Journal of King Saud University – Computer and Information Sciences 35 (2023) 791–809

HOSTED BY

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Multi objective trust aware task scheduling algorithm in cloud computing using whale optimization

Sudheer Mangalampalli [a,*], Ganesh Reddy Karri [a], Utku Kose [b]

[a] School of Computer Science and Engineering, VIT-AP University, Amaravati, India
[b] Dept. of Computer Engineering, Suleyman Demirel University, Turkey

ARTICLE INFO

Article history:
Received 29 November 2022
Revised 20 January 2023
Accepted 20 January 2023
Available online 25 January 2023

Keywords:
Task Scheduling
Makespan
Energy Consumption
Trust

ABSTRACT

Task Scheduling is an enormous challenge in cloud computing model as to map diverse tasks arises from various sources there should be an efficient scheduling mechanism which provision resources dynamically to users based on their corresponding requests. Ineffective scheduling leads to increase in makespan, energy consumption and violates SLA made between cloud user and service provider thereby quality of service will be degraded and trust on the cloud service provider will be degraded. Trust typically based on quality-of-service parameters such as Availability of virtual resources, Success rate of tasks, Turnaround efficiency of tasks which are included in SLA. In this paper, we designed a Multi objective trust aware scheduler which takes priority of tasks, VMs and schedule tasks to appropriate virtual resources while minimizing makespan, energy consumption. Whale Optimization algorithm used to model our task scheduler. Entire simulation carried out on Cloudsim. Workload used in this simulation is of both fabricated and real-time worklogs captured from HPC2N and NASA. Our proposed approach compared against existing metaheuristic approaches i.e., ACO, GA, PSO approaches. From Simulation results, we observed that there is a significant improvement in makespan, Energy consumption, total running time and trust parameters i.e., Availability, Success rate, Turnaround efficiency.

1. Introduction

Cloud Computing paradigm opened the doors for many enterprises to migrate, compute and host their applications in cloud environment which gives seamless access to many of services with ease and hassle-free nature. More over all these services are tailor made and customized according to the needs of customer. Initially with the advent of big data, existing commodity hardware was cannot cope up with workloads which are heterogeneous and flowing onto the infrastructure and coming from diversified resources. Therefore, to accommodate these heterogeneous and diversified workloads, many IT companies want to migrate their existing infrastructure onto cloud environment. This is because cloud computing model have various advantages such as scalability, elasticity, on demand services, high resilient virtual architectures, flexibility (Francis, 2018). Therefore with these advantages companies want to migrate their existing infrastructure to cloud environment by minimizing their administrative responsibilities and trying to give the customers great flexibility, high availability by using virtualized architectures in datacenters. In these days, every user in various domains are more likely to use cloud computing applications because of flexibility, availability and moreover that for cloud paradigm users are distributed around the globe therefore, to accommodate these concurrent user requests in cloud environment we need an effective scheduler to provision virtual resources to cloud users (Arunarani et al., 2019). When users are requesting services of cloud environment concurrently and these requests made by customers from various heterogeneous and diversified resources therefore to handle these type of requests and to provision resources to various cloud users an effective and dynamic task scheduler is needed. Task scheduler in cloud paradigm need to be effective and dynamic. Moreover that it should work based on the workload coming onto the cloud console. Ineffective scheduling mechanism in cloud paradigm leads to degradation of quality of service of cloud service provider and thereby lost

* Corresponding author.
E-mail addresses: sudheerkietmtech@gmail.com (S. Mangalampalli), utkukose@sdu.edu.tr (U. Kose).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2023.01.016
1319-1578/

of trust for cloud service provider and business will be degraded. Therefore, in cloud paradigm an effective scheduler need to be employed which schedules tasks based on the workloads coming onto cloud console dynamically and it can be beneficial to both the cloud user and cloud service provider. Therefore, by employing an effective scheduler in cloud paradigm is beneficial to both the cloud provider and cloud user through which both can take the benefit. In this paper, we discussed about Trust aware scheduling mechanism and trust can be computed based on SLA parameters such as Availability, success rate, turnaround efficiency. These parameters also effects quality of service and it also indirectly effects makespan, energy consumption. Many of earlier authors formulated task schedulers using various metaheuristic and nature inspired algorithms i.e. PSO (Alsaidy et al., 2020), GA (Velliangiri, 2021), ACO (Sharma and Garg, 2022) and many more approaches. Many authors also addressed parameters named makespan, SLA violation, Energy consumption but no authors addressed trust parameters evaluated in terms of availability, success rate, turnaround efficiency related with makespan and energy consumption by considering priorities of tasks and VMs. In this manuscript, we carefully evaluated priorities for both tasks and VMs and accurately mapping tasks to VMs using whale optimization algorithm (Jia et al., 2021) while minimizing makespan, energy consumption and improving availability, success rate of tasks and turnaround efficiency thereby improving trust value. In this research, we observed that there is a hidden relationship between trust parameters i.e. when trust parameters are improved then makespan and energy consumption minimized in cloud paradigm.

