth level. In [32], the authors are presenting a security-prioritized HEFT (SPHEFT) algorithm in the cloud computing environment to optimize the guarantee ratio. SPHEFT is integrated by the security requirements into the HEFT [11] algorithm by giving more priority to the tasks having a high degree of security constraints. SPHEFT creates the clusters for distinct upward rank values and then again sorts each cluster as per the security demand of the tasks. Thus, the tasks with higher security demand will get more chances to execute at first. Therefore, it will improve the guarantee ratio over the HEFT algorithm.

2.3 Multiple workflow allocation

In the multiple workflow allocation, more than one workflow task is grouped to form a batch for processing on suitable machines to achieve the desired QoS parameters. To tackle the multiple workflow allocation problems, Bittencourt and Madeira [35] have first introduced four strategies to schedule multiple workflows namely sequential-based strategy, gap search strategy, interleave strategy, and merge-based strategy. The sequential-based strategy, schedules the workflows sequentially, one after another on the available resources. The gap search strategy works the same as the first strategy but it finds the gaps between tasks already executed, and then accommodatable tasks from the workflow at hand are being scheduled into the found gaps without interfering with their starting time. Interleave strategy uses both the first and second strategies, however, the strategy schedules tasks of each workflow in turns, interleaving their tasks in the schedule of the available resources. The merge-based multiple workflow strategy merges all workflows into a single one and then schedules this resulting workflow as a single workflow. Here, a significant number of works have been proposed for multiple workflow allocation to reduce turnaround time [38, 39]. The strategy proposed in [36] aggregates multiple workflows to achieve near-optimal throughput for heterogeneous cloud computing. In [37], the authors analyze the allocation strategy for multiple workflows including two and four stages like labeling, adaptive information, prioritization, and parallel machines in the grid environment. In [38], the authors have proposed a novel approach (SLBBS) for the multiple workflow allocation problem in a computational grid environment to optimize the turnaround time. SLBBS divides the multiple workflows as per depth level and allocation of tasks is assigned on best-fit resource levelwise. Level-based Batch scheduling Strategy with Idle slot Reduction (LBSIR) [39] strategy reduces the drawbacks of SLBBS by incorporating an idle slot reduction policy. The work reported in [40], concurrently executes multiple workflows using a rescheduling algorithm and dynamic task rearrangement to exploit task allocation flexibility under precedence constraints. In [41], the authors present the cluster-based allocation strategy for

multiple workflow applications with soft deadlines to achieve the quality of schedule in terms of fairness and execution time. In [42], the authors present deadline budget workflow allocation to optimize the time and cost in the cloud environment. Some of the work has been proposed for multi-objective of multiple workflow allocation reducing the makespan and economic cost [43, 44] and to optimizing time and energy [45, 46].

In the literature, various approaches have been proposed for solving single workflow [9–19] and multiple workflows [35–43] allocation problems in heterogenous distributed systems. However, many cloud-based workflow applications need to be processed in batch mode to enhance the systems performance. Therefore, to develop superior multiple workflow allocation models is a requirement in various domains. Further, as we see in Table 1, a significant number of approaches [22–32] have also been proposed for security-aware workflow allocation problems for considering only a single workflow. In the majority of the work, the task execution order is computed by the ranking method, and then the allocation of the resources is done as per the task execution order. In this scenario, high-security demand tasks with low rank may be allocated to untrustworthy machines leading to more failure in the system. But in the proposed work, workflow tasks are grouped into partitions as per depth level. In each partition, the task execution order is computed by using the security demand levels. In this way, high-security demand tasks always get allocated first and have a chance of getting higher trustable machines leading to lower failures in the system.

Finally, in multiple workflow scenario, the models [35–46] have also been reported for cloud environments to optimize the makespan, schedule length, speedup, energy, total budget, and utilization. However, as per our knowledge, we have not found any work on multiple workflow allocation considering security constraints in the cloud environment. Therefore, this paper is the first attempt toward the allocation of the batch of workflow applications satisfying the security requirements of the workflow tasks.

3 System model

This section presents an insight into the SPMWA developed for multiple workflow allocation, representing each workflow by using DAGs. e.g., the notation used, workflows descriptions, virtual machine descriptions, security model, parameter estimation, and problem statement. The proposed model aims to produce an efficient schedule for the workflow tasks and optimizes considered parameters from the quality of service of the cloud.

3.1 Symbols used

To describe the various models, symobls, and parameters involved in the process have been listed in Table 2 as follows:

3.2 Multiple workflow model

In this work, we consider a set of multiple workflows, $\mathcal{W}f = \{\mathcal{W}f_i : i = 1, 2, 3, \ldots\ldots N_{\mathcal{W}f}\}$ and each $\mathcal{W}f_i$ has tasks set, $\tau_i = \{T_{ij} : 1 \leq j \leq N_{\mathcal{W}f_i}\}$. Each workflow consists of tasks with parent (predecessor) and child (successor) relationships. Here, j^{th} task of i^{th} workflow (T_{ij}) has a set of successor tasks ($Succ(T_{ij})$) and predecessor tasks ($Pred(T_{ij})$). The successor tasks depend on the predecessor tasks with a communication requirement termed edge weight (e_{ixy}) between all possible pair of tasks. The following characteristics of multiple workflows in this work are given below:

- The batch of $N_{\mathcal{W}f}$ the number of compute-intensive workflows, represented by DAG.
- The number of tasks in each $\mathcal{W}f_i$ is N_{wf_i}.
- Each $T_{ij} \in \mathcal{W}f_i$ has an associated level attribute (l_{ij}).
- Precedence and dependency constraints have been handled through level attributes.
- The number of depth levels ($l_{\mathcal{W}f_i}$) in an i^{th} workflow is defined as $l_{\mathcal{W}f_i} = \max_{\forall j}\{l_{ij} : \forall T_{ij} \in \mathcal{W}f_i\}$.
- The depth level of a batch of workflows ($\mathcal{W}f$) is defined as, $L = \max_{\forall i}(l_{\mathcal{W}f_i})$.
- The multiple workflows are divided into L partitions (d^l) as per the depth levels such that $d^l = \{T_{ij} : \forall T_{ij} \& l_{ij} = l\}$.
- The e_{ixy} is for communication between T_{ix} and T_{iy}, and it is considered in MIs.

The set of multiple workflows is presented in Fig. 1 with all associated attributes. We can see that each partition tasks are shown by using the same color e.g. (T_{11}, T_{12}, T_{21},…,$T_{N_{\mathcal{W}f}3}$), (T_{13}, T_{14}, T_{22},….,$T_{N_{\mathcal{W}f}4}$), ($T_{1N_{\mathcal{W}f_1}}$, T_{23}, T_{24},….,$T_{N_{\mathcal{W}f}7}$) and ($T_{2N_{\mathcal{W}f_2}}$,…,$T_{N_{\mathcal{W}f}N_{\mathcal{W}f_{N_{\mathcal{W}f}}}}$) are at depth levels 1, 2, 3, and L depicted by using sky blue, magenta, canary, and green color respectively. Further, we can see in Fig. 1, T_{13} is dependent on T_{11} and T_{12} to start its execution with edge weights, whereas the tasks are at the same precedence level within the batch of tasks. For example, T_{11}, T_{12}, T_{21},…,$T_{N_{\mathcal{W}f}3}$ can be executed in parallel.

Table 1 A comparison of related works for workflow allocation models

Ref.	Techniques/approaches used	Workflow types	QoS parameters	Compared algorithms	Advantages	Limitations
[10] (2009)	Level based heuristic	Single	Speedup	NA	Simple level attribute is used for task ordering	Communication requirement is not considered in the mapping of task onto machines
[11] (2002)	Earliest finish time-based method using upward / downward rank	Single	Makespan, Speedup	CPOP, MH, DLS, LMT	Minimize the total execution time and effectively achieve a better schedule length ratio	Suffer from the idle gap and load balancing
[13] (2013)	Prediction of earliest finish time using optimistic cost table and allocation order is as upward or downward rank	Single	Schedule length ratio, Speedup	HEFT, HCPT, PETS, lookhead	Minimize the total execution time and effectively achieve a better schedule length ratio	Suffer from the idle gap and load balancing
[14] (2017)	HEFT-based model using communication ratio for assignment of tasks	Single	Schedule length ratio, Speedup, Average surplus time, Redundant computing ratio	PEFT, HDCPD, HEFT	It reduces the communication overhead and schedule length ratio	Suffer from the idle gap and load balancing
[15] (2019)	Extended version of HEFT with task dependency ratio-based allocation	Single	Schedule length ratio, Speedup	HEFT, PEFT, CEFT	It reduces the communication overhead and schedule length ratio	Suffer from the idle gap and load balancing
[22] (2008)	Assigns the tasks based on the critical path to minimum security demands	Single	Makespan, Guarantee Ratio, Risk Probability	EDF, LLF	Satisfies the deadline meet of the tasks	It is valid only for homogeneous clusters
[23] (2009)	Genetic algorithm	Single with multi-objective	Success Probability		To meet the deadline and budget	The scheduler sometimes has to select a less trustable machine for tasks of shorter processing time or cheaper price
[24] (2010)	Tasks are processed on the machine that takes the least EFT based on security upward rank	Single	Makespan, Risk probability, and Speedup	HEFT	Decrease the total execution time and effectively achieve a better schedule length ratio	Suffer from the idle gap and load balancing
[25] (2012)	Security constraints are incorporated in dynamic level scheduling (DLS)	Single	Schedule length and Guarantee Ratio	DLS, BSA	To assurance of tasks execution and reduce the failure probability	It does not attempt to idle time gaps between two tasks on the same resource
[28] (2017)	Tasks are processed based on upward rank with duplicating predecessor tasks to idle time slots considering laxity times on resources and data encryption	Single with multi-objective	Makespan, Resource Utilization, and monetary cost	Earliest finish time-Maximum Effective Reduction	To reduce monetary cost without delaying their successors' start and workflows' makespan	Suffer from the load balancing
[29] (2019)	Schedule planning based on the stochastic trust model	Single	Makespan, Speedup	HEFT, SHEFT	It optimizes both securities for trust model and makespan	Suffer from the load balancing

Table 1 (continued)

Ref.	Techniques/approaches used	Workflow types	QoS parameters	Compared algorithms	Advantages	Limitations
[32] (2022)	Tasks are executed onto trustable VMs ensuring security demands on a priority basis	Single	Makespan, Security Overhead and Guarantee Ratio	HEFT	It reduces the communication overhead and schedule length ratio	Suffer from the idle gap and load balancing
[36] (2011)	Level-based mapping heuristic	Multiple	Throughput	HEFT	To maximize the throughput	Communication requirements among tasks are not taken into consideration
[37] (2012)	Multi-stage multiple workflow allocation policies with user run time estimates	Multiple	Approximation factor, critical path slowdown, and waiting time	CPOP, HEFT	To reduces the performance degradation	suffer from a significant time complexity
[39] (2015)	Levelized-based mapping heuristic with idle gap reduction at each depth level	Multiple	Turnaround Time, Flowtime, Utilization	SLBBS, HEFT, DLS, CPOP	Using idle gap reduction performance is improved	Extra waiting time in queue is due to batch formation
[40] (2015)	Reservation adjustment on rescheduling heuristic with the support of task rearrangement	Multiple	Makespan, Utilization, Speedup, and running time	HEFT, DCP, DAGMap	To exploit the scheduling flexibility of multiple workflows	Extra overhead has occurred for the task rearrangement
[41] (2016)	Clustering-based workflow allocation scheme	Multiple	Makespan, Fairness, Utilization	Path Clustering Heuristic	To makes a qualitative schedule and improve the effectiveness of resource utilization	Low parallelism and time restrictions quality of schedule
[42] (2019)	Assigns the scheduling priorities to the workflows based on a weighted upward-rank priority policy	Multiple	Schedule length ratio, Speedup, Success rate	MSLBL, HBCS, BHEFT	To improve the plain upward-rank priority policy	Suffer from the idle gap
[43] (2021)	Allocation of each task is done based on EST and EFT which is computed by the deadline of the task	Multiple	Resource utilization, Rental Cost, Success Rate, Deadline deviation	ROSA, NOSF	To satisfy the deadline and meet the tasks	Suffer from the load balancing

3.3 Machine model

In this work, we consider a cloud computing environment having a set of n heterogeneous VMs, $V = \{V_k \mid 1 \leq k \leq n\}$ interconnected via a fully connected network. The computational VM under this work with following characteristics are:

- We consider n, number of heterogeneous VMs.
- Each VM has $\mathcal{R}\ell_k^l$ due to previously assigned tasks.
- VM distances (V_{kr}) among the heterogeneous VMs. V_{kr} actually calculates the distance between V_k and V_r by finding the number of hop counts between them.
- The processing capacity (PC_k) of VMs is in MIPS.
- Once a VM has started task execution, it continues without interruptions, and after completing the execution it immediately sends the output data to all the children tasks in parallel.
- The expected time to compute matrix E_{ijk} is the estimated execution time of task T_{ij} on V_k.

Table 2 Definition of symbols

Symbols	Meaning
$N_{\mathcal{W}f}$	Number of workflows
Wf_i	i^{th} workflow
N_{Wf_i}	Number of tasks in i^{th} workflow
T_{ij}	j^{th} task in the i^{th} workflow
$Succ(T_{ij})$	Set of successor tasks
$Pred(T_{ij})$	Set of predecessor tasks
SD_{ij}	Security demand of T_{ij}
Wl_{ij}	Workload of T_{ij}
n	Number of VMs
V_k	k^{th} VM
l_{ij}	Level attribute of T_{ij}
d^l	Set of depth level of T_{ij}
$N^l_{wf_i}$	Number of tasks in each depth level
C^l_{SD}	Set of clusters of SD at d^l
e_{ixy}	Edge weight between T_{ix} and T_{iy}
V_{kr}	Machine distance between V_k and V_r
V_{Fit}	Fit Virtual Machine
TL_{CS}	Trust level of cloud system
$T_{failure}$	Set of failed tasks
E_{ijk}	Expected time to execution of T_{ij} on V_k
ST_{ijk}	Start Time of T_{ij} on V_k
FT_{ijk}	Finish Time of T_{ij} on V_k
$\mathcal{R}\ell^l_k$	Initial ready time of V_k at starting of the allocation
PC_k	Processing Capacity of V_k
$I\mathcal{g}^l_k$	Idle gap on V_k at level l
$\mathcal{P}\ell^l_k$	Processing time of allocated tasks of level l at the V_k
CC^l_k	Communication cost on the V_k at level l
SO_{ijk}	Security overhead of T_{ij} on V_k
E^{net}_{ijk}	Net processing time of T_{ij} on V_k
NTF	Number of task failure
λ_k	Failure coefficient for V_k
$F\not{p}_{ijk}$	Failure probability of T_{ij} on V_k
$F\not{p}$	Failure probability of entire IaaS System
$T\mathcal{P}\ell^l_k$	Total processing time on V_k at level l
$(T\mathcal{P}\ell^l_k)^u$	Updated total processing time on V_k at level l
Ms	Makespan of the multiple workflows submitted to IaaS system
μ	Service Rate of queuing system at global queue
Γ	Arrival Rate of queuing system at global queue
Q_u	Queuing unit
Qt	Queuing time

The measure of communication cost depends on two parameters e_{ixy} between the tasks (i.e., T_{ix} and T_{iy}) and V_{kr} between the VMs (i.e., V_k and V_r). Suppose two tasks' $T_{ix} \in d^l$ and $T_{iy} \in d^{l-s}$ of the workflow $\mathcal{W}f_i$ assigned on V_k and V_r respectively, is represented by $CC^{l-s,l}_{ixykr}$. It can be computed as [38, 39]:

$$CC^{l-s,l}_{ixykr} = \pi\left(e_{ixy} \times V_{kr}\right) \tag{1}$$

where π is considered as one for liner behaviour, CC^l_k gives the total communication cost of tasks assigned on V_k at depth level l and needs communication to their predecessor tasks allocated to other VMs. This value is zero for tasks with predecessors being assigned on the same VM. In this model, this is taken as the maximum of all $CC^{l-s,l}_{ixykr}$ among tasks assigned on V_k due to having the possibility of parallel communication and written as

$$CC^l_k = \max_{\forall k}(CC^{l-s,l}_{ixykr}) \tag{2}$$

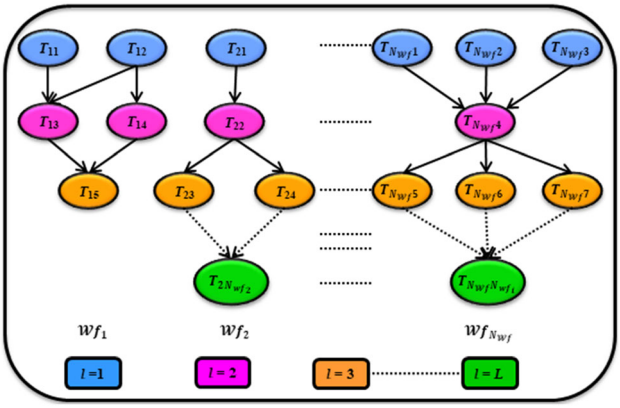

Fig. 1 A sample multiple workflows application

3.4 Security model

Security in cloud computing can be perceived as the protection mechanism against unauthorized access, use, and modification of cloud resources. To safeguard infrastructure of the cloud-based systems from security risk, a variety of rules, processes, controls, and technologies are used. The security risk is related to vulnerabilities, failure probability, and attacks or threats. The cloud environment is risk prone due to challenging security threats. Therefore, it is very crucial to provide the best-in-class security by vendors tailored for the cloud infrastructure. Thus, the cloud vendors use their own unique security standards, methods, and models to satisfy the client's requirements. Further, cloud system security offers many benefits, including centralized security, reduced cost and administration, and reliability. In workflow execution in the IaaS cloud, a workflow management system (WMS) allocates the workflow tasks onto the secure cloud resources so that they could be executed without failures. A secure workflow system requires taking into account a variety of security services for modeling any

security-sensitive applications, such as authentication, integrity, and confidentiality as discussed follows:

- *Authentication*: It refers to trustworthily confirming the task execution agents' identities. Authentication, authorization, and accounting (AAA) is a security organizing module for authentication, authorization, and accounting. When a user tries to access cloud resources from CSP, then AAA verifies the user's authentication information. If the user is authenticated, then AAA verifies the user's access into the system. The authentication methods are available to fulfill the authenticated users, such as HMAC-MD5, HMAC-SHA-1, and CBC-MAC-AES [22, 24, 55].
- *Integrity*: Integrity services guarantee that no one can tamper or modify the data and applications while executing on the IaaS cloud. When the attacker modifies the data, the integrity of the data is compromised. Integrity can be achieved using various hash functions such as Tiger, RIPEMD-160, SHA-1, etc. [28, 55, 57].
- *Confidentiality*: Confidentiality is important for users to store their confidential resources in the cloud. Confidentiality is the defense against eavesdropping and other passive attacks on cloud resources. A passive attacker could disclose that sensitive information is being transferred insecurely or without encryption. Confidentiality can be achieved by using various encryption algorithms like IDEA, DES, etc. [22, 55, 57].