Contributions in this manuscript are represented below.

1. Developed a Multi Objective trust aware scheduling technique using whale optimization (Jia et al., 2021) algorithm.
2. To effectively map tasks onto virtual resources by considering priorities of tasks and VMs based on unit electricity cost.
3. Trust is evaluated based on SLA parameters i.e., Availability, success rate of tasks and turnaround efficiency of tasks. A deadline constraint is posed in our work to carefully execute tasks on VMs in such a way that allocation of tasks to a VMs assigned after pending tasks completion.

In this section, initially we have discussed about the need of cloud computing for various enterprises and discussed the advantages of cloud paradigm over on premises environment. After discussion of basic knowledge about cloud paradigm, we discussed importance of task scheduling in cloud paradigm and how it effects basic parameters such as makespan and explained about importance of SLA based trust parameters and then we explained about various methodologies by existing authors and in next subsection we discussed about motivation for carrying out this research and our contributions towards this research.

Rest of the manuscript is organized as below. Section 1 represents Introduction, Section 2 represents Related Works, Section 3 represents Multi objective Trust aware Task Scheduling in Cloud computing, Section 4 represents Simulation and Results, Section 5 represents Conclusion and Future work.

2. Related works

In (Dubey and Sharma, 2021); authors formulated a task scheduling algorithm to schedule multiple independent tasks to virtual resources. It developed combining CR and PSO approaches. It simulated on Cloudsim tool. It compared against existing CR, PSO baseline approaches and from results, it proved that CR-PSO greatly minimizes makespan, total cost, energy consumption over baseline approaches. Heuristic random initialization made authors to use metaheuristic algorithms for task scheduling in cloud computing. In (Alsaidy et al., 2020) a modified heuristic initialized PSO used in combination with LJFP; MCT algorithms which works based on longest jobs to be executed on fastest processor i.e. LJFP and MCT works based on minimum time to complete job. Experimentation conducted on MATLAB and it evaluated against existing MCT, SJFP,LJFP, PSO, Min-Min, Max-Min algorithms. From results it evident that proposed approach minimizes makespan, total execution time, energy consumption. In (Beegom and Rajasree, 2019); a task scheduling algorithm formulated for minimization of makespan, cost. A discrete modelled Integer PSO used as methodology for task scheduling in cloud computing. This simulation carried out on Cloudsim and synthetic datasets are used to give input to algorithm. It evaluated over RND-PSO, SPV-PSO algorithms. From results, it proved that Integer PSO shown huge impact over baseline approaches for specified parameters. In (Nabi, 2022); authors designed a task scheduling algorithm aims to minimize makespan and improves resource utilization, throughput. They used inertia weight strategy to balance workload and balance between local and global search process. LDAIW strategy with PSO used as methodology. It compared against existing inertia weight mechanisms and simulations conducted on Cloudsim. From experiments, results revealed that proposed inertia weight mechanism integrated with PSO outperforms mentioned parameters. In (Agarwal and Srivastava, 2019); authors formulated task scheduling mechanism aims at minimization of makespan and improves resource utilization. PSO used as methodology to solve task scheduling and it simulated on Cloudsim simulator. It compared over existing approaches i.e. GA, ACO. From results, it evident that proposed PSO based scheduling mechanism outperformed over existing approaches. An adaptive inertia weight based scheduling model developed by authors in (Zhou, 2018) which avoids local optimum in solution space. Modified PSO used as methodology for scheduling problem. According to the workload velocity PSO swarm adjust accordingly by adapting inertia weight. It simulated on Cloudsim and varied number of tasks and evaluated over different baseline PSO variants. From results; it shown huge impact over existing approaches for specified parameter. In (Panwar, 2019); an order of preference task scheduling mechanism proposed to maximize resource utilization, minimize processing cost, makespan. This scheduling model developed in two phases. In first phase, top order preference of tasks are identified and in second phase with adaptive inertia weights PSO algorithm used to schedule tasks. Entire simulation conducted on Cloudsim. It evaluated over baseline approaches PSO, DPSO, ABC,IABC,FUGE. From results, it evident that TOPSIS-PSO shown huge impact over existing approaches for mentioned parameters. In (Ebadifard and Babamir, 2018); a task scheduling algorithm developed which aims to address parameters makespan, resource utilization. PSO based approach used to tackle scheduling problem which adds a load balance technique to PSO and addressed problem of task scheduling. Cloudsim simulator used for experimentation purpose. It evaluated over RR, IPSO, load balancing approach, IRASA algorithms. From results, it shown that it outperforms existing algorithms for above mentioned specified parameters. In (Pang, 2019); a hybrid scheduling approach formulated by combining two algorithms EDA, GA. This approach has two phases. In first phase, EDA will generate solutions randomly based on workload and in second phase, using GA operators task scheduler generate schedules for corresponding tasks. Simulations conducted on Cloudsim with random generated workload. It compared against EDA, GA approaches. From results, it observed that EDA-GA outperforms existing approaches by minimizing task completion time and maximizing load balance of tasks. In (Abualigah and Alkhrabsheh, 2022); authors focused on minimizing task transfer time as they developed a reschedule strategy to increase effectiveness of scheduling tasks. They used a hybrid approach

i.e. MVO-GA which schedule and reschedule tasks based on speed, capacity of tasks, size, number of tasks, number of VMs, throughput. Entire experimentation carried out on MATLAB 2017. It compared over existing MVO, GA approaches. From results it proved that, task transfer time, estimated completion time are minimized over existing algorithms. A hybrid task scheduling approach proposed by authors in (Agarwal and Srivastava, 2018) by combining PSO; GA algorithms in which PSO used diversification property and GA uses intensification property. This approach implemented on Cloudsim. It compared against existing baseline algorithms PSO, GA. From results, it shown that 22.2 % of improvement in makespan over existing approaches. In (Senthil Kumar and Venkatesan, 2019); a scheduling model formulated using a hybrid approach GA-ACO. In this mechanism, GA initializes population and ACO uses updated pheromone which enhances quality of service of scheduler. It implemented on Cloudsim simulator. It evaluated over baseline approaches i.e. ACO, GA algorithms. From results, proposed approach outperforms existing algorithms for throughput, task completion time, response time. In (Elaziz et al., 2021); a hybrid approach for task scheduling proposed by combining WOA, COBL algorithms. In this approach WOA used as local search algorithm, COBL used as global search for scheduling tasks. Simulations carried out on Cloudsim. It evaluated against existing WOA, COBL and results proved that HGSWC outperforms baseline algorithms by minimizing makespan. In (Prasanna Kumar and Kousalya, 2020) a nature inspired based task scheduling algorithm developed based on crow search. It implemented on Cloudsim. Random generated tasks given as input to algorithm in which low processing tasks are scheduled both to low and high capacity VMs and high processing tasks are scheduled both to low and high capacity VMs. It compared over MIN-MIN; ACO algorithms. Results proved that CSA based scheduler shown huge impact in minimization of makespan. In (Duan, 2018); a task scheduling algorithm developed which aims at minimization of makespan, completion time. It modeled with incremental GA approach which adapts to the environment based on workload by adjusting its parameters i.e. crossover and mutation. Experimentation conducted on Cloudsim. From results, it proved that makespan, computation time greatly minimized over baseline GA approaches for specified parameter. In (Senthil Kumar and Venkatesan, 2019); a hybridized task scheduling mechanism developed based on GA, PSO algorithms. This algorithm gives priority to tasks if tasks are repeated in cloud environment and they are stored in queue manager. Remaining tasks are stored in On demand queue and both of these are given as input to HGPSO to schedule tasks. It implemented on Cloudsim. It evaluated against GA, PSO, GA-PSO-HEFT algorithms. Simulation results revealed that HGPSO outperforms compared approaches by minimizing completion time, scalability, availability. In (Abdullah et al., 2019); a task scheduling algorithm formulated to address total task time. It modeled by using MOPSO with importance strategy to identify best global solution in workload. It implemented on Cloudsim. It evaluated over baseline PSO, GA algorithms. From results, it proved that MOPSO-IS outperformed over compared approaches for specified parameter. Authors in (Zhang, 2018) formulated a task scheduling model which addresses task execution time; power consumption. RC-GA used as methodology in which resource crowd model used initially to select solutions when huge requests are propagated to cloud interface from various users and GA improved selection operator used to identify best solutions to map tasks to VMs. Total simulations conducted on Cloudsim. It evaluated over NSGA-II, Random approaches and simulation results revealed that RC-GA outperforms compared baseline approaches for specified parameters. In (Aggarwal, 2020); authors developed a scheduling strategy to identify tradeoff between user requirements and resource utilization. They used SA-FFOA as methodology to address the problem. Extensive experiments are conducted on Cloudsim with synthetic and real-time workloads. It compared against FFOA, DE, ABC approaches and observed that results shown that it is