In authentication service, the machines are authenticated and allowed for the workflow task to process them in workflow execution. A workflow authorization model is capable of authorization in such a way that machines have access to necessary objects and synchronize the authorization flow with the workflow during execution. After this, the associated information of workflow tasks needs to be transferred among machines during processing due to communicating and dependent tasks. Hence, it is the service provider's responsibility to make these transfers confidential and unaltered to maintain confidentiality and integrity, respectively. The task failure could result from the security threat or inaccessibility from the security-imposed barricade in the cloud system. In the proposed SPMWA model, authentication service is considered during secure workflow task allocation. Authentication is the initial process which allows entering the authorized machines to satisfy the security requirement for execution of the workflow tasks from the various users. In this way, the cloud system can prevent the tasks failure during the workflow task's executions. The authentication must be validated to protect the data transfer from security attacks to a certain extent according to its service level requirement. For task T_{ij} having a security level (SL^a_{ij}) in the aspect of authentication of m_i^{th} method used, an authentication method providing a higher security level must be applied if the task is outsourced to the cloud such as $SL^a_{ij} \leq Sl^a_{m_i}$ where $Sl^a_{m_i}$ represents the security level provided by the m_i^{th} authentication method ($m_i \in \{0, 1, 2, 3\}$). We also assume that, $\mu^a_{m_i-1} < \mu^a_{m_i}, \forall m_i \in \{0, 1, 2, 3\}$ without loss of generality as shown in Table 3, where $\mu^a_{m_i}$ is the authentication time overhead of m_i. For task T_{ij}, the optimal authentication method that satisfies the security level requirement with the least amount of time overhead is that which meets the following criteria $Sl^a_{m_i-1} < SL^a_{ij} \leq Sl^a_{m_i}$. These authentication methods are shown in Table 3.

Table 3 Authentication methods for authentication services [55, 58]

Authentication methods	Security level $\left(SL^a_{m_i}\right)$	Time overhead $\left(\mu^a_{m_i}\right)$ ms
No Method	0	0
HMAC-MD5	0.55	90
HMAC-SHA-1	0.91	148
CBC-MAC-AES	1	163
Private Cloud	1.00	–

Each authentication method is assigned a security level (SL^a_{mi}), in agreement with the concert. If it is assigned to 1 then CBC-MAC-AES method will be used. *SLs* for the other two methods can be computed as per Eq. (3) where $\mu^a_{m_i}$ is the performance of the m_i^{th} authentication method i.e., $0 \leq m_i \leq 3$.

$$\mathrm{SL}^a_{m_i} = \mu^a_{m_i}/163;\ 0 \leq m_i \leq 3 \tag{3}$$

Assume that SO_{ijk} is the security overhead to fulfill authentication service for workflow tasks and T_{ij} is a function security level. The following characteristics related to the security of multiple workflows and VMs are given below:

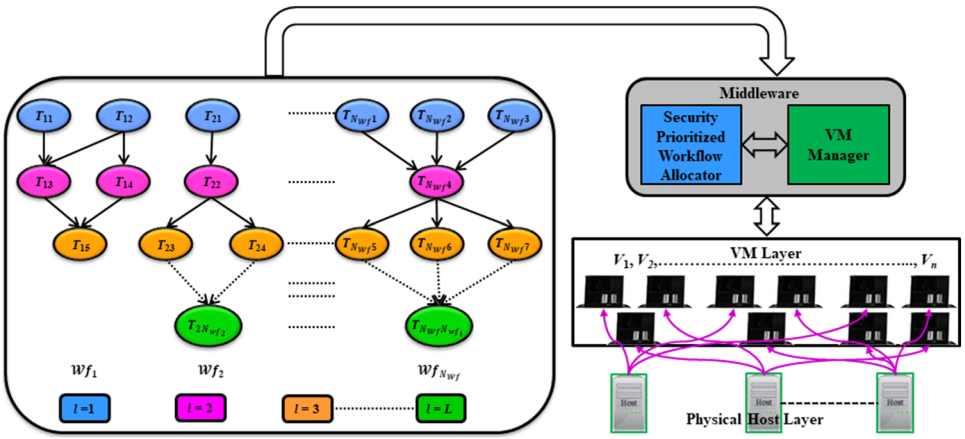

Fig. 2 Multiple workflow allocation problem

- Each T_{ij} has a SD_{ij} that needs to be satisfied on VMs.
- SD_{ij} varies in the range from very high to very low, representing their priority, such that $SD_{ij} \in \{0.0, 0.1, 0.2,\ldots\ldots,1.0\}$. For example, $SD_{ij} = 1.0$, $SD_{ij} = 0.5$, and $SD_{ij} = 0.1$ are very high, average, and low-security demands respectively.
- Each virtual machine (VM_k) has a trust level (TL_k) normalized in the range [0, 1].
- Trust level of a cloud system (TL_{CS}) is defined as the trust level of the highest trustable VM such as $TL_{CS} = \max_{\forall k}\{TL_k\}$.
- The range of failure coefficient, $\lambda_k \in (0.1–5.5)$.

Now, in the allocation process, each task from the cloud users requiring SD_{ij} has been mapped onto fit VMs. The security overhead of T_{ij} on V_k (SO_{ijk}) with security demands can be estimated as per Eq. (4).

$$SO_{ijk} = \begin{cases} \mu^a_{m_i}(SD_{ij} - TL_k)/PC_k; & SD_{ij} > TL_k \\ 0; & SD_{ij} \leq TL_k \end{cases} \quad (4)$$

SD_{ij} is a security demand of T_{ij} that can be specified security levels by the workflow tasks. TL_k is the trust level of V_k for fulfilling the security demand of the tasks. $\mu^a_{m_i}$ is a time overhead considered by the authentication methods [24, 55, 58] to fulfill the security levels of authentication methods. In this workflow risk analysis model, we have to consider failure probability as a function of SLs and the distribution of failure rate for any fixed time interval following the Poisson probability distribution. Accordingly, the workflow's task failure probability for combined security services onto a particular VM can be signified by an exponential distribution as follows [24, 28–30].

$$F\not{\prime}_{ijk} = \begin{cases} 1 - e^{-\lambda_k\left(SD_{ij} - TL_k\right)}; & SD_{ij} > TL_k \\ 0; & SD_{ij} \leq TL_k \end{cases} \quad (5)$$

In cloud computing, λ_k is taken as a failure coefficient, and it varies over VMs. The negative exponent shows the failure probability raised by the difference $SD_{ij} - TL_k$.

3.5 Problem statement and parameter estimation

In this section, the workflow allocation problem and parameter estimation of performance metrics for the proposed model have been discussed in detail. In multiple workflows allocation problem, a set of workflows (Wf) needs to map on a set of n heterogeneous virtual machines (V) in an IaaS cloud computing environment to optimize the number of task failure and failure probability of workflow systems satisfying various constraints. A pictorial representation of the same problem is presented in Fig. 2. Here, multiple workflows are submitted to the cloud from different users. Security prioritized workflow allocator is responsible for capturing all required information and performs high security prioritized workflow tasks allocation effectively. VM manager manages and offers the set of VMs having required objects, i.e., trust level, computing capacity, etc. The physical machine comprises clusters, servers, supercomputers, and the fundamental physical computing resources that make up a cloud infrastructure. Through virtualization, users can use a virtualized version

of machines of the physical machine without any management overhead.

The problem statement can be represented by the mapping function, f: $\mathcal{W}f \times V \rightarrow \{1, 0\}$, producing an allocation schedule such that.

Minimize $F\mathcal{P}$, NTF subject to constraints:

1.
$$\sum_{j=1}^{N_{wf_i}} Allocation[i][j][k] = P; \text{where}, 0 \leq P \leq N_{wf_i}$$

2.
$$\sum_{k=1}^{n} Allocation[i][j][k] = 1$$

3.
$$\sum_{i=1}^{N_{wf}} \sum_{j=1}^{N_{wf_i}} \sum_{k=1}^{n} Allocation[i][j][k] = \sum_{i=1}^{N_{\mathcal{W}f}} N_{wf_i}$$

4.
$$ST_{ijk} \geq max\{FT[pred(T_{ij})]\}$$

The SPMWA model aims to optimize the failure probability and number of task failure as per the requirements of the cloud users. Some of the main parameters considered in the study that is given as follows:

The Expected Time to execution (E_{ijk}) can be written by using Eq. (6) as follows:

$$E_{ijk} = \frac{Wl_{ij}}{PC_k} \tag{6}$$

Now, the allocation of tasks from various depth levels are assigned on the fit virtual machine using the proposed SPMWA model explained in Sect. 4. The Finish Time (FT_{ijk}) of T_{ij} on each V_k is computed by using Eq. (7) as follows:

$$FT_{ijk} = ST_{ijk} + E_{ijk} + SO_{ijk} \tag{7}$$

where ST_{ijk} is the Start Time of T_{ij} on V_k to be assigned task and can be taken as the FT_{ijk} of the last task assigned on that virtual machine. After the allocation of all tasks from the specified partition, i.e., $\forall T_{ij} \in d^l$, the processing time $(\mathcal{P}\ell_k^l)$ on V_k at lth level has been estimated. It is the summation of the difference between the finish time and start time of the tasks of specified partition on VMs and can be written as

$$\mathcal{P}\ell_k^l = \sum_{V_k \leftarrow \forall T_{ij} \in d^l} (FT_{ijk} - ST_{ijk}) \tag{8}$$

$\mathcal{R}\ell_k^l$ is the initial ready time of each VM. It is the previous workloads assigned on them and affects only task allocation for the first level only. Total processing time on V_k $(T\mathcal{P}\ell_k^l)$ are the sum of initial ready time, total communication cost, and processing time on the V_k as presented by using Eq. (9)

$$T\mathcal{P}\ell_k^l = \begin{cases} R\ell_k^l + \mathcal{P}\ell_k^l; \; if \; T_{ij} \in d^1 \\ CC_k^l + \mathcal{P}\ell_k^l; \; otherwise \end{cases} \tag{9}$$

After allotment of $\forall T_{ij} \in d^l$, few idle gaps are left on some VMs. The idle gap list $Ig^l = \{Ig_k^l{:}g_k, \forall \text{ k}\}$ is maintained on the respective VM in various depth levels. The idle gap size (Ig_k^l) at depth level l on V_k can be computed as

$$Ig_k^l = \max_{\forall k}(T\mathcal{P}\ell_k^l) - T\mathcal{P}\ell_k^l \tag{10}$$

After this, $succ(T_{ij})$ from the next depth level partition (d^{l+1}) are selected and suitable tasks are accommodated within the idle gaps generated as per idle gap reduction method presented in Algorithm #3. Further, $T\mathcal{P}\ell_k^l$ is updated due to some additional assignments on VMs. The updated total processing time $(T\mathcal{P}\ell_k^l)^u$ on VM can be written as

$$(T\mathcal{P}\ell_k^l)^u = T\mathcal{P}\ell_k^l + \sum_{\forall T_{ij} \in d^{l+1}}^{L} E_{ijk}^{net} \tag{11}$$

where E_{142}^{net} is the net processing time of successor task which is inserted into idle gaps i.e., $E_{ijk}^{net} = E_{ijk} + SO_{ijk} + CC_{ixykr}^{l-s,l}$. Now, Makespan is the total processing time taken for the submitted batch of workflows. It is estimated as the addition of queuing time and summation of $\max(T\mathcal{P}\ell_k^l)^u$ for all depth levels and written as

$$Ms = Qt + \sum_{l=1}^{L} \max_{\forall k}(T\mathcal{P}\ell_k^l)^u \tag{12}$$

where Qt is the Queuing Time which is estimated as the average waiting time of the multiple workflows in the global queue as

$$Qt = \begin{cases} \dfrac{1}{\mu - \Gamma}; \; if \; \beta \leq Q_u \\ \dfrac{1}{\mu - \Gamma}\left(\dfrac{\beta}{Q_u}\right); \; if \; \beta \geq Q_u \end{cases} \tag{13}$$

where β is the sum of the workloads of all workflow tasks. The virtual machines may not be available to the system when being infected by malicious attacks or intrusions. Many parallel workflows are executed in the risk-prone environment of the cloud computing system. Therefore, it is essential to have some guaranteed security services to execute the workflow tasks with negligible failures. If task failures happen, then it is desired that the number of task failure should be insignificant to prove the security of the

system. Thus, the number of task failure (NTF) is an important QoS parameter to assess how often tasks with a given security requirement being allocated on the VMs exhibit insufficient trust. Also, the failure probability of the workflow system needs to be estimated as one of the main targets. Hence, we do evaluate the proposed SPMWA on security metrics such as the number of task failure and failure probability of the entire batch of workflows in the cloud system. Here, the aim is to ensure that all the tasks must be satisfied on trustable VMs. Yet, in some cases, it is also a possibility to assign some tasks on untrustworthy VMs. So, we defined a task failure set, $T_{failure} = \{T_{ij}: V_k \leftarrow T_{ij}$ such that $SD_{ij} > TL_{k,}$ i.e.,$T_{failure} \subset \cup\tau_i,\ i = 1,\ 2,\ldots N_{wf}\}$, where on allocation T_{ij} assigned on insufficient trustworthy VMs. Thus, total tasks failures can be determined as the size of the set of failure tasks, $o(T_{failure})$ by using Eq. (14) as follows:

$$NTF = o(T_{failure}) \tag{14}$$

The failure probability is the performance metric to assess the rate of failure tasks assigned on a specified virtual machine that could result from the inaccessibility of a security-imposed barricade or severe attack. Thus, the task failure probability of T_{ij} assigned on V_k can be computed as:

$$F\not{\prime}_{(T_{ij})} = \sum_{k=1}^{n} z_{ijk} * F\not{\prime}_{ijk} \tag{15}$$

where z_{ijk} is the assignment vector, indicating whether T_{ij} is assigned on V_k or not, such as

$$z_{ijk} = \begin{cases} 1, \ T_{ij} \ is \ assigned \ to \ V_k \\ 0, \ otherwise \end{cases} \tag{16}$$

The failure probability of the cloud system ($F\not{\prime}$) for a considered batch of the workflow ($\mathcal{W}f$) executed in a risk-prone environment (attack or failure). $F\not{\prime}$ can be computed as:

$$F\not{\prime} = 1 - \prod_{T_{ij} \in \tau} \left(1 - F\not{\prime}_{(T_{ij})}\right) \tag{17}$$

NTF, $F\not{\prime}$, and *Ms* are used as performance metrics to evaluate our proposed security prioritized multiple workflow allocation model presented in Sect. 4.

4 Security prioritized multiple workflow allocation model

In this section, a security prioritized multiple workflow allocation (SPMWA) model with precedence constraints is presented for a cloud computing environment. In this model, more priority is given to high-security demand workflow tasks at allocation, and the main target is to minimize the failure probability of the IaaS cloud system during processing. Workflow tasks are allocated in accordance to a level attribute. After the allocation of each partition, idle gaps are eliminated by inserting suitable tasks from the next partition. An algorithmic template for SPMWA is presented in **Algorithm #1**.

Firstly, the batch of multiple workflows is divided into depth levels before the allocation as mentioned already in Sect. 3.2. In this phase, the allocation of workflow tasks from each partition is assigned in sequential mode onto the set of virtual machines. In each partition, the allocation preference is given to the higher security demand tasks followed by lower ones and assigned to the fit VMs. The fit VM for a specified task is the VM which satisfies its security demand against the trust level offered and also takes the least finish time to execute it. The allocation of task selection from each partition is accomplished as follows:

- Divide the batch of multiple workflows into partitions (d^l) as per depth level. In d^l list, tasks have the same level attribute and can be executed in parallel.
- In each partition, clusters are created by grouping the tasks having the same security demand (SD) $C^l_{SD} = \{T_{ij} : \forall SD_{ij} = SD\}$ where $SD \in \{0.0, 0.1, 0.2\ldots1.0\}$. In this way, at most, eleven clusters can be formed in each partition by grouping the tasks of the same security demands. For allocation, the clusters are selected in higher to lower order of security demand.
- Then, sort each cluster tasks as per the workloads in descending order as shown in **Algorithm #1** in step 4. Workflow's tasks are selected one by one from the cluster. In this way, largest tasks are allocated first.
- Select $T_{ij} \in \boldsymbol{C^l_{SD}}$ form the sorted cluster, then find V_{Fit} by ***Call VM Selection*** () as per step 10 in **Algorithm #1**. **Algorithm #2** selects the best fit machine, $V_{Fit,}$ and returns the status of tasks, Fail equals to 0 or 1. After this NTF is updated as per step 13. Further, the set of failed tasks ($T_{failure}$) is updated as steps 14–17.
- After assignment of $\forall T_{ij} \in d^l$, Compute the value of CC^l_k and $\mathcal{P}\ell^l_k$ as per Eq. (2) and Eq. (8) respectively. Consequently, the communication cost will be zero for the first level of workflow task because it has no parent task.
- After completion of all tasks in each d^l, find the idle gap list and reduce the idle gap by inserting suitable tasks into them by ***Call Idle Gap Reduction*** () in **Algorithm #1**. The procedure of idle gap reduction is explained in **Algorithm #3**.
- Finally, CC^l_k and $(T\mathcal{P}\ell^l_k)^u$ are computed after the allocation of tasks in the idle gap list as per Eq. (2) and Eq. (11) respectively. Finally, a schedule will be generated and QoS parameters are computed.

Algorithm #1: SPMWA ( )

Input: N_{Wf}, N_{wf_i}, Wl_{ij}, e_{ixy}, SD_{ij}, n, TL_k, λ_k, PC_k, $\mathcal{R}t_k$, V_{kr}
Output: A schedule (S) on Wf onto V, $T_{failure}$, NTF, $F\mathcal{P}$, Ms
Begin

1. Compute E_{ijk}, SO_{ijk}, $F\mathcal{P}_{ijk}$ matrices and Qt // *As per eq.* (6), *eq.* (4), *eq.* (5) *and eq.* (13) *respectively*
2. Partition (d^l) formation // *As per depth level*
3. Create Clusters C_{SD}^l in each partition // *The distinct value of security demand*
4. Sort $C_{SD}^l \in d^l$ // *In descending order as according to* Wl_{ij}
5. $NTF = 0$
6. $T_{failure} = \emptyset$
7. ***For*** $\forall\, d^l \in l = 1 - L$ ***do***
8. ***For*** $\forall\, C_{SD}^l$ ***do*** // *Select cluster of higher to lower SD*
9. ***For*** $\forall\, T_{ij} \in C_{SD}^l$ ***do***
10. $[V_{Fit}, Fail]$ = ***Call VM Selection ( )*** // *This returns the fit VM*
11. $V_{Fit} \leftarrow T_{ij} \in C_{SD}^l$
12. $C_{SD}^l = C_{SD}^l - \{T_{ij}\}$
13. $NTF = NTF + Fail$
14. ***If*** ($Fail$==1)
15. $T_{failure} = T_{failure} \oplus T_{ij}$ // T_{ij} *is included in set of failed tasks*
// T_{ij} *is executed on* V_{Fit} *with some failure probability*
16. ***Else***
// T_{ij} *is executed on* V_{Fit} *with zero failure probability*
17. ***End If***
18. Compute $F\mathcal{P}_{ijk}$ // *As per eq.* (5)
19. ***End For***
20. ***End For***
21. Compute CC_k^l and $T\mathcal{P}t_k^l$ // *As per eq.* (2) *and eq.* (9) *respectively*
22. ***Call Idle Gap Reduction ( )***
23. Update $(T\mathcal{P}t_k^l)^u$ // *As per eq.* (11)
24. ***End For***
25. Find $T_{failure}$, NTF, $F\mathcal{P}$, and Ms // *As per eq.* (14), *eq.* (17), *and eq.* (12) *respectively*

End

4.1 VM selection

In this part, the VM selection procedure is presented for the task at hand for mapping. The VM selection aims to find a fit VM for each task assignment that satisfies the security demands of tasks against the trust levels of VMs. Initially, the security requirement of the task to be assigned (SD_{ij}) is compared to the trust level of the cloud system, i.e., $TL_{CS} = max\ (TL_k,\ k = 1,2,\ldots,n)$. The VM selection procedure works into two cases as follows:

Case 1 (If $SD_{ij} \leq TL_{CS}$): It implies that the cloud system has a set of VMs having trust levels greater than or equal to the security demand of the task. These trustable VMs can fully satisfy the security demand of the task and can execute it without any risk of failure. Further, the best fit virtual machine (V_{Fit}) is the VM which is a fully trustworthy machine offering the least finish time for the task among all fit VMs (step 4, Algorithm #2). The VM selection procedure returns V_{Fit} and that can execute the task without any risk or zero failure probability as the requirement is satisfied fully. Hence, the variable *Fail* returns 0 indicating the task is not failed. Thus, in this case, all the task is assigned onto a fully trustworthy machine as shown in Algorithm #2 as per steps 2–5.

Case 2 (If $SD_{ij} > TL_{CS}$): The cloud system is capable enough of providing the services on demand with elasticity. It implies that the cloud is able to supply the required trustable resources for processing irrespective of the place, time, and amount demanded. Yet there may be some cases, often not happen, when the task's security demand is not

fully satisfied against the trust level of the VM available. It means that the cloud system has a set of VMs having a trust level lesser than the security demand of the task. In this scenario, our SPMWA model finds the highest trustworthy machine from the cloud system (max TL_{CS}). In such a special case, this is rarely happen at the cloud platform as cloud systems are committed to supply the needed resources on demand. Then, the SPMWA model finds the highest trustworthy VM in the system (steps 6–13, Algorithm #2). For this, the procedure tries to find the VM against the security demand of the tasks by decrementing it by one level. This process is continued till the updated SD_{ij} reaches TL_{CS} so that, the task could be assigned on the highest trustworthy available VM (steps 9–11, Algorithm #2). In this way, the failure probability can be minimized to a possible minimum value. The task is considered partially failed (*Fail* is set to 1) and counted as task failure.

Thus, the SPMWA model tries to assign each task to the highest trustworthy machine in the cloud system so that the failure probability is either zero or minimal. All the number of task failure (NTF) are partially failures, not fully failures. The same stepwise VM selection procedure is presented in Algorithm #2.

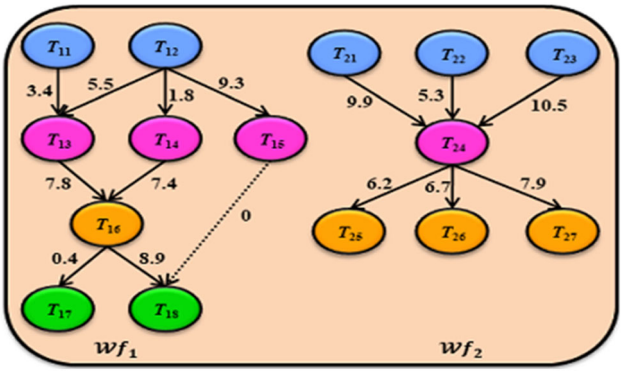

Fig. 3 A sample of two multiple workflows application

visualization, we have shown in the illustration Sect. 4.1 with Fig. 4. At first, after allocation of all workflow tasks at each level computes $max\left(\mathcal{TP}\ell_k^l\right) \in d^l$. Now, determine the idle gap list, $Ig^l = \left\{Ig_k^l : g_k, \forall\ \mathrm{k}\right\}$ at each level as per Eq. (10). For insertion of tasks from next levels/partitions, i.e., $T_{ij} \in d^{l+1}$, we need to find out the start time of to be inserted tasks, the finish time of their predecessor, and the communication cost between them. To find out the best fitted idle gap for the tasks at hand are being allocated, two

Algorithm #2: $[V_{Fit}, Fail]$ **= VM Selection** $(V, \mathcal{R}t_k, TL_k, V_{kr}, \lambda_k, T_{ij}, e_{ixy}, SD_{ij}, E_{ijk}, SO_{ijk}, ST_{ijk}, FT_{ijk})$

1. $Fail = 0$
2. **Case 1** *//assigned task with zero failure probability*
3. ***If*** $(SD_{ij} \le TL_{CS})$
4. $V_{Fit} \leftarrow \underset{SD_{ij} \le TL_k \,\&\&\, min(FT_{ijk})}{V_k}$
   // V_{Fit} satisfies the security demand fully and also offers least finish time among all
5. *Return* V_{Fit}
6. ***Else***
7. **Case 2** *//assigned task with minimum failure probability*
8. ***While*** $(SD_{ij} > TL_{CS})$
9. $SD_{ij} = SD_{ij} - 0.1$
   // Decrement security demand level upto TL_{CS}.
   // Still, T_{ij} is assigned on the highest trustworthy available VM in the system.
10. ***End While***
11. $Fail = 1$ *//Set flag fail to* 1
12. $V_{Fit} \leftarrow \underset{SD_{ij} \le TL_k \,\&\&\, min(FT_{ijk})}{V_k}$
    // V_{Fit} satisfies the security demand partially and also offers least finish time among all
13. *Return* V_{Fit}
14. ***End If***

4.2 Idle gap reduction

The idle gap reduction begins after the allocation of all workflow tasks on fit VMs at each partition. The idle gaps on VM at any given depth level are reduced by inserting the suitable tasks from the next level $succ(T_{ij})$. For better conditions must satisfy, (i) The start time of inserted tasks should be greater than or equal to the finish time of their predecessors i.e., $ST_{ijk} \ge FT_{ixk}$ (ii) The gap size must be greater than or equal to the expected time to complete the inserted task such that $g_k \ge E_{ijk} + SO_{ijk} + CC_{ijxkr}^{l,l+1}$ as per steps 5–8 in Algorithm #3. Idle gaps (Ig_k^l) has been reduced

by allocating the task with fulfilling the security demand and size of the task from the successor level so that these tasks may be adjusted in the gaps onto V_{Fit}. The workflow tasks have been inserted to preserve the precedence constraints of the multiple workflows. For any Ig_k^l, the neighboring tasks in the workflow tasks assigned to that VM are considered so that communication overhead can be minimized by the insertion. This phase avoids VMs idle gaps as much as probable, resulting in optimizing the makespan of the system. The idle gap reduction phase algorithm is presented in Algorithm #3.

Table 4 The heterogeneous VMs parameters

Number of VM (n)		V_1	V_2	V_3
Processing capacity (PC_k)		10	08	15
Ready time ($\mathcal{R}\ell_k$)		24	10	20
Trust level (TL_k)		0.4	0.6	0.7
Failure coefficient (λ_k)		1.7	1.1	1.5
V_{kr}	V_1	0	5	2
	V_2	5	0	1
	V_3	2	1	0

Algorithm #3: ***Idle Gap Reduction*** (d^l, d^{l+1}, V, $\mathcal{R}t_k$, TL_k, V_{kr}, λ_k, T_{ij}, e_{ixy}, SD_{ij}, E_{ijk}, SO_{ijk}, ST_{ijk}, FT_{ijk}, CC_k^l, $T\mathcal{P}t_k^l$, NTF)

1. Compute $max\left(T\mathcal{P}t_k^l\right)$
2. Find Ig^l list // *As per eq.* (10)
3. ***For*** $\forall\, T_{ij} \in d^{l+1}$ ***do*** // *Find all* $T_{ix} \in pred\,(T_{ij}) \in d^l$
4. ***For*** each V_k ***do***
5. ***If*** $\left(ST_{ijk} \geq FT_{ixk}\ \&\&\ g_k \geq E_{ijk} + SO_{ijk} + CC_{ijxkr}^{l,l+1}\right)$
   // T_{ij} must start after its all-predecessor tasks ($T_{ix} \in pred\,(T_{ij}) \in d^l$)
   // Idle gap size should be fitted for T_{ij}
6. $[V_{Fit}, Fail]$ = ***Call VM Selection*** ( )
7. $V_{Fit} \leftarrow T_{ij}$
8. $Ig_k^l = Ig_k^l \oplus \{g_k - E_{ijk}^{net}\}$

   // *Update idle gaps list by subtracting the net processing time* (E_{ijk}^{net}) *of inserted task from the idle gap at VM*
   // E_{ijk}^{net} *is sum of execution time, security overhead and communication cost, i.e.* $E_{ijk}^{net} = E_{ijk} + SO_{ijk} + CC_{ijxkr}^{l,l+1}$
9. $NTF = NTF + Fail$
10. ***End If***
11. ***End For***
12. ***End For***

4.3 An illustrative example

An illustration has been demonstrated for a better understanding of the SPMWA model. We have considered a virtual machine set with three instances, $V = \{V_1, V_2, V_3\}$ for illustration purposes. The various characteristics namely, processing capacities, ready time, trust levels, failure coefficients, and machine distances associated to the cloud VMs are presented in Table 4. Now, the trust level of a cloud system, $TL_{CS} = max(TL_1, TL_2, TL_3) = max(0.4, 0.6, 0.7) = 0.7$.

Further, we consider two workflows, $\mathcal{W}f_1$ and $\mathcal{W}f_2$, consisting of 8 and 7 tasks, respectively. And depth levels are 4 and 3 respectively, as shown in Fig. 3. The corresponding edge weights (inter-task communication) between tasks have been shown by edge labels in Fig. 3. For example, 3.4 is the edge weight (e_{113}) between the tasks T_{11} and T_{13} of $\mathcal{W}f_1$. Now, the workflow tasks attributes, such as level attribute (l_{ij}), workload (Wl_{ij}), and security demand (SD_{ij}) of each task are presented in Table 5. The information regarding the workflow's tasks have been shown for the three VMs in Table 5. Let $\mu_{m_i}^a$= 90 ms, service rate (μ) = 0.05, arrival rate (Γ) = 0.03, and queue unit (Q_u) = 10,000 MIs. The queuing time (Qt) = 50 for the batch of workflows which is computed by using Eq. (13).

The expected time to compute (E_{ijk}), security overhead (SO_{ijk}), and failure probability ($F\not{p}_{ijk}$) of each task are computed onto all VMs as per Eqs. (6), (4), and (5), respectively. And the same values are presented in Table 6.

Table 5 The workflow tasks information

$\mathcal{W}f_1$				$\mathcal{W}f_2$			
T_{ij}	l_{ij}	Wl_{ij}	SD_{ij}	T_{ij}	l_{ij}	Wl_{ij}	SD_{ij}
T_{11}	1	30	0.8	T_{21}	1	20	0.7
T_{12}	1	12	0.4	T_{22}	1	25	0.4
T_{13}	2	39	0.9	T_{23}	1	10	0.7
T_{14}	2	50	0.6	T_{24}	2	90	0.6
T_{15}	2	120	0.5	T_{25}	3	55	0.4
T_{16}	3	48	0.7	T_{26}	3	120	0.6
T_{17}	4	30	0.4	T_{27}	3	125	0.7
T_{18}	4	40	0.7	–			

At first, the SPMWA divides the workflows shown in Fig. 3, into the four partitions (d^l) as (T_{11}, T_{12}, T_{21}, T_{22}, T_{23}), (T_{13}, T_{14}, T_{15}, T_{24}), (T_{16}, T_{25}, T_{26}, T_{27}) and (T_{17}, T_{18}), according to depth levels 1, 2, 3, and 4 respectively. Afterward, in each partition, the various clusters with distinct security demands of the workflow tasks are created. Then, each cluster is sorted as per Wl_{ij} of the tasks in descending order. For example, the first partition's tasks have three distinct security demands i.e., 0.8, 0.7, and 0.4. So, three clusters are created from the first partition (d^1), namely,$C^1_{0.8}$, $C^1_{0.7}$, and $C^1_{0.4}$ and sorted according to their workloads. Thus, the remaining partitions are also treated in a similar manner. In allocation, SPMWA gives higher priority to the clusters consisting of high-security demand tasks. It implies that the higher-level clusters are allocated prior to the lower ones. In any cluster, the tasks allocation order is maintained as per the workload by sorting them in descending order. So, the larger task gets assigned before the smaller tasks in the specified cluster. As we know that SPMWA follows the level/partition-wise tasks allocation policy. Hence, the complete allocation order of clusters in the partitions and their associated tasks within the specified clusters are presented as follows:

$$d^1 = \{C^1_{0.8} = \{T_{11}\}, C^1_{0.7} = \{T_{21}, T_{23}\}, C^1_{0.4} = \{T_{22}, T_{12}\}\}$$

$$d^2 = \{C^2_{0.9} = \{T_{13}\}, C^2_{0.6} = \{T_{24}, T_{14}\}, C^2_{0.5} = \{T_{15}\}\},$$

$$d^3 = \{C^3_{0.7} = \{T_{27}, T_{16}\}, C^3_{0.6} = \{T_{26}\}, C^3_{0.4} = \{T_{25}\}\} \text{ and}$$

$$d^4 = \{C^4_{0.7} = \{T_{18}\}, C^4_{0.4} = \{T_{17}\}\}$$

At first, the ready time ($\mathcal{R}\ell_k$), i.e. (24, 10, 20) acts as the start time (ST_{ijk}) of T_{ij} on respective VMs as shown in Table 7. Now, the cluster from the first partition (d^1) with the highest security demand i.e., $C^1_{0.8}$={T_{11}}, is selected for allocation having only one task. As we can see in Table 7, the finish times of T_{11} on VMs are computed by using Eq. (7). Now, T_{11} has security demand, (SD_{11}) = 0.8 and the cloud system can offer maximum trust level, TL_{CS} = max (0.4, 0.6, 0.7) = 0.7. As, $SD_{11} > TL_{CS}$, it means the cloud system has no VM which can completely satisfy SD_{11} and can execute it risk-free. Therefore, T_{11} decrements its security demand by one level to approach its value to the highest trustable VM in the system i.e., TL_3 = 0.7 (steps 7–9, Algorithm #2). Algorithm #2 returns $V_{Fit} = V_3$ and $Fail = 1$ for T_{11}. While we see that V_2 offered the least finish time, $FT_{112} = 16$. Yet, SPMWA determines V_3 as the fit machine for T_{11} by giving priority to the security requirement over completion time, (i.e., $TL_3 = 0.7$ is higher than $TL_2 = 0.6$). Thus, T_{11} is assigned on V_3 starting from $ST_{113} = 20$ and finishing at

Table 6 Computed values of E_{ijk}, SO_{ijk}, and $F\not{r}_{ijk}$ for the tasks on respective VMs

T_{ij}	$E_{i,j1}$	$E_{i,j2}$	E_{ij3}	SO_{ij1}	SO_{ij2}	SO_{ij3}	$F\not{r}_{ij1}$	$F\not{r}_{ij2}$	$F\not{r}_{ij3}$
T_{11}	3	3.75	2	3.6	2.25	0.6	0.4933	0.1975	0.1393
T_{12}	1.2	1.5	0.8	0	0	0	0	0	0
T_{13}	3.9	4.875	2.6	4.5	3.375	1.2	0.5726	0.2811	0.2592
T_{14}	5	6.25	3.33	1.8	0	0	0.2882	0	0
T_{15}	12	15	8	0.9	0	0	0.1563	0	0
T_{16}	4.8	6	3.2	2.7	1.125	0	0.3995	0.1042	0
T_{17}	3	3.75	2	0	0	0	0	0	0
T_{18}	4	5	2.667	2.7	1.125	0	0.3995	0.1042	0
T_{21}	2	2.5	1.33	2.7	1.125	0	0.3995	0.1042	0
T_{22}	2.5	3.125	1.667	0	0	0	0	0	0
T_{23}	1	1.25	0.667	2.7	1.125	0	0.3995	0.1042	0
T_{24}	9	11.25	6	1.8	0	0	0.2882	0	0
T_{25}	5.5	6.875	3.667	0	0	0	0	0	0
T_{26}	12	15	8	1.8	0	0	0.2882	0	0
T_{27}	12.5	15.625	8.33	2.7	1.125	0	0.3995	0.1042	0

Table 7 Illustration of allocation of tasks at depth level = 1

Tasks	Start Time			Finish Time			Assigned VM	Status
T_{ij}	ST_{ij1}	ST_{ij2}	ST_{ij3}	FT_{ij1}	FT_{ij2}	FT_{ij3}		
T_{11}	24	10	**20**	30.60	16	**22.60**	V_3	**F**
T_{21}	24	10	**22.60**	28.70	13.63	**23.93**	V_3	NF
T_{23}	24	10	**23.93**	27.70	12.38	**24.60**	V_3	NF
T_{22}	24	**10**	24.60	26.50	**13.13**	26.26	V_2	NF
T_{12}	24	**13.13**	24.60	25.20	**14.63**	25.40	V_2	NF

Various symbols meaning for Tables 7-12
: Security demand T_{ij} is fully satisfied on respective *VM*
F: Failure
NF: Non-Failure

FT_{113} = 22.60. The task failure set is updated by including T_{11}, $T_{failure} = \{T_{11}\}$ (step 15, Algorithm #1). Here, T_{11} assignment is considered as the allocation with task failure (**F**) having failure probability 0.1393 computed as per Eq. (5). The start time and finish time of T_{11} are presented without a rectangle box. In Table 7, the numerical values of the start and finish time of V_{Fit} (best fit VM for allocation) for respective tasks are shown in bold. And, the values shown in rectangle boxes represent that corresponding VMs can fully satisfy the security demand of the specified task and vice versa.

Again, the next cluster is taken for allocation, $C^1_{0.7} = \{T_{21}, T_{23}\}$ having two tasks. For, T_{21}, the finish time is computed again by using Eq. (7) and presented in Table 7. Here, $SD_{21} = TL_{CS}$. It means T_{21} can be executed without any failure on the cloud system. As per the allocation process mentioned earlier, $V_{Fit} = V_3$ with $TL_3 = 0.7$ against $SD_{21} = 0.7$. So, T_{21} is assigned and executed on V_3 without any risk of failure (i.e., $F\not{p}_{213} = 0$). The status of T_{21} on allocation is Non-failure (NF). Finally, the start and finish times are made bold in a rectangle box. T_{23} is also assigned on V_3 similar to T_{21} with status NF. For cluster, $C^1_{0.4} = \{T_{22}, T_{12}\}$, both tasks have a security demand is 0.4. The trust levels of all VMs are greater than or equal to 0.4. Therefore, these tasks can be allocated and executed on any VM without risk. However, as per SPMWA for both tasks (T_{22} and T_{12}) fit machine is $V_{Fit} = V_2$ as V_2 offers the least finish time among all as shown in Table 7. Hence, T_{22} and T_{12} are assigned on V_2 with status as NF. In this way, the first-level tasks are assigned. At this level, only the T_{11} task is executed with risk due to the unavailability of VM, and the rest of the tasks are executed risk-free with zero failure probability.

As shown in Table 8, the processing time and total processing time on V_k at the first depth level are computed as per Eqs. (8) and (9) respectively. The communication cost values are zero for first-level tasks on all VMs. After this, the idle gap reduction (Algorithm **#3**) procedure finds the idle gaps $(I\mathscr{g}^1_k)$ on V_k by using Eq. (10) and presented in Table 8. The successor tasks of d^1 from the next partition (d^2) (T_{13}, T_{24}, T_{14}, and T_{15}) are taken and tried to accommodate into the suitable idle gaps in accordance to Algorithm #3. Consequently, only $T_{14} \in d^2$ is fitted in the idle gap ($I\mathscr{g}^1_2 : 9.97$) on V_2 because the net processing time of T_{14} is less than the size of the idle gap $(I\mathscr{g}^1_2)$ on V_2 i.e., $E^{net}_{142} = E_{142} + SO_{142} + max(CC^{12}_{14222}) = 6.25 + 0 + 0 = 6.25 < 9.97$. Here, CC^{12}_{14222} is computed by using Eq. (1). After that, SPMWA computes $(T\mathcal{P}\ell^1_k)^u$ as per Eq. (11) and given in Table 8.

The level-wise tasks allocation and idle gap reduction along with ready time, communication cost, processing time, and maximum updated processing time on virtual machines are also presented in Fig. 4 for better clarity. The illustration of allocation of tasks and respective idle gap reduction for remaining partitions have been presented in Tables 9, 10, 11, 12 presenting various intermediary computations following the similar procedure as discussed earlier for the first partition tasks. In the second partition, only one task T_{13} is allocated to fit VM with risk of failure, and the other remaining tasks across the partitions are assigned to their corresponding fit machine without risk or zero failure probability.

On allocation of all workflows, the set of failed tasks is $T_{failure} = \{T_{11}, T_{13}\}$ with the total NTF = 2 as per Eq. (14). The failure probability of the set of workflows on cloud system is computed as $F\not{p} = 0.3623$ by using Eq. (17). As it is observed from the above illustration that the number of task failure is only two and happened only due to the

unavailability of the demanded VM in the cloud system. As per Eq. (12), makespan (Ms) of the set of workflows can be computed as sum of maximum $(T\mathcal{P}\ell_k^l)^u$ on each depth level presented in Tables 8, 10, 12, and Fig. 4. $Ms = 50 + (24.60 + 15.30 + 21.70) = 50 + 61.60 = 111.60$ units. Therefore, it is expected that on larger workflow sets and VMs the proposed model would exhibit better performance behavior.

4.4 Time complexity analysis

The computational time of the proposed model for parallel multiple workflows is defined in terms of the number of workflows ($N_{\mathcal{W}f}$), depth level (L), number of tasks (N_{wf_i}) and edges weight between (e_{ixy}) in a single workflow, and the number of VMs (n). The time complexity of the SPMWA model is the computational time taken for complete execution, as a function input parameter which is discussed in various steps as follows:

- Partitioning: The $\mathcal{W}f$ of depth level is divided into L partitions. As per the algorithmic template of SPMWA, the time complexity for this process can be computed as $O\left(L \times N_{\mathcal{W}f} \times N_{wf_i}\right)$.
- Sorting: Partitions are sorted as per security demand in descending order with time complexity $O(N_l log N_l)$,
- Allocation: The computational time of task allocation for each partition and adjusting the idle gaps by

Table 8 Illustration of idle gap reduction for depth level = 1

Parameter	V_1	V_2	V_3	Remarks
CC_k^1	0	0	0	No communication as first-level tasks have no parents
$\mathcal{P}\ell_k^1$	24	14.63	24.60	Computed as per Eq. (8)
$T\mathcal{P}\ell_k^1$	24	14.63	**24.60**	Computed as per Eq. (9)
$I\mathcal{g}_k^1$	0.60	9.97	0	Idle gaps by subtracting, $max(T\mathcal{P}\ell_k^1) - T\mathcal{P}\ell_k^1$ as per Eq. (10). T_{14} is inserted into $I\mathcal{g}_2^1$ as its E_{142}^{net} is less than $I\mathcal{g}_2^1$ i.e., $6.25 < 9.97$
$(I\mathcal{g}_k^1)^u$	×	**3.72**	×	$I\mathcal{g}_k^1$ is updated by subtracting net processing time from $I\mathcal{g}_2^1$ i.e., $9.97 - 6.25 = I\mathcal{g}$ 3.72
$(T\mathcal{P}\ell_k^1)^u$	24	20.86	**24.60**	$(T\mathcal{P}\ell_k^1)^u$ is updated as per Eq. (11), wherever tasks are inserted

Bold denotes the maximum of all VMs

× : No task can fit in the idle gap due to the smaller gap size

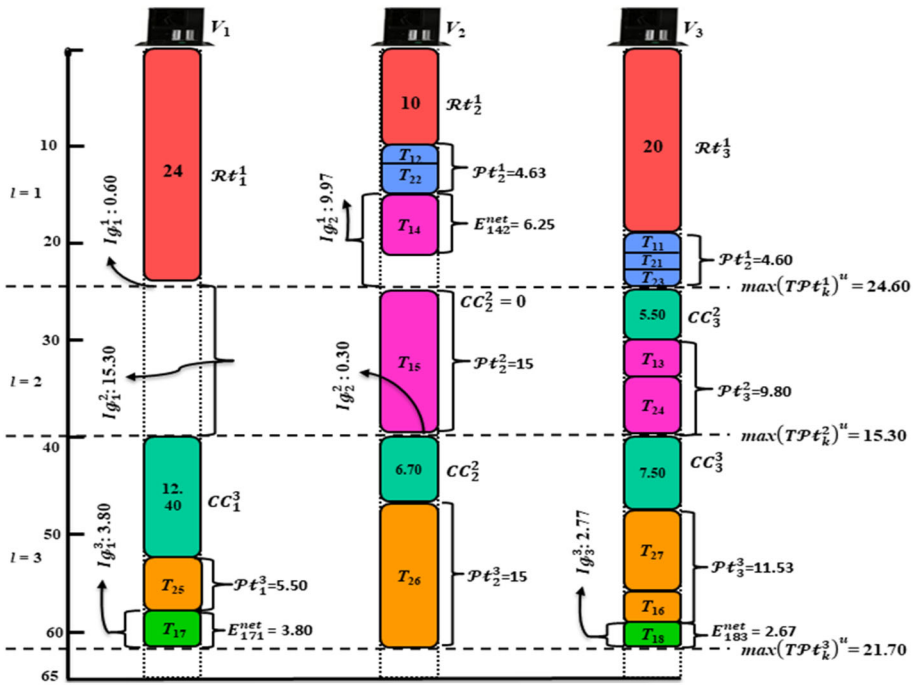

Fig. 4 Allocation Schedule by using the SPMWA model

Table 9 Illustration of allocation of tasks at depth level = 2

Tasks	Start Time			Finish Time			Assigned VM	Status
T_{ij}	ST_{ij1}	ST_{ij2}	ST_{ij3}	FT_{ij1}	FT_{ij2}	FT_{ij3}		
T_{13}	24.60	24.60	24.60	33.00	32.85	**28.40**	V_3	**F**
T_{24}	24.60	24.60	28.40	35.40	35.87	**34.40**	V_3	NF
T_{15}	24.60	24.60	34.40	37.50	**39.60**	42.40	V_2	NF

Various symbols meaning for Tables 7-12
: Security demand T_{ij} is fully satisfied on respective *VM*
F: Failure
NF: Non-Failure

Table 10 Illustration of idle gap reduction for depth level = 2

Parameter	V_1	V_2	V_3	Remarks
CC_k^2	0	0	5.50	Total communication cost CC_k^2 is computed as per Eq. (2)
$\mathcal{P}\ell_k^2$	0	15	9.80	$\mathcal{P}\ell_k^2$ is computed as per Eq. (8)
$T\mathcal{P}\ell_k^2$	0	15.00	**15.30**	$T\mathcal{P}\ell_k^2$ is computed as per Eq. (9)
$I\mathcal{g}_k^2$	15.30	0.30	0	Find Idle gaps as per Eq. (10)
$(I\mathcal{g}_k^2)^u$	×	×	×	× : No task can fit in the idle gap due to the small gap sizes
$(T\mathcal{P}\ell_k^2)^u$	0.00	15.00	**15.30**	$(T\mathcal{P}\ell_k^2)^u$ is computed as per Eq. (11)

Bold denotes the maximum of all VMs

Table 11 Illustration of allocation of tasks at depth level = 3

Tasks	Start Time			Finish Time			Assigned VM	Status
T_{ij}	ST_{ij1}	ST_{ij2}	ST_{ij3}	FT_{ij1}	FT_{ij2}	FT_{ij3}		
T_{27}	39.90	39.90	39.90	55.10	56.65	**48.23**	V_3	NF
T_{16}	39.90	39.90	48.23	47.40	47.02	**51.43**	V_3	NF
T_{26}	39.90	39.90	51.43	53.70	**54.90**	59.43	V_2	NF
T_{25}	39.90	54.90	51.43	**45.40**	61.77	55.09	V_1	NF

Various symbols meaning for Tables 7-12
: Security demand T_{ij} is fully satisfied on respective *VM*
F: Failure
NF: Non-Failure

checking the fit VM created during the allocation phase is in the same order. The allocation of each task checks the fit VM using ***IF-Else*** condition can be done in time $O(1)$. In the same way, each task allocation assigned onto the set of VMs can be done in $O(L \times N_l \times n)$.

Hence, the total time complexity for SPMWA model is $O(L \times N_{\mathcal{W}f} \times N_{wf_i}) + O(N_l log N_l) + O(L \times N_l \times n) \cong O(N_{\mathcal{W}f} \times N_l log N_l)$ where N_l is the number of tasks in l^{th} partition, $N_l \approx N_{wf_i}, N_{\mathcal{W}f} > n$ and $L = log N_l$.

Table 12 Illustration of idle gap reduction for depth level = 3

Parameter	V_1	V_2	V_3	Remarks
CC_k^3	12.40	6.70	7.40	CC_k^3 is computed as per Eq. (2)
$\mathcal{P}\ell_k^3$	**5.50**	15	11.53	$\mathcal{P}\ell_k^3$ is computed as per Eq. (8)
$T\mathcal{P}\ell_k^3$	17.90	**21.70**	18.93	$T\mathcal{P}\ell_k^3$ is computed as per Eq. (9)
$I\mathcal{g}_k^3$	3.80	0	2.77	Finds idle gaps as per Eq. (10). T_{17} & T_{18} are inserted into $I\mathcal{g}_1^3$ and $I\mathcal{g}_3^3$ on V_1 & V_3 respectively
$(I\mathcal{g}_k^3)^u$	0.00	×	0.10	$I\mathcal{g}_1^3$ and $I\mathcal{g}_3^3$ is updated by subtracting net processing times, ($E_{171}^{net} = 3.80$) and ($E_{183}^{net} = 2.67$)
$(T\mathcal{P}\ell_k^3)^u$	**21.70**	**21.70**	21.60	$(T\mathcal{P}\ell_k^3)^u$ is computed as per Eq. (11)

Bold denotes the maximum of all VMs

5 Performance evaluation

In this section, to evaluate the performance of SPMWA, simulation experiments have been conducted on a system having the configuration, Intel (R) i7-8700 CPU@3.20 GHz, 64 GB RAM on a single physical machine by implementing a workflow allocator prototype in MATLAB 8.5. We compare the performance of the SPMWA model with four standard workflow allocation models such as HEFT, LBSIR, SPM1, and SPM2. HEFT [11] and LBSIR [39] are designed to generate time effective schedule without considering the security requirements of the workflow tasks. In the literature section, we have seen that HEFT is rank based list scheduling heuristic that is still competitive for workflow allocation problems. HEFT works in two phases as follows: in the priority phase, it determines the upward rank for maintaining the precedence of the tasks, and in the resource selection phase, it selects the resources which have the earliest finish time.

Moreover, LBSIR is a level-based heuristic for multiple workflow allocation problem to optimize total completion time. After partitioning of workflow tasks in accordance to level attribute, LBSIR also works in two phases viz. allocation and idle slot reduction. In the allocation phase, the mapping of the task is done on the machine that offers the least execution time. In the second phase, best-fitted successor tasks will be accommodated into the idle slots left during the allocation phase between two tasks at the same machines getting a better quality of the schedule. In each partition, the selection of workflow tasks is accomplished in two ways, largest module selection (LMS) and smallest module selection (SMS), resulting in two variants of LBSIR namely LBSIR with largest module selection (L-LMS) and LBSIR with smallest module selection (L-SMS) respectively. Therefore, we are considering them for performance evaluation in our work.

As mentioned earlier in Sect. 2.3 in detail, the work presented by Bittencourt and Madeira [35] suggested four strategies to manage the multiple workflow allocation. For performance evaluation, we have taken two strategies namely sequential-based and merge-based multiple workflows strategies proposed in [35]. The security upward ranks are computed following the method presented in [24]. Further, the security prioritized VM selection and allocation has been done in accordance to SPHEFT [32]. This way, two versions of Security Prioritized Multiple workflow allocation (SPM) models namely SPM1 (Merge-Based) and SPM2 (Sequential-Based) respectively have been designed for comparison purposes.

5.1 Parameter setting

In this section, an experimental study has been carried out for random DAGs and real application DAGs such as Montage, CyberShake, and LIGO, and the results are presented in Sect. 5.2 and Sect. 5.3 respectively. The parameters/variables associated to workflow applications and heterogeneous VMs in IaaS cloud environments are randomly generated in the specified range using a uniform

Table 13 Common input parameters for the experiments

S. No	Input parameters	Range
1	Expected Time to execution (E_{ijk})	1–300,000
2	Edge weight (e_{ixy})	1–3,000
3	Security demand (SD_{ij}) of T_{ij}	0.0–1.0
4	Machine distance (V_{kr})	1–100
5	Communication cost (CC_k^l)	1–300,000
6	Ready time ($\mathcal{R}\ell_k$)	1–1,000
7	Trust level (TL_k)	0.1–0.6
8	Failure coefficient (λ_k)	0.1–5.5
9	Service rate (μ)	0.05
10	Arrival rate (Γ)	0.03
11	Queue unit (Q_u)	20,000

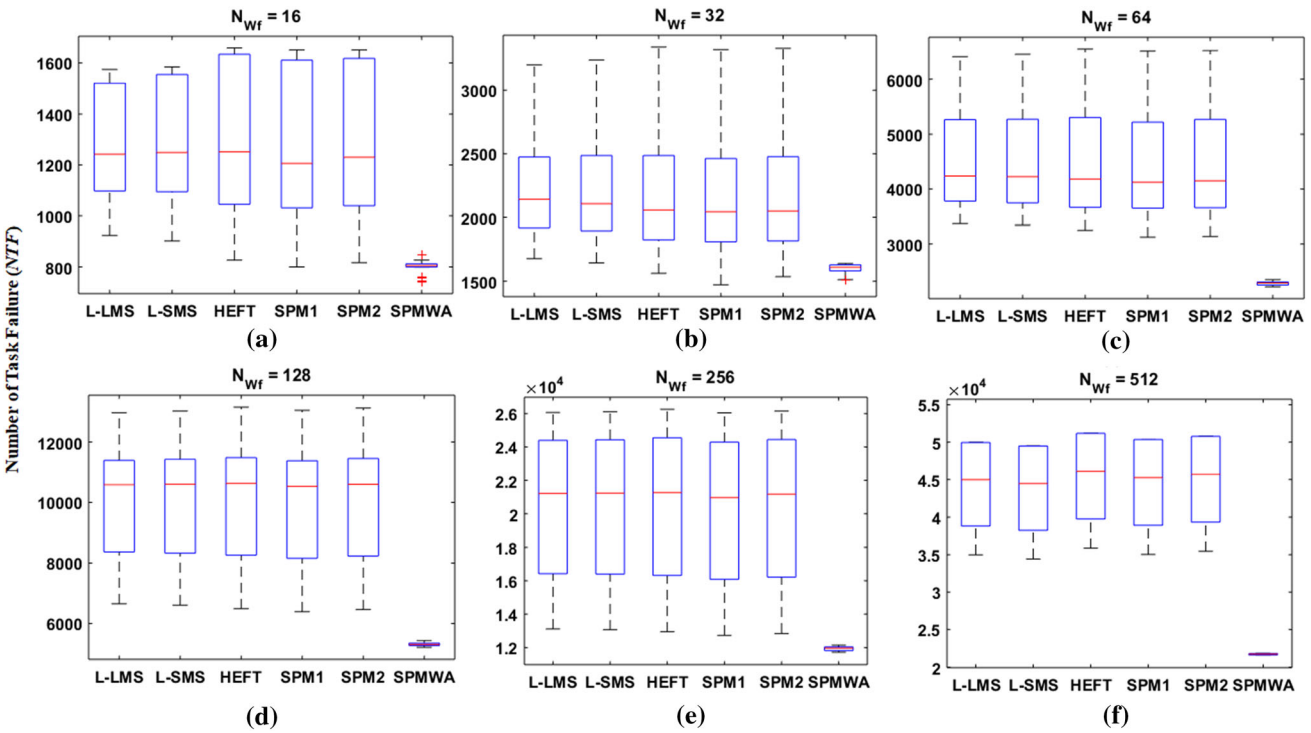

Fig. 5 Boxplots of NTF on varying number of workflows

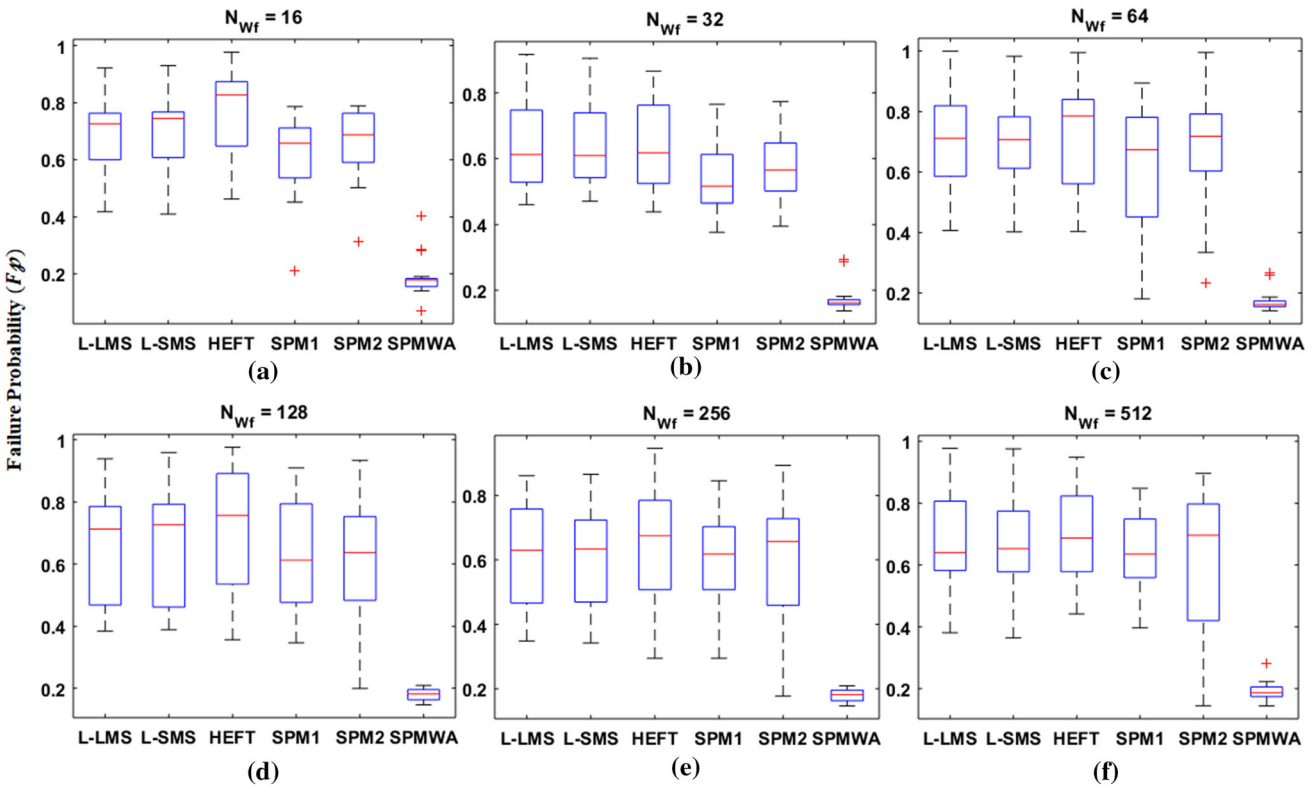

Fig. 6 Boxplots of failure probability on varying number of workflows

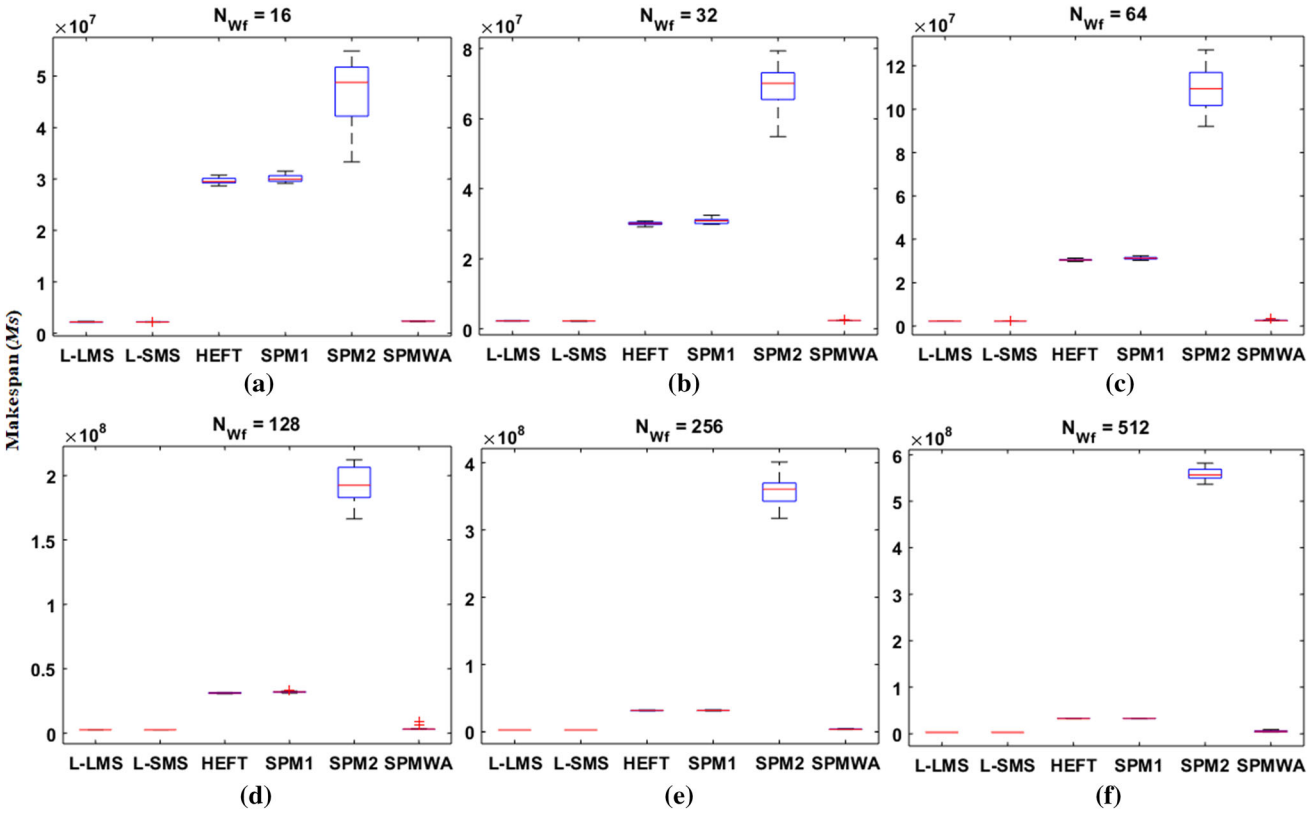

Fig. 7 Boxplots of Makespan on varying number of workflows

distribution. The security demand of the task and the trust level of the virtual machine are in the normalized range [0, 1]. For reproduction of the results, the common parameter setting is given in Table 13 for Sect. 5.2 and Sect. 5.3. And, the other remaining parameters used in the various experiments are given in separate cases clearly.

(i) Parameter setting for random workflows

Case 1: Experiments on varying number of workflows

In this case, the number of workflows is varied from 16 to 512, and the results are shown in Figs 5, 6, 7 and Tables 14, 15, 16 in Sect. 5.2. The fixed input parameters used in these experiments are as follows:

Number of VMs (n) = 64, Number of depth level (d_{max}) = 8, Number of tasks in each depth level ($N^{l}_{wf_i}$) = 16, Number of tasks (N_{wf_i}) in a workflow = 128.

Case 2: Experiments on varying number of VMs

In this case, the number of VMs is varied from 16 to 512, and the results are shown in Figures 8, 9, 10 and Tables 17, 18, 19 in Sect. 5.2. The fixed input parameters used in these experiments are as follows:

Number of workflows ($N_{\mathcal{W}f}$) = 64, Number of depth level (d_{max}) = 16, Number of tasks in each depth level ($N^{l}_{wf_i}$) = 32, Number of tasks (N_{wf_i}) in a workflow = 512.

(ii) Parameter setting for real application workflows

In this case, the number of real application workflows is varied from 16 to 512, and the results are shown in Figs. 12, 13, 14 and in Sect. 5.3. Due to fixed structure, the number of tasks in Montage, LIGO, and Cybershake in the study is 25, 30, and 40, and the depth levels are 9, 5, and 6 respectively. The fixed input parameters in these experiments are as follows: Number of VMs (n) = 32, Number of depth level (d_{max}) = 9(montage)/5(CyberShake)/6(LIGO).

All the experiments are repeated 20 times to find out the representative values of the objective parameters to handle the randomness during experiments. Tables 14, 15, 16, 17, 18, 19 have been presenting the minimum (Min), maximum (Max), average (Avg), and standard deviation (Std) for all the cases. The superior values are also shown in bold in all the tables.

Table 14 Computed NTF of LBSIR, HEFT, SPM1, SPM2, and SPMWA for varying workflows

Number of task failure							
$N_{\mathcal{W}f}$		L-LMS	L-SMS	HEFT	SPM1	SPM2	SPMWA
16	Min	924	902	827	800	817	**741**
	Max	1574	1584	1659	1651	1651	**848**
	Avg	1284.46	1287.25	1298.61	1278.52	1285.75	**798.75**
	Std	218.21	233.98	292.91	268.94	271.74	**16.09**
32	Min	1678	1644	1562	1472	1536	**1509**
	Max	3197	3234	3336	3316	3326	**1639**
	Avg	2289.72	2282.45	2263.05	2242.45	2253.05	**1597.63**
	Std	485.84	512.48	574.73	547.43	563.44	**30.06**
64	Min	3368	3344	3249	3124	3138	**2219**
	Max	6412	6457	6550	6513	6524	**2349**
	Avg	4585.45	4577.62	4557.35	4482.35	4509.74	**2279.71**
	Std	968.21	995.02	1058.29	991.01	1017.17	**38.62**
128	Min	6651	6606	6492	6389	6463	**5212**
	Max	12,972	13,026	13,154	13,051	13,125	**5438**
	Avg	10,284.35	10,295.52	10,309.44	10,206.41	10,280.42	**5309.65**
	Std	1821.48	1852.18	1920.18	1819.56	1897.76	**59.54**
256	Min	13,115	13,070	12,947	12,729	12,844	**11,722**
	Max	26,066	26,110	26,251	26,042	26,148	**12,145**
	Avg	20,096.62	20,105.53	20,116.01	19,901.32	20,026.13	**11,936.22**
	Std	4878.26	4913.78	5016.06	4725.71	4917.64	**93.9972**
512	Min	34,972	34,423	35,864	35,029	35,453	**21,574**
	Max	50,006	49,510	51,208	50,373	50,797	**21,822**
	Avg	43,997.22	43,483.41	45,081.17	44,246.22	44,670.23	**21,691.42**
	Std	6496.15	6534.49	6646.27	6410.88	6506.02	**112.15**

5.2 Experimental results for random workflows

In this section, the experiments have been conducted discussing the results for randomly generated workflows for the first two cases such that a varying number of workflows and VMs. For this, the E_{ijk} for the VM and workflow tasks has been generated by using the standard ETC simulation benchmark model [59] with high machine and workflow tasks heterogeneity for an inconsistent environment. So, the range of E_{ijk} becomes 1–300,000. As per Eq. (2), CC_k^l is becomes in the range 1–300,000 which is equal to the range of E_{ijk}.

Case 1: Varying Number of Workflows

For the first case, the experimental results presenting a varying number of workflows for L-LMS, L-SMS, HEFT, SPM1, SPM2, and SPMWA are shown in Figs. 5, 6, 7 using boxplot for better representation and graphical self-interpretation and Tables 14, 15, 16. For performance comparison, simulation results of each algorithm viz. L-LMS, L-SMS, HEFT, SPM1, SPM2, and SPMWA have been done.

As expected, the number of task failure (NTF) is increasing as the number of workflows is increased for fixed VMs for all considered models, as shown in Fig. 5(a–f) and Table 14. The performance of the SPMWA model is observed to be superior among all the models for all batch of workflows. The reason behind this is strictly satisfying the security demand of the tasks on allocation if any VM with sufficient trust is available. It reduces the number of tasks failure (NTF) on the allocated VMs. As shown in Fig. 5(a–f) and Table 14, SPMWA clearly exhibits its superior performance in terms of best, average, worst, and standard deviation values of NTF in comparison to LBSIR variants, HEFT, SPM1, and SPM2 for varying number of workflows from 16 to 512. In this case, the average performance gain of SPMWA over LBSIR, HEFT, SPM1, and SPM2 on account of NTF is approximately 43%. As expected, the performance order on NTF is SPMWA, SPM1, SPM2, LBSIR, and HEFT.

As shown in Fig. 6(a–f) and Table 15, SPMWA outperforms to all considered models on account of the failure probability. This is due to the allocation of high-security demand tasks in each partition to higher trustworthy VMs satisfying requirements fully. In this case, the failure probability of tasks becomes zero. Moreover, if the security demand of a task is not satisfied against the trust level of

Table 15 Computed failure probability of LBSIR, HEFT, SPM1, SPM2, and SPMWA for varying workflows

Failure probability							
N_{Wf}		L-LMS	L-SMS	HEFT	SPM1	SPM2	SPMWA
16	Min	0.41780771	0.40980595	0.46193543	0.21205328	0.31205328	**0.06971988**
	Max	0.92090465	0.92964141	0.97639928	0.78672588	0.78886909	**0.40148686**
	Avg	0.68648617	0.69446336	0.77561393	0.62194596	0.66105399	**0.18566491**
	Std	0.41501985	0.40705119	0.47109597	0.47109597	0.64510466	**0.13233911**
32	Min	0.46045686	0.46999685	0.43834144	0.37655183	0.39491497	**0.13698623**
	Max	0.91601856	0.90399986	0.86506832	0.76506832	0.77338433	**0.29265141**
	Avg	0.63717201	0.64492252	0.64188587	0.54313586	0.57743300	**0.17441038**
	Std	0.29102402	0.27838174	0.30338879	0.30338879	0.35941159	**0.11378482**
64	Min	0.40711625	0.40268412	0.40370586	0.18034843	0.23244708	**0.14080143**
	Max	0.99939836	0.98242828	0.99483059	0.89483059	0.99522263	**0.26776503**
	Avg	0.70510046	0.70345435	0.72040759	0.61539719	0.66522918	**0.17238576**
	Std	0.29926094	0.30029151	0.35589094	0.33589093	0.51083522	**0.12855644**
128	Min	0.38386372	0.38904075	0.35648627	0.34648627	0.19911258	**0.14911514**
	Max	0.93804049	0.95868924	0.97562432	0.90998349	0.93333231	**0.21254426**
	Avg	0.66633752	0.66249589	0.70514810	0.62914810	0.63311068	**0.16806132**
	Std	0.37922802	0.37987547	0.45096312	0.39096312	0.42820587	**0.12094725**
256	Min	0.34862577	0.34223097	0.29485761	0.29485760	0.17675254	**0.14655866**
	Max	0.86124714	0.86586449	0.94563898	0.84563898	0.89369889	**0.20926695**
	Avg	0.61311255	0.61084080	0.63812877	0.58797635	0.59581893	**0.17935648**
	Std	0.35899546	0.37922294	0.43186124	0.38186124	0.60543060	**0.11955267**
512	Min	0.38112709	0.36434602	0.44223935	0.39695359	0.14430318	**0.14387614**
	Max	0.97701689	0.97506096	0.94835382	0.84835382	0.89705327	**0.28048212**
	Avg	0.67207622	0.67015136	0.69554231	0.64454231	0.62710382	**0.19013570**
	Std	0.4699139	0.4670278	0.61785037	0.41785037	0.50060319	**0.12983303**

the VM, then it is assigned onto the VMs which offer the least risk of failure. Hence, the Min and Max failure probability of SPMWA is below 7–20% in all cases while other models give a minimum (Min) failure probability of 18% and maximum (Max) failure probability of 99% as shown in Table 15. In Fig. 6(a–f), SPMWA clearly exhibits it is superior performance in terms of best, average, standard deviation, and worst values of failure probability in comparison to LBSIR variants, HEFT, SPM1, SPM2 for all batch sizes from 16 to 512. So, it is evident that SPMWA outperforms among all algorithms on failure probability. In this case, the average performance gain on the failure probability of SPMWA over SPM1, SPM2, LBSIR, and HEFT has been improved by approximately 73%. The performance order on the failure probability is the same as earlier.

In Fig. 7(a–f) and Table 16, the trend of the makespan (Min, Avg, Max) values are increased when the number of workflows increases as expected. Both variants of the LBSIR clearly show superior performance on makespan for all varying batch sizes. However, SPMWA is showing better results in terms of makespan other models excluding the LBSIR variants. The reason for the superior results of LBSIR variants on makespan is that LBSIR variants target makespan to minimize without considering the security demand of the task. The average improvement of LBSIR variants over the SPMWA is approximately 21% in terms of makespan. However, SPMWA has better performance than HEFT, and SPM variants. The performance improvement against them is approximately 89%, 90%, and 97% respectively, with performance order such as LBSIR, SPMWA, HEFT, SPM1, and SPM2.

Case 2: Varying number of VMs

For the second case, the experimental results presenting varying the number of VMs for L-LMS, L-SMS, HEFT, SPM1, SPM2, and SPMWA are shown in Figs 8, 9, 10 and Tables 17, 18, 19 for $n = 16$ to $n = 512$. For the performance comparison, simulation results of each algorithm viz. L-LMS, L-SMS, HEFT, SPM1, SPM2, and SPMWA have been done. The best, worst and average values of the number of task failure, failure probability, and makespan of obtained solutions are reported in Tables 17, 18, 19, respectively.

As presented in Fig. 8(a–f) and Table 17, the number of task failure is in a decreasing trend when the number of VMs is increased on keeping the fixed batch of workflows

Table 16 Computed makespan of LBSIR, HEFT, SPM1, SPM2, and SPMWA for varying workflows

Makespan							
$N_{\mathcal{W}f}$		L-LMS	L-SMS	HEFT	SPM1	SPM2	SPMWA
16	Min	**2,110,829.18**	2,138,696.01	28,667,300.19	29,167,300.19	33,341,319.27	2,254,976.04
	Max	2,298,726.71	**2,283,559.53**	30,739,599.57	31,539,599.57	54,889,142.03	2,426,075.19
	Avg	**2,192,722.47**	2,195,217.02	29,597,752.57	30,122,752.57	47,072,052.68	2,338,265.46
	Std	56,075.01	**37,165.21**	577,629.74	577,629.20	6,111,452.65	50,181.79
32	Min	2,163,456.65	**2,126,016.92**	29,084,308.53	29,784,308.53	54,878,332.13	2,298,038.54
	Max	2,337,844.07	**2,318,937.21**	30,689,093.44	32,403,995.69	79,312,983.17	2,550,408.46
	Avg	2,248,891.15	**2,220,131.21**	30,044,742.04	30,719,742.04	69,255,595.31	2,387,295.98
	Std	56,643.31	**46,756.39**	408,641.04	4,017,416.53	6,801,229.94	52,197.22
64	Min	**2,193,138.47**	2,219,679.56	29,820,023.81	30,375,530.05	92,144,758.67	2,411,878.69
	Max	2,339,394.26	**2,385,564.43**	31,371,995.13	32,471,995.13	127,334,623.90	3,538,371.51
	Avg	**2,258,316.83**	2,273,152.16	30,519,671.32	31,323,671.37	110,109,338.62	2,618,076.28
	Std	48,037.12	**44,710.99**	376,786.31	387,768.40	9,949,552.69	240,623.47
128	Min	2,286,174.14	**2,285,655.97**	30,326,332.42	31,026,332.42	166,326,253.35	2,681,616.65
	Max	2,427,401.00	**2,399,587.74**	31,547,308.31	32,947,308.31	212,303,263.54	8,797,213.34
	Avg	2,355,962.45	**2,332,364.97**	31,052,999.29	31,863,554.17	193,454,869.52	3,379,220.62
	Std	38,403.78	**33,643.58**	393,435.67	401,412.03	13,421,121.18	1,437,976.08
256	Min	2,423,674.01	**2,411,595.22**	30,733,293.49	30,831,538.64	317,175,268.82	3,078,776.53
	Max	2,552,990.92	**2,502,598.09**	32,314,380.89	32,412,626.04	401,123,807.44	4,586,197.46
	Avg	2,485,048.72	**2,449,196.02**	31,540,369.97	31,638,615.12	356,740,890.52	3,701,275.03
	Std	35,497.37	**27,017.53**	464,163.37	471,197.53	21,401,720.74	408,133.48
512	Min	2,423,674.01	**2,411,595.22**	30,733,293.49	30,831,538.64	317,175,268.84	3,078,776.53
	Max	2,797,480.59	**2,726,130.62**	32,886,957.22	32,891,608.43	582,194,393.82	8,313,529.27
	Avg	2,567,562.21	**2,522,091.28**	31,668,274.88	31,737,687.93	419,200,114.41	4,161,610.04
	Std	29,869.31	**19,985.91**	447,346.32	446,695.02	16,565,528.34	1,872,780.05

for all considered models. This is because of the better exploitation of parallelism available in the workflows. In various runs, Min and Max NTF for SPMWA are 9975 and 13,231 in all cases, while for other models these values are 12,129 and 24,608, respectively. As we see in Fig. 8(e–f), when the number of VMs increases from 16 to 512, the average number of task failure for SPMWA is decreasing. It is due to the fact that when the number of VMs is increased, the chances of getting more trustworthy machines increase. For this metric, SPMWA outperforms on all other models in the study. In this case, the average performance gains of SPMWA on NTF over SPM1, SPM2, and LBSIR, HEFT are approximately in the range of 37%-40%. The performance order is SPMWA, SPM1, SPM2, LBSIR, and HEFT.

As can be seen in Fig. 9(a–f) and Table 18, the failure probability is slightly decreasing when the number of VMs is increased. The average failure probability of SPMWA is below 18% in all cases while other algorithms attain the Min failure probability 44%. As shown in Fig. 9(d–f), when we test the SPMWA model on a large number of VMs, then the average failure probability of the proposed model becomes lower e.g., $F_{\not p} = 15 - 12\%$ for a number of VMs from 256 to 512. Because when the number of machines is increased, the chances of more trustworthy machines to assign the task increase. So, it is evident that SPMWA outperforms all algorithms on failure probability. In this case, the average performance gain on failure probability for SPMWA over SPM1, SPM2, and LBSIR, HEFT has been improved by approximately 77%. The performance order is SPMWA, SPM1, SPM2, LBSIR, and HEFT.

When the number of VMs increased, best, worst, and average values of the makespan have the same trend as number of task failure as shown in Fig. 10(a–f) and Table 19. In this case, LBSIR variants clearly show superior performance on makespan among all from 16 to 256 VMs. The reason behind the superior results of LBSIR variants on makespan is the same as mentioned earlier. The SPMWA is showing better results in terms of makespan among all strategies except LBSIR variants. However, on a large number of VMs, the makespan of LBSIR gets closer

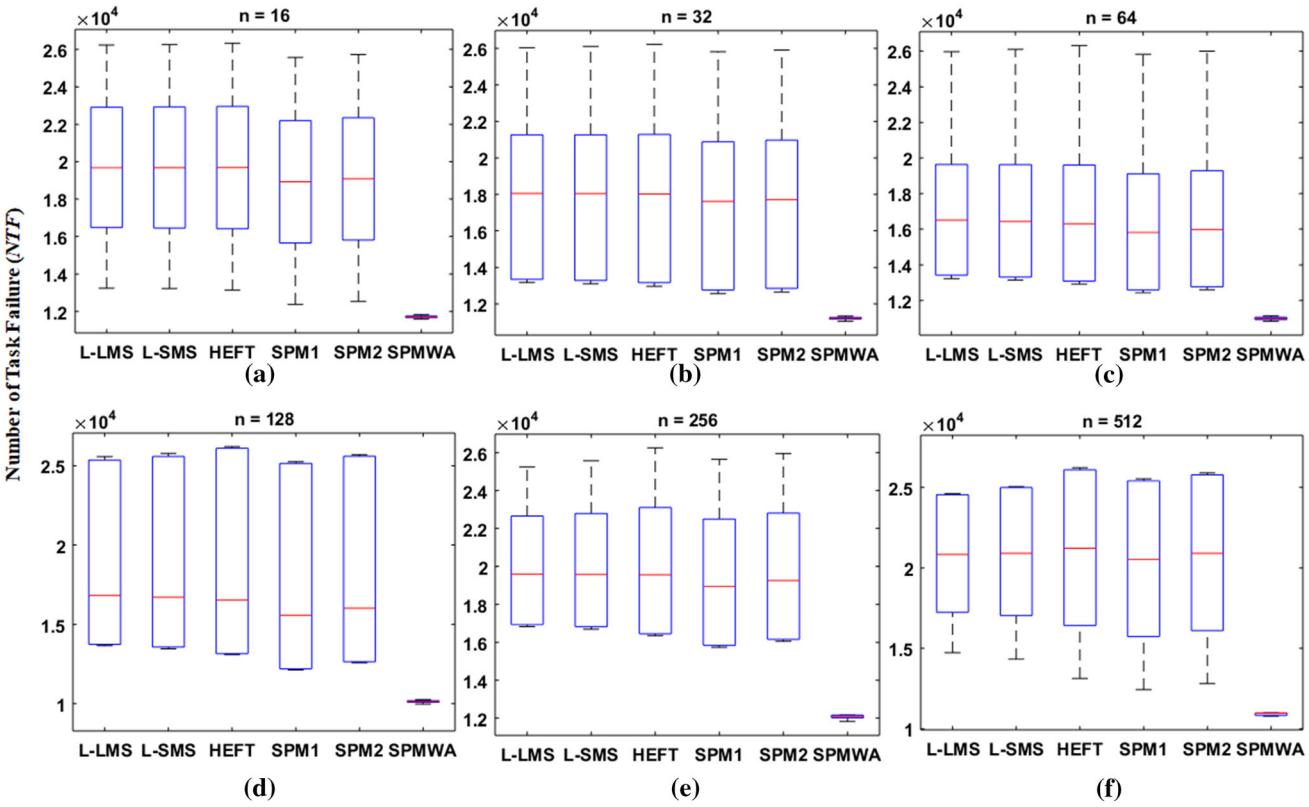

Fig. 8 Boxplots of NTF on varying number of VMs

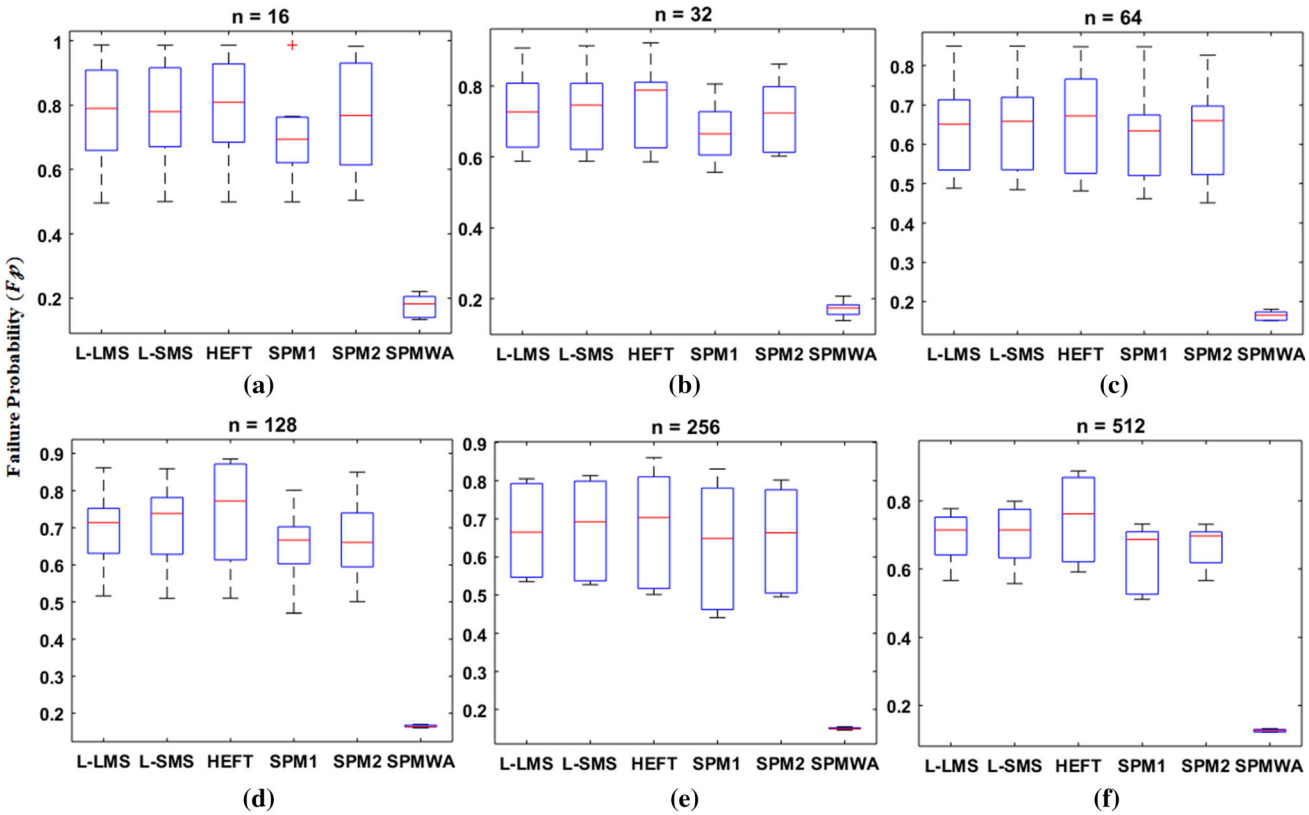

Fig. 9 Boxplots of failure probability on varying number of VMs

Table 17 Computed NTF of LBSIR, HEFT, SPM1, SPM2, and SPMWA for varying VMs

Number of task failure							
n		L-LMS	L-SMS	HEFT	SPM1	SPM2	SPMWA
16	Min	13,241	13,227	13,135	12,377	12,534	**11,587**
	Max	26,228	26,248	26,326	25,568	25,725	**11,824**
	Avg	20,153.35	20,157.7	20,164.51	19,406.49	19,563.63	**11,710.75**
	Std	3946.65	3966.80	4008.57	3901.80	3927.98	**63.33**
32	Min	13,175	13,106	12,968	12,566	12,657	**11,044**
	Max	26,035	26,106	26,220	25,818	25,909	**11,330**
	Avg	18,522.35	18,510.55	18,484.15	18,082.15	18,173.15	**11,204.55**
	Std	4755.81	4800.28	4899.66	4691.15	4759.91	**69.67**
64	Min	13,229	13,143	12,928	**12,435**	12,928	12,928
	Max	25,961	26,094	26,316	25,823	26,316	**13,231**
	Avg	17,155.40	17,108.25	17,014.45	16,521.45	17,014.45	**13,083**
	Std	4246.76	4326.69	4480.69	4319.06	4411.13	**87.88**
128	Min	13,652	13,465	13,089	12,129	12,579	**9975**
	Max	25,579	25,782	26,215	25,255	25,705	**10,258**
	Avg	19,050.80	19,032.10	19,015.60	18,055.46	18,505.52	**10,128.30**
	Std	5153.62	5320.87	5693.37	5789.44	5957.33	**76.027**
256	Min	16,832	16,696	16,350	15,739	16,053	**11,825**
	Max	25,251	25,571	26,253	25,642	25,956	**12,170**
	Avg	20,256.00	20,279.60	20,340.80	19,729.77	20,043.81	**12,060.41**
	Std	3440.38	3622.43	4046.12	3989.96	4009.79	**108.87**
512	Min	14,721	14,317	13,115	12,425	12,800	**10,768**
	Max	24,608	25,046	26,217	25,527	25,902	**10,986**
	Avg	20,457.33	20,527.83	20,707.33	20,017.14	20,392.62	**10,904.67**
	Std	4446.75	4824.98	5887.19	5712.76	5754.78	**97.84**

to the makespan of SPMWA as can be seen in Fig. 10(d–e). SPMWA also tries to fill the idle gaps produced due to the size difference between tasks at the depth level by accommodating the best-fitted task which in turn, ensures a better makespan. Based on the outcomes of Table 19 and Fig. 10a, it is observed that HEFT and SPM1 algorithm performs better than SPMWA on makespan for a very small number of machines (e.g., $n = 16$). However, when the number of VMs is increased, the proposed model clearly shows better performance on makespan against HEFT and SPM1. The performance gain on the makespan of the proposed model over HEFT, SPM1, and SPM2 is approximately 48%, 49%, and 98% but lagging to LBSIR variants by 27%.

5.3 Experimental results for real application workflows

In this section, the experiments have been conducted for real application workflows (Montage_25, Cybershake_30, and LIGO_40) on a varying number of workflows. The parameters taken in this study are as mentioned earlier in Sect. 5.1.

The Montage_25, Cybershake_30, and LIGO_40 having 25, 30, and 40 tasks respectively, are taken for the experiments from the Pegasus website [60] as shown in Fig. 11a, b, and c. The Montage workflow has scope in astronomy. It creates unique sky mosaics from a set of input images in the "Flexible Image Transport System (FITS)" format. The majority of its tasks are simple. I/O intensives are used to describe processing capacity. The CyberShake is employed in synthetic seismograms to determine characteristics. The Laser Interferometer Gravitational Wave Observatory (LIGO) is a spacecraft that detects gravitational waves. It developed a scientific methodology to identify gravitational waves produced by various cosmic events [50, 51]. These real workflows are frequently used to examine the quality of schedules generated by various models proposed in the literature Here, SPMWA, L-LMS, L-SMS, HEFT, SPM1, and SPM2 models are evaluated to see how failure probability affects schedule quality.

Figures 12, 13, 14 present the number of task failure, failure probability, and makespan for Montage, CyberShake, and LIGO when varying the number of real workflows from 16 to 512. SPMWA outperforms to all other models from all batch sizes of real workflows on account of

Table 18 Computed failure probability of LBSIR, HEFT, SPM1, SPM2, and SPMWA for varying VMs

Failure probability							
n		L-LMS	L-SMS	HEFT	SPM1	SPM2	SPMWA
16	Min	0.49540295	0.49950648	0.49872453	0.49872453	0.50345186	**0.13297740**
	Max	0.98680623	0.98579043	0.98581180	0.98581180	0.98232667	**0.22050118**
	Avg	0.77642564	0.77190286	0.79293019	0.70216307	0.75924079	**0.17629113**
	Std	0.48769091	0.49113956	0.50370895	0.49018641	0.50345919	**0.13286830**
32	Min	0.58767142	0.58744126	0.58600623	0.55600620	0.60219977	**0.13815901**
	Max	0.90676816	0.91287493	0.92131572	0.75879882	0.86091510	**0.20723362**
	Avg	0.71058551	0.71592901	0.73081432	0.65195606	0.69947640	**0.17024516**
	Std	0.37314403	0.38085002	0.38940868	0.35940851	0.42239395	**0.12117481**
64	Min	0.48785941	0.48426803	0.48115135	0.46115134	0.45078723	**0.15086870**
	Max	0.84946963	0.84974616	0.84775085	0.84775086	0.82626065	**0.18011457**
	Avg	0.63917655	0.64113203	0.65274838	0.62035838	0.63357437	**0.16377027**
	Std	0.32596674	0.33262440	0.54784002	0.34788459	0.36904375	**0.11115614**
128	Min	0.51594003	0.50953323	0.51002192	0.47002192	0.50065642	**0.16018718**
	Max	0.86117738	0.85865077	0.88502721	0.80156616	0.84966351	**0.16999579**
	Avg	0.68823673	0.69982738	0.73366906	0.64866906	0.66084780	**0.16451630**
	Std	0.36267195	0.38331087	0.42852453	0.38682207	0.40060956	**0.01321524**
256	Min	0.53506523	0.52714807	0.50124283	0.44124283	0.49532881	**0.14548126**
	Max	0.80501808	0.81290424	0.86007077	0.83007077	0.80193763	**0.15432674**
	Avg	0.66857609	0.67448073	0.67757188	0.63169688	0.64839048	**0.15007623**
	Std	0.26384422	0.28230710	0.32522541	0.29541152	0.34954520	**0.01262143**
512	Min	0.56644160	0.55733438	0.59102072	0.51102072	0.56611105	**0.12142489**
	Max	0.77708917	0.79899192	0.88670522	0.73176042	0.73133870	**0.13227547**
	Avg	0.69362313	0.69910254	0.74645350	0.63245350	0.66643355	**0.12648988**
	Std	0.21966101	0.24158836	0.30457895	0.28540093	0.30482349	**0.01102571**

the NTF as presented in Fig. 12. The number of task failures increases on increasing the number of workflows (Montage_25, Cybershake_30, and LIGO_40) for all considered models keeping all the other input parameters fixed. Again, the performance order is SPMWA, SPM1, SPM1, LBSIR, and HEFT. The performance gain of SPMWA has been enhanced over SPM1, SPM2, LBSIR, and HEFT (taking the average of all three sets of workflows), in the range of 30–36% approximately.

As shown in Fig. 13, SPMWA outperforms all other models for every batch size from small to larger real workflows on account of the failure probability. The trend of SPMWA on failure probability keeps slightly increasing on increasing the number of real workflows for Montage_25, Cybershake_30, and LIGO_40 keeping all the other input parameters fixed. The performance order is SPMWA, SPM1, SPM1, LBSIR, and HEFT as expected. The performance gain of SPMWA has been enhanced over SPM1, SPM1, LBSIR, and HEFT, as observed in the average of all three workflows, is in the range of 47–55% approximately.

Makespan is an increasing trend when the batch size is increased for this case as shown in Fig. 14. Both variants of LBSIR clearly show superior in terms of makespan on all scientific workflows from smaller to larger batch sizes. However, on a large number of every real workflow, the makespan of SPMWA gets closer to the makespan of LBSIR as can be seen in Fig. 14. The overall performance gain of LBSIR has been enhanced over SPMWA as observed is 19%, 20%, and 38% approximately on Montage, CyberShake, and LIGO workflows. However, SPMWA is showing better than HEFT, SPM1, and SPM2, in terms of makespan, and the performance gain is 18%, 20%, and 33% on Montage, CyberShake, and LIGO scientific workflows with 16 to 512 workflows sets. The performance order on makespan is LBSIR, SPMWA, HEFT, SPM1, and SPM2. The worst performance is observed by SPM2 because workflows are assigned one after another in sequential exploiting only task-level parallelism in the workflow.

In summary of experimental results of Sects. 5.2 and 5.3, SPMWA performs remarkably best among all considered models in terms of the failure probability, henceforth, number of task failure by achieving the higher optimal values on the objective for both cases. The performance order is SPMWA, SPM1, SPM2, LBSIR, and

Table 19 Computed makespan of LBSIR, HEFT, SPM1, SPM2, and SPMWA for varying VMs

Makespan							
n		L-LMS	L-SMS	HEFT	SPM1	SPM2	SPMWA
16	Min	8,512,959.05	**8,201,739.78**	15,668,448.06	15,769,660.06	613,190,501.16	13,183,460.56
	Max	9,312,779.38	**8,927,320.09**	19,154,708.15	19,259,211.24	783,110,656.21	100,056,710.61
	Avg	8,940,234.59	**8,555,342.02**	17,438,495.77	17,539,800.84	709,258,390.97	33,972,887.07
	Std	221,436.94	**209,862.75**	823,479.05	821,803.23	50,169,565.28	18,062,834.27
32	Min	**5,644,477.92**	5,673,719.76	32,799,855.38	33,841,969.62	502,843,194.24	8,036,199.47
	Max	6,030,910.50	**5,941,773.09**	36,409,929.45	37,452,043.74	757,204,237.60	50,217,421.58
	Avg	5,869,920.80	**5,786,199.27**	34,549,249.66	35,591,365.27	594,399,131.93	13,235,907.51
	Std	92,156.19	**72,777.61**	894,851.99	898,844.07	60,298,052.76	9,133,623.45
64	Min	**5,003,724.19**	5,019,282.28	68,178,764.37	69,184,023.62	445,102,983.80	5,772,076.21
	Max	5,291,563.54	**5,185,603.88**	70,659,538.05	71,664,797.34	672,233,549.25	7,256,913.62
	Avg	5,135,411.32	**5,087,294.12**	69,400,701.13	70,405,960.38	552,985,099.51	6,307,084.66
	Std	65,056.83	**50,890.35**	603,660.65	616,360.58	55,732,933.72	364,982.65
128	Min	4,889,497.04	**4,808,678.27**	40,080,655.99	41,983,778.26	431,487,279.94	5,157,102.31
	Max	5,079,650.43	**5,027,206.43**	50,054,006.75	53,982,731.75	632,393,993.84	5,435,445.64
	Avg	4,963,315.58	**4,914,525.22**	44,055,232.92	46,995,263.32	527,168,905.45	5,263,599.11
	Std	70,365.61	**68,786.77**	429,810.99	434,915.46	8,678,336.63	103,610.21
256	Min	4,814,654.54	**4,797,045.14**	31,957,014.94	32,189,723.48	414,034,242.48	4,922,210.01
	Max	5,039,167.21	**5,032,021.87**	37,507,992.62	37,602,494.26	638,281,462.46	5,203,070.31
	Avg	4,910,896.32	**4,902,282.71**	34,549,249.66	35,591,365.27	492,026,662.32	5,054,511.21
	Std	80,239.50	**66,530.59**	1,129,308.71	113,541.94	5,866,618.01	84,690.28
512	Min	4,813,498.43	**4,791,330.13**	10,696,905.31	10,767,980.59	328,779,014.82	4,999,300.37
	Max	5,010,587.18	**4,893,578.68**	13,818,311.32	13,145,775.69	495,928,907.85	5,237,851.58
	Avg	**4,871,778.62**	4,848,027.02	11,468,842.95	12,128,691.46	423,812,604.62	5,092,729.74
	Std	74,465.97	39,097.12	383,713.79	394,589.28	4,922,408.73	84,086.40

HEFT. It is due to the fact that SPMWA is a security-adaptive workflow allocation method. Therefore, it ensures the allocation of high-security demand tasks in each partition to higher trustworthy VMs satisfying requirements fully. As successful task execution with the least failure probability is an inevitable need in the IaaS cloud. On makespan LBSIR variants are better, however, for a large number of VMs SPMWA becomes closer to them. For real application workflows such as Montage, CyberShake, and LIGO, the proposed SPMWA model is also exhibiting the best performance on failure probability and number of task failure for all considered real workflow sets. The performance trend is the same as in the case of random workflows. Thus, SPMWA is very flexible from a programming point of view, which is also an important concern of designers of workflow allocation with security prioritization applications. Thus, we can argue that the SPMWA may greatly contribute to a secure and robust system transaction with satisfied desired constraints.

6 Conclusion and future work

Nowadays, cloud computing is adopted in many emerging application areas such as e-commerce services, medical services, transaction processing systems, scientific workflow processing, and many more. Workflow allocation satisfying the security requirements in the cloud system is also one of the core issues. In this paper, Security Prioritized Multiple Workflow Allocation (SPMWA) model is proposed that integrates the security constraints into allocation to minimize the failure probability of the workflow execution in the cloud computing environment. SPMWA gives more chance to high-security demanded tasks to get allocated onto the higher trustworthy VMs during allocation. The performance of SPMWA has been compared with standard algorithms, namely HEFT, LBSIR, SPM1, and SPM2. Our experimental results reveal that SPMWA outperforms to LBSIR, HEFT, SPM1, and SPM2, on account of the number of task failure, and failure

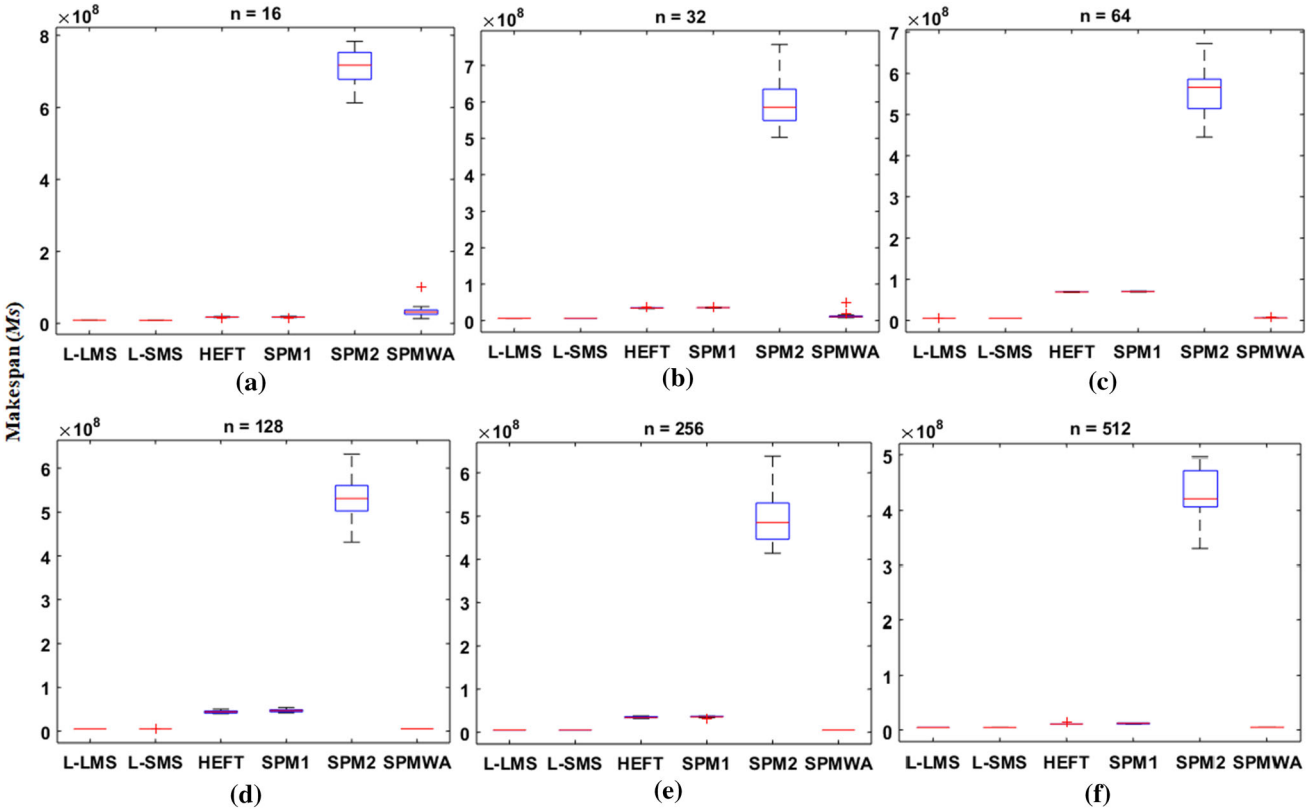

Fig. 10 Boxplots of Makespan on varying number of varying VMs

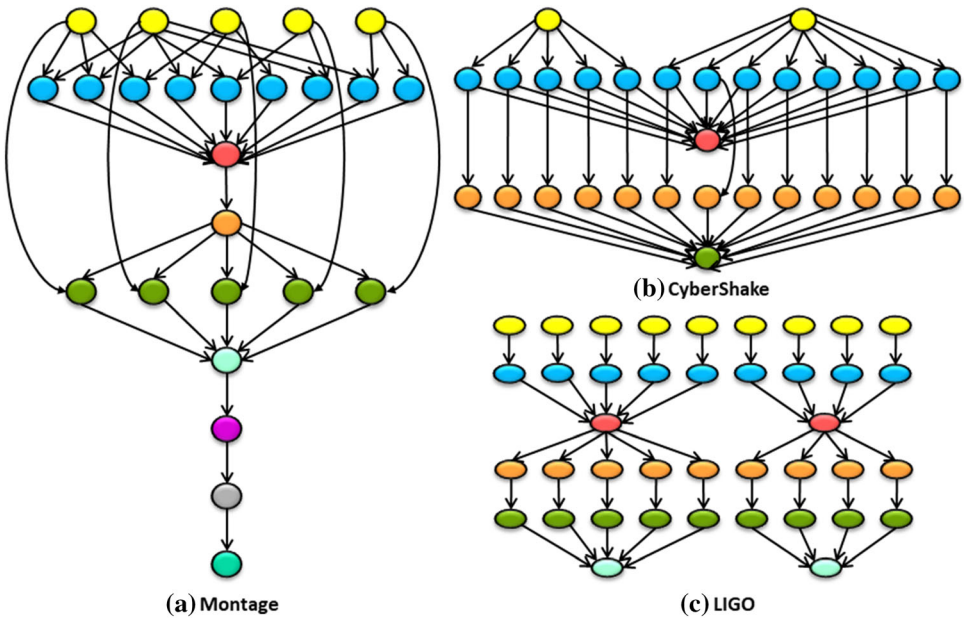

Fig. 11 Real Workflow Applications **a** Montage **b** CyberShake **c** LIGO

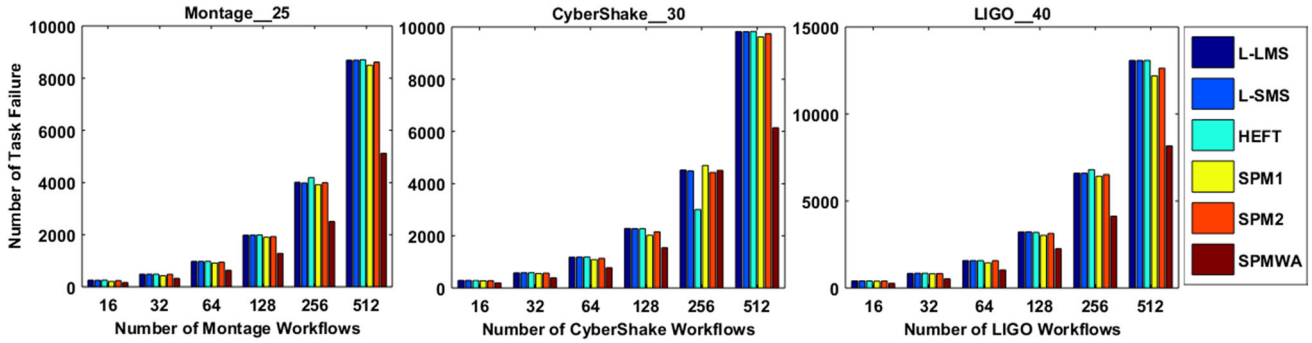

Fig. 12 NTF on varying Montage, CyberShake, and LIGO workflows

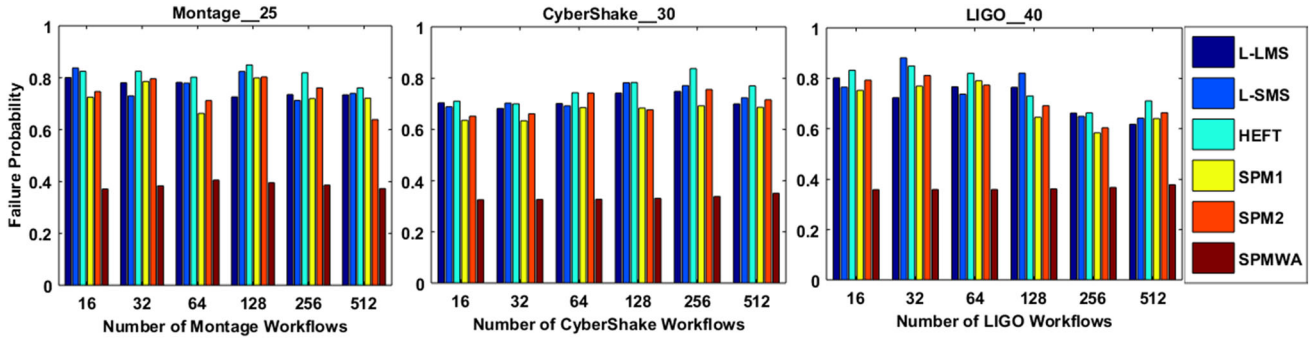

Fig. 13 Failure probability on varying Montage, CyberShake, and LIGO workflows

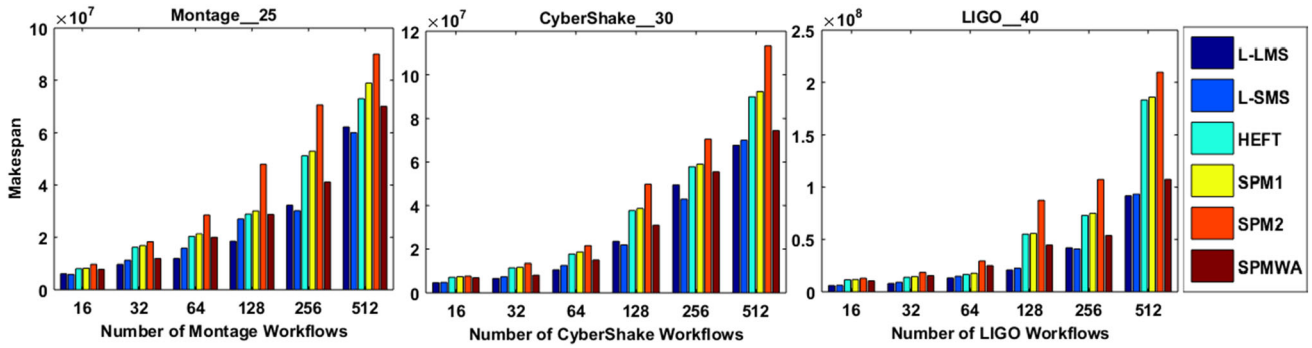

Fig. 14 Makespan on varying Montage, CyberShake, and LIGO workflows

probability. For makespan, SPMWA is better than HEFT, SPM1, and SPM2, and lagging to LBSIR.

There are still some limitations of the proposed SPMWA model for security-sensitive workflow applications which need to be addressed as potential future work as follows:

- To be extended for a more robust and efficient model considering all three security services namely, authentication, integrity, and confidentiality.
- To include some more cloud system's constraints in the model such as deadline, budget, etc.
- To extend the model for a multi-objective model considering QoS parameters like time, energy, and economic cost.

Appendix

The overall security overhead (*SO*) of the system for considered workflows can be estimated as per Eq. (18):

$$\mathrm{SO} = \sum_{i=1}^{N_{wf}} \sum_{j=1}^{N_{wfi}} \sum_{k=1}^{n} z_{ijk} \times \mathrm{SO}_{ijk} \tag{18}$$

Table 20 Computed overall security overhead of LBSIR, HEFT, SPM1, SPM2, and SPMWA for varying workflows

Overall security overhead							
N_{Wf}		L-LMS	L-SMS	HEFT	SPM1	SPM2	SPMWA
16	Min	113.878096	115.030312	104.686522	104.686524	85.419552	**56.853354**
	Max	207.362053	209.309384	198.897617	198.897624	391.849433	**101.814233**
	Avg	144.083288	145.600578	144.301426	144.301434	147.981614	**77.805978**
	Std	23.633497	24.817673	24.812143	23.843851	66.637739	**15.905697**
32	Min	175.927568	179.334120	174.455822	154.455826	172.997254	**91.313074**
	Max	386.101166	383.442475	400.126517	344.305014	484.433017	**224.817317**
	Avg	272.481354	269.967266	274.830309	244.930318	286.062844	**147.783164**
	Std	46.971745	44.477684	50.742505	48.276455	79.369407	**39.739664**
64	Min	400.988740	397.951671	376.858159	326.297975	367.957261	**202.696358**
	Max	753.813885	747.544550	763.135104	701.135145	1161.136016	**452.433005**
	Avg	525.095995	523.057243	526.524768	468.424774	586.207333	**297.043262**
	Std	92.474661	92.419327	102.516305	98.489435	215.278283	**72.997449**
128	Min	882.333177	882.161041	871.115944	763.115944	709.944191	**431.602415**
	Max	1660.043605	1654.793154	1603.299024	1495.299732	1825.580261	**973.329669**
	Avg	1114.698473	1116.106257	1105.603162	997.706955	1064.343871	**637.945337**
	Std	177.673618	175.330058	167.399579	149.883991	255.551838	**138.298118**
256	Min	1423.351823	1456.711929	1421.928264	1295.929345	1639.606494	**653.428546**
	Max	3103.519866	3087.179547	3115.023356	2967.789461	3189.871066	**1932.100962**
	Avg	2342.943414	2342.635992	2339.194670	2219.855512	2427.303731	**1289.295109**
	Std	390.0128203	384.724584	401.799264	371.224583	392.041131	**325.969640**
512	Min	1423.351823	1456.711929	1421.928261	1295.929378	1639.606497	**653.428546**
	Max	4971.211792	5028.407332	4973.146925	4923.146902	5115.683332	**2375.469877**
	Avg	2891.060718	2891.031292	2882.951786	2786.985443	2936.848612	**1527.564146**
	Std	677.731077	686.527198	668.935473	651.336207	788.647480	**221.045410**

Table 21 Computed overall security overhead of LBSIR, HEFT, SPM1, SPM2, and SPMWA for varying VMs

Overall security overhead							
n		L-LMS	L-SMS	HEFT	SPM1	SPM2	SPMWA
16	Min	1149.456355	1153.569466	1151.527183	786.402604	1129.112147	**524.880818**
	Max	4726.162125	4620.540699	4736.561231	4371.436652	4188.790791	**2687.117522**
	Avg	2477.007069	2473.978435	2477.787318	2261.825093	3094.814365	**1300.21333**
	Std	946.704357	938.863488	952.409404	912.185067	951.066741	**689.093760**
32	Min	1450.806625	1449.520709	1449.355474	1078.100685	1306.965493	**604.369247**
	Max	3492.597552	3501.213754	3582.016864	3210.762075	3353.079962	**2055.811222**
	Avg	2408.308394	2425.405193	2432.743771	2153.05264	2535.232593	**1277.178375**
	Std	513.341248	523.281742	532.501493	511.3202451	551.718885	**454.894286**
64	Min	1761.781622	1773.263286	1735.059697	1499.935118	1526.920929	**867.9732894**
	Max	3029.834651	3028.31135	3083.038141	2847.991823	4082.185677	**2067.451576**
	Avg	2390.990002	2386.778492	2388.158882	2112.834036	2441.035448	**1248.518066**
	Std	402.614627	403.266269	404.337281	395.898240	619.744245	**299.955819**
128	Min	1928.920292	1926.961601	1875.702622	1564.702622	1745.421641	**1014.121044**
	Max	2771.750385	2766.911167	2724.994167	2413.994167	3029.252951	**1718.020903**
	Avg	2388.901911	2383.044714	2387.521093	2058.995513	2428.776197	**1238.186599**
	Std	297.110144	300.120272	306.586975	351.0750379	480.082266	**214.501951**
256	Min	2105.542868	2101.700467	2164.921713	1795.297135	1940.76719	**1019.141374**
	Max	2649.761968	2638.119367	2641.694363	2272.069785	3336.220046	**1602.213114**

Table 21 (continued)

Overall security overhead							
n		L-LMS	L-SMS	HEFT	SPM1	SPM2	SPMWA
	Avg	2350.136092	2355.683101	2369.995513	2017.896515	2414.826886	**1196.865305**
	Std	151.461386	153.375373	147.806118	**141.7967387**	433.853777	229.640587
512	Min	2257.397769	2159.162044	2281.009815	1870.124786	1563.533646	**1153.76281**
	Max	2537.340705	2563.924182	2595.887716	2356.974516	4593.38743	**1574.455287**
	Avg	2268.238102	2269.974091	2289.553048	1918.386363	2302.581454	**1181.530862**
	Std	92.177048	**88.790449**	103.976430	99.649307	1055.784145	133.408417

Table 22 The overall security overhead on varying Montage, CyberShake, and LIGO workflows

Overall security overhead						
$N_{\mathcal{W}f}$	L-LMS	L-SMS	HEFT	SPM1	SPM2	SPMWA
Montage_25						
16	19.611503	12.186401	12.725582	9.954082	10.545810	**6.734785**
32	44.600799	30.397318	34.623158	28.623158	30.575113	**21.164171**
64	76.109843	64.342475	76.655145	67.805413	82.699239	**46.149068**
128	157.002155	144.353633	179.775017	161.174377	185.670904	**112.237208**
256	354.127825	324.457812	514.987451	478.225402	438.457813	**302.145784**
512	589.149880	608.517516	1047.505071	1000.002145	838.957523	**433.544829**
CyberShake_30						
16	25.701806	15.075343	12.002836	11.000165	17.284475	**10.801113**
32	47.760595	34.624505	37.475029	30.123529	45.127304	**28.620902**
64	89.931613	78.253742	119.888245	101.200188	112.336365	**64.487630**
128	160.870883	147.278908	267.182974	207.718294	199.222863	**133.797719**
256	372.457812	364.445201	514.987451	478.225402	448.554455	**349.458126**
512	596.9100443	624.1387225	1106.78399	978.122545	809.2619064	**581.028619**
LIGO_40						
16	38.0597726	20.536363	23.930908	20.571208	30.074275	**15.097862**
32	64.1380220	46.102428	54.204967	47.670492	68.205209	**36.387757**
64	116.779883	99.190207	134.955155	121.457814	138.826944	**84.487089**
128	271.251774	279.093340	366.077922	304.724571	325.134196	**220.424729**
256	526.781246	539.457813	701.584971	613.897451	609.145781	**427.568912**
512	825.949989	834.938621	1511.862342	1345.124578	1171.025079	**681.463123**

where z_{ijk} is the assignment vector, SO_{ijk} is the security overhead of T_{ij} on V_k and it is computed as per Eq. (4).

As shown in Table 20, the best, worst, and average values of the security overhead are increased when the number of workflows increases as expected. However, SPMWA outperforms among all cases in terms of security overhead. It is obvious that security overhead is directly proportional to failure probability. Therefore, it follows the same performance order as expected among all models with the same reason explained above in the case of failure probability. Here, also the majority of the tasks are executed without risk or at least possible risk resulting in less security overhead. The performance gain of SPMWA over LBSIR, HEFT, SPM1, and SPM2 on account of security overhead is 45%, 45%, 41%, and 46% respectively. The performance order for the security overhead is SPMWA, SPM1, LBSIR, HEFT, and SPM2 for all batch sizes of the workflows which is almost the same for failure probability. (see Tables 20, 21, 22)

Again, the best, worst and average values of the security overhead are decreasing when the number of the VMs is increased as expected presented in Table 21. Thus, SPMWA outperforms all cases in terms of security overhead. The average performance gain on security overhead has been improved in the range of 40% to 50% over SPM1,

SPM2, LBSIR, and HEFT. The performance order is SPMWA, SPM1, LBSIR, HEFT, and SPM2 for all sets of virtual machines.

The trend of average security overhead is increasing for all batch sizes of Montage, CyberShake, and LIGO keeping other parameters fixed as shown in Table 22. SPMWA performs best among all other models on each real workflow for every batch size in terms of security overhead. Even when the security demand for a task is not fulfilled, the proposed model assigns that task where its failure risk is least. Further, the Also, the performance of SPMWA has been improved on all real workflows for every batch size over SPM1, SPM2, LBSIR, and HEFT as observed in the range of 32–43%, 19–36%, and 25–40% approximately on Montage, CyberShake, and LIGO respectively. The performance order on security overhead is SPMWA, SPM2, SPM1, LBSIR, and HEFT.

Acknowledgements This work is supported by the grant from University Grant Commission-Basic Scientific Research (UGC-BSR) (Research Startup Grant with sanction No. F.30-377/2017 (BSR), Code-10057/B30528).

Author contributions MS contributed to the study conception and design of the model. MA performs design, implementation, and data acquisition of the model. The data analysis, manuscript preparation, and editing are completed by all authors. All authors have read and agreed to be published a version of the manuscript.

Funding This work is supported by Research Startup Grant with sanction No. [F.30-377/2017] (BSR) and Code: [10057/B30528]. Dr. Mohammad Shahid has received this research supported by University Grant Commission (UGC), India.

Data availability Data is generated during this study for the subset of cipher suites supported by TLS V1.0 to V1.2 protocols. These cipher suites TLS V1.0 to V1.2 protocols are taken as follows: http://tools.ietf.org/html/rfc5430; https://pegasus.isi.edu/workflow_gallery/; https://ciphersuite.info/.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

1. Mell, P., & Grance, T. (2011). *The NIST definition of cloud computing*. National Institute of Standards and Technology Special Publication, 800–145.
2. Hwang, K., Dongarra, J., Fox, G.C.: Distributed and Cloud Computing: From Parallel Processing to the Internet of Things. Morgan kaufmann, Burlington (2013)
3. Wang, Y., Lu, P.: Dataflow detection and applications to workflow scheduling. Concurr. Comput.: Pract. Exp. **23**(11), 1261–1283 (2011)
4. Wang, Y., & Shi, W. (2013). On scheduling algorithms for mapreduce jobs in heterogeneous clouds with budget constraints. In *International Conference on Principles of Distributed Systems* (pp. 251–265). Springer, Cham.
5. Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., Vahi, K.: Characterizing and profiling scientific workflows. Futur. Gener. Comput. Syst. **29**(3), 682–692 (2013)
6. Arunarani, A.R., Manjula, D., Sugumaran, V.: FFBAT: a security and cost-aware workflow scheduling approach combining firefly and bat algorithms. Concurr. Comput.: Pract. Exp. **29**(24), e4295 (2017)
7. Alam, M., Mahak, Haidri, R.A., Yadav, D.K.: Efficient task scheduling on virtual machine in cloud computing environment. Int. J. Pervasive Comput. Commun. **17**(3), 271–287 (2021)
8. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-Science: an overview of workflow system features and capabilities. Futur. Gener. Comput. Syst. **25**(5), 528–540 (2009)
9. Sih, G.C., Lee, E.A.: A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures. IEEE Trans. Parallel Distrib. Syst. **4**(2), 175–187 (1993)
10. Iverson, M. A., Özgüner, F., & Follen, G. J. (1995). Parallelizing existing applications in a distributed heterogeneous environment. In *4th Heterogeneous Computing Workshop (HCW'95)*.
11. Topcuoglu, H., Hariri, S., Wu, M.Y.: Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. **13**(3), 260–274 (2002)
12. Dubey, K., Kumar, M., Sharma, S.C.: Modified HEFT algorithm for task scheduling in cloud environment. Procedia Comput. Sci. **125**, 725–732 (2018)
13. Arabnejad, H., Barbosa, J.G.: List scheduling algorithm for heterogeneous systems by an optimistic cost table. IEEE Trans. Parallel Distrib. Syst. **25**(3), 682–694 (2013)
14. Huang, T., Li, T., Dong, Q., Zhao, K., Ma, W., Yang, Y.: Communication-aware task scheduling algorithm for heterogeneous computing. Int. J. High Perform. Comput. Netw. **10**(4–5), 298–309 (2017)
15. Li, T., Cao, D., Lu, Y., Huang, T., Sun, C., Dong, Q., Gong, X.: DBEFT: a dependency-ratio bundling earliest finish time algorithm for heterogeneous computing. IEEE Access **7**, 173884–173896 (2019)
16. Haidri, R. A., Katti, C. P., & Saxena, P. C. (2017). Cost effective deadline aware scheduling strategy for workflow applications on virtual machines in cloud computing. *Journal of King Saud University-Computer and Information Sciences*.
17. Belgacem, A., Beghdad-Bey, K.: Multi-objective workflow scheduling in cloud computing: trade-off between makespan and cost. Clust. Comput. **25**(1), 579–595 (2022)
18. Hussain, M., Wei, L.F., Rehman, A., Abbas, F., Hussain, A., Ali, M.: Deadline-constrained energy-aware workflow scheduling in geographically distributed cloud data centers. Futur. Gener. Comput. Syst. **132**, 211–222 (2022)
19. Konjaang, J.K., Murphy, J., Murphy, L.: Energy-efficient virtual-machine mapping algorithm (EViMA) for workflow tasks with deadlines in a cloud environment. J Netw. Comput. Appl. **203**, 13400 (2022)
20. Thaman, J., Kumar, K.: Performance aware planning algorithms for cloud environments. Int. J. Distrib. Syst. Technol. (IJDST) **9**(1), 1–15 (2018)
21. Mahmud, R., Srirama, S.N., Ramamohanarao, K., Buyya, R.: Quality of experience (QoE)-aware placement of applications in fog computing environments. J. Parallel Distrib. Comput. **132**, 190–203 (2019)
22. Xie, T., Qin, X.: Security-aware resource allocation for real-time parallel jobs on homogeneous and heterogeneous clusters. IEEE Trans. Parallel Distrib. Syst. **19**(5), 682–697 (2008)
23. Wang, M., Ramamohanarao, K., Chen, J.: Trust-based robust scheduling and runtime adaptation of scientific workflow. Concurr. Comput.: Pract. Exp. **21**(16), 1982–1998 (2009)

24. Xiaoyong, T., Li, K., Zeng, Z., Veeravalli, B.: A novel security-driven scheduling algorithm for precedence-constrained tasks in heterogeneous distributed systems. IEEE Trans. Comput. **60**(7), 1017–1029 (2010)
25. Wang, W., Zeng, G., Tang, D., Yao, J.: Cloud-DLS: dynamic trusted scheduling for cloud computing. Expert Syst. Appl. **39**(3), 2321–2329 (2012)
26. Tan, W., Sun, Y., Lu, G., Tang, A., & Cui, L. (2012, November). Trust services-oriented multi-objects workflow scheduling model for cloud computing. In: *Joint international conference on pervasive computing and the networked world* (pp. 617–630). Springer, Berlin, Heidelberg.
27. Rathanam, G.J., Rajaram, A.: Trust based meta-heuristics workflow scheduling in cloud service environment. Circuits Syst. **7**(04), 520 (2016)
28. Chen, H., Zhu, X., Qiu, D., Liu, L., Du, Z.: Scheduling for workflows with security-sensitive intermediate data by selective tasks duplication in clouds. IEEE Trans. Parallel Distrib. Syst. **28**(9), 2674–2688 (2017)
29. Sujana, J.A.J., Geethanjali, M., Raj, R.V., Revathi, T.: Trust model-based scheduling of stochastic workflows in cloud and fog computing. In: Cloud Computing for Geospatial Big Data Analytics, pp. 29–54. Springer, Cham (2019)
30. Shahid, M., Alam, M., Hasan, F., & Imran, M. (2021). Security-aware workflow allocation strategy for IaaS cloud environment. In *Proceedings of International Conference on Communication and Computational Technologies* (pp. 241–252). Springer, Singapore,
31. Alam, M., Shahid, M., & Mustajab, S. (2021). SAHEFT: security aware heterogeneous earliest finish time workflow allocation strategy for IaaS cloud environment. In: *2021 IEEE Madras Section Conference (MASCON)* (pp. 1–8). IEEE.
32. Alam, M., Shahid, M., & Mustajab, S. (2022). Security prioritized heterogeneous earliest finish time workflow allocation algorithm for cloud computing. *Lecture Notes on Data Engineering and Communications Technologies, Vol. 114, pp. xx-xx, 2021, chapter No. 17 in 2nd Congress on Intelligent Systems* (*CIS-2021*). Springer, Singapore.
33. Peng, K., Zhu, M., Zhang, Y., Liu, L., Zhang, J., Leung, V., Zheng, L.: An energy-and cost-aware computation offloading method for workflow applications in mobile edge computing. EURASIP J. Wirel. Commun. Netw. **2019**(1), 1–15 (2019)
34. Bisht, J., Vampugani, V.S.: Load and cost-aware min-min workflow scheduling algorithm for heterogeneous resources in fog, cloud, and edge scenarios. Int. J. Cloud Appl. Comput. (IJCAC) **12**(1), 1–20 (2022)
35. Bittencourt, L.F., Madeira, E.R.: Towards the scheduling of multiple workflows on computational grids. J. Grid Comput. **8**(3), 419–441 (2010)
36. Saovapakhiran, B., Michailidis, G., & Devetsikiotis, M. (2011). Aggregated-DAG scheduling for job flow maximization in heterogeneous cloud computing. In: *2011 IEEE Global Telecommunications Conference-GLOBECOM 2011* (pp. 1–6). IEEE.
37. Hirales-Carbajal, A., Tchernykh, A., Yahyapour, R., González-García, J.L., Röblitz, T., Ramírez-Alcaraz, J.M.: Multiple workflow scheduling strategies with user run time estimates on a grid. J. Grid Comput. **10**(2), 325–346 (2012)
38. Shahid, M., Raza, Z.: Level-based batch scheduling strategies for computational grid. Int. J. Grid Util. Comput. **5**(2), 135–148 (2014)
39. Shahid, M., Raza, Z., Sajid, M.: Level based batch scheduling strategy with idle slot reduction under DAG constraints for computational grid. J. Syst. Softw. **108**, 110–133 (2015)
40. Chen, W., Lee, Y.C., Fekete, A., Zomaya, A.Y.: Adaptive multiple-workflow scheduling with task rearrangement. J. Supercomput. **71**(4), 1297–1317 (2015)
41. Bochenina, K., Butakov, N., Boukhanovsky, A.: Static scheduling of multiple workflows with soft deadlines in non-dedicated heterogeneous environments. Futur. Gener. Comput. Syst. **55**, 51–61 (2016)
42. Zhang, H., Zheng, X., Xia, Y., Li, M.: Workflow scheduling in the cloud with weighted upward-rank priority scheme using random walk and uniform spare budget splitting. IEEE Access **7**, 60359–60375 (2019)
43. Ma, X., Xu, H., Gao, H., Bian, M.: Real-time multiple-workflow scheduling in cloud environments. IEEE Trans. Netw. Serv. Manag. **18**(4), 4002–4018 (2021)
44. Arabnejad, V., Bubendorfer, K., Ng, B.: Dynamic multi-workflow scheduling: a deadline and cost-aware approach for commercial clouds. Futur. Gener. Comput. Syst. **100**, 98–108 (2019)
45. Jiang, J., Lin, Y., Xie, G., Fu, L., Yang, J.: Time and energy optimization algorithms for the static scheduling of multiple workflows in heterogeneous computing system. J. Grid Comput. **15**(4), 435–456 (2017)
46. Taghinezhad-Niar, A., Pashazadeh, S., Taheri, J.: QoS-aware online scheduling of multiple workflows under task execution time uncertainty in clouds. Clust. Comput. (2022). https://doi.org/10.1007/s10586-022-03600-8
47. Anisetti, M., Ardagna, C. A., Bonatti, P. A., Damiani, E., Faella, M., Galdi, C., & Sauro, L. (2014) e-Auctions for multi-cloud service provisioning. In *2014 IEEE International Conference on Services Computing* (pp. 35–42). IEEE.
48. Goertzel, K. M., Winograd, T., McKinley, H. L., Oh, L. J., Colon, M., McGibbon, T., ... & Vienneau, R. (2007). Software Security Assurance State-of-the-Art Report (SOAR). *Information Assurance Technology Analysis Center*, Herndon, VA, 1–392.
49. Ardagna, C.A., Asal, R., Damiani, E., Vu, Q.H.: From security to assurance in the cloud: a survey. ACM Comput. Surv. (CSUR) **48**(1), 1–50 (2015)
50. Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P.J., Mayani, R., Chen, W., Ferreira da Silva, R., LivnyM, W.K.: Pegasus: a workflow management system for science automation. FuturGenerComputSyst **46**, 17–35 (2015)
51. Brown, D.A., Brady, P.R., Dietz, A., Cao, J., Johnson, B., McNabb, J.: A case study on the use of workflow technologies for scientific analysis: gravitational wave data analysis. In: Workflows for e-Science, pp. 39–59. Springer, London (2007)
52. Hartmanis, J.: Computers and intractability: a guide to the theory of np-completeness (Michael R. Garey and David S. Johnson). Siam Rev. **24**(1), 90 (1982)
53. Omer, S., Azizi, S., Shojafar, M., Tafazolli, R.: A priority, power and traffic-aware virtual machine placement of IoT applications in cloud data centers. J. Syst. Architect. **115**, 101996 (2021)
54. Hoseiny, F., Azizi, S., Shojafar, M., Tafazolli, R.: Joint QoS-aware and cost-efficient task scheduling for fog-cloud resources in a volunteer computing system. ACM Trans. Internet Technol. (TOIT) **21**(4), 1–21 (2021)
55. Xie, T., Qin, X.: Scheduling security-critical real-time applications on clusters. IEEE Trans. Comput. **55**(7), 864–879 (2006)
56. Kashyap, R., Vidyarthi, D.P.: Security-aware scheduling model for computational grid. Concurr. Comput.: Pract. Exp. **24**(12), 1377–1391 (2012)
57. Wang, B., Wang, C., Huang, W., Song, Y., Qin, X.: Security-aware task scheduling with deadline constraints on heterogeneous hybrid clouds. J. Parallel Distrib. Comput. **153**, 15–28 (2021)
58. Li, Z., Ge, J., Li, C., Yang, H., Hu, H., Luo, B., Chang, V.: Energy cost minimization with job security guarantee in Internet data center. Futur. Gener. Comput. Syst. **73**, 63–78 (2017)

59. Braun, T.D., Siegel, H.J., Beck, N., Bölöni, L.L., Maheswaran, M., Reuther, A.I., et al.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distrib. Comput. **61**(6), 810–837 (2001)
60. https://pegasus.isi.edu/workflow_gallery/

Mahfooz Alam is a Research Scholar in Department of Computer Science, Aligarh Muslim University (AMU), Aligarh, India. M. Alam has also served for six years as an Assistant Professor in Department of Computer Science at Al-Barkaat College of Graduate Studies (ABCGS), Aligarh, India. He received his M.Tech. in Computer Science and Engineering from Dr. A. P. J. Abdul Kalam Technical University, Lucknow, India. He also completed his M.C.A. and B.C.A. in Computer Applications from Indira Gandhi National Open University (IGNOU) New Delhi, India. He has published many research papers in international conferences and international journals of repute (including IEEE Access & Xplore, Springer, Elsevier, Wiley, IGI Global, etc.). His research interest areas are Cloud Computing, Workflow Scheduling, Load Balancing, Parallel and Distributed Computing.

Mohammad Shahid is an Assistant Professor in the Department of Commerce, Aligarh Muslim University, Aligarh, India. He has received his Ph.D. from School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India. He earned his M.Tech in Computer Science and Technology from the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi and MCA from Department of Computer Science, Aligarh Muslim University, Aligarh, India. His research interest area is Grid/Cloud Computing, Workflow Scheduling, Evolutionary Computing, Meta-heuristic Optimization, and Portfolio Optimization. He has published many research papers in international conferences and international journals of repute (including Elsevier, Springer, InderScience, Wiley, Taylor & Francis, IGI Global, etc.). He was awarded a research startup grant from UGC-BSR (2017), India. You can contact him at: Department of Commerce, AMU, Aligarh 202002, India.

Suhel Mustajab is working as an Associate Professor, Department of Computer Science, Aligarh Muslim University (AMU), Aligarh. He joined Department of Computer Science in the year 1990; he did his Graduation in Electrical engineering from Z.H.C.E.T, AMU, Aligarh in 1986, Masters in Computer Science from AMU, Aligarh in 1990 and Ph.D. from AMU. Aligarh. He has 32 years of Teaching experience at National and International level and has guided around 160 dissertations at PG level and around 100 at UG level. His area of Interests are Soft computing, Image processing, Cloud Computing, and Internet of Things (IoT). On the administrative side, he has been working as Coordinator, National Education Policy, Aligarh Muslim University, Nodal officer, National Academic Depository, AMU. He has worked as Chairman, Department of Computer Science from 2013 to 2014, Director, Computer Centre, AMU, Aligarh from 2012 to 2013. He has worked as HOD/Program Coordinator, Department of Information Science & Technology, MCMAS (UMR-Rolla), Muscat for Four Years from July, 2004 till July 2008. He was actively involved in the development of the Salary and Accounting system of AMU, during his tenure as Systems Manager, CU-CAO AMU from 2000 to 2003 (on Deputation).