dominant over existing approaches for makespan, execution cost and time. In (Pirozmand, 2021); authors designed an energy conscious task scheduling approach formulated based on GA algorithm. This approach used random workload and this was given to GAECS as input and used for scheduling. It implemented on MATLAB 2014a. It evaluated against GSA,ABC, DA, Linear-PSO, Sigmod-PSO, Chaotic-PSO, logarithm-PSO, simulated-PSO. From results it evident that GAECS shown significant impact over existing baseline approaches for specified metrics. In (Srichandan et al., 2018); efficient task scheduling approach formulated as per the perception of authors to give quality of service to cloud users mentioned in SLA. Therefore, a multi-objective hybrid approach by combining GA and BFA algorithms was proposed. This algorithm tackles scheduling approach in two phases by minimizing makespan in first and energy consumption in second phase. It implemented on MATLAB 2013a and used different random generated workloads to test proposed heuristic and from results it evident that MHBFA outperformed existing approaches for specified parameters. In (Sharma and Jain, 2019); authors proposed a scheduling mechanism using enhanced ACO approach. This algorithm modeled based on enhancing ACO by splitting jobs into various bunches to create sub lists to optimally allocate tasks to VMs. It implemented on Cloudsim toolkit and generated schedules with random workload. It compared over basic ACO approach and observed results. From results, it was observed that EACO greatly minimizes makespan, execution cost far better when compared with ACO. In (Ajmal, 2021); authors developed a scheduling approach which divides all tasks inflowing into cloud console into different groups. The developed approach modeled based on a hybrid model by combining GA, ACO algorithms and it autodetects overloading of tasks at various VMs and assign them to new VMs appropriately. It implemented on Cloudsim and it compared against baseline approaches i.e. GA, ACO and from results, it observed that hybrid approach greatly minimizes 64 % in execution time and 11 % in overall datacenter cost. In (Amer, 2022); a hybridized approach used to develop a scheduling algorithm by enhancing opposition based learning and using standard HHO approach. Opposition based learning is enhanced to increase exploration phase and HHO used for scheduling. It implemented on Cloudsim, random and real time workloads are used in simulation. It evaluated over baseline approaches HHO, OBL, PSO and results revealed that ELHHO shows huge impact over existing algorithms for parameters i.e. cost, throughput, resource utilization, degree of imbalance. In (Chen, 2020); authors used an improvised version of whale optimization to explore its search capability. The main aim of this approach is to minimize cost of scheduling while improving utilization of virtual resources. Experimentation conducted on MATLAB 2018b with randomized workloads. It compared over existing ACO, Whale optimization approaches and results shown that it improvised in terms of total cost of scheduling process over existing approaches. In (Mangalampalli et al., 2022); authors developed a task scheduling algorithm which addresses parameters i.e. makespan, total power cost in datacenters, energy consumption, migration time which impacts cloud paradigm from cloud provider and cloud consumer perspectives. CSO used as methodology to tackle task scheduling problem. All simulations conducted on Cloudsim and workloads used for this approach are randomized and real time worklogs. From simulation results, it proved that proposed approach by authors shown significant improvement over existing algorithms ACO, CS. In (Mangalampalli et al., 2022); authors aims to minimize energy consumption due to high emissions of carbon dioxide in datacenters. While minimizing energy consumption power cost also can be minimized using appropriate scheduling

technique. Whale optimization algorithm used as methodology to tackle scheduling problem. All experimentations conducted on Cloudsim and it compared against baseline approaches i.e. PSO, CS. Results revealed that proposed whale scheduler outperforms existing approaches for makespan, total power cost at datacenters, energy consumption.

From Table 1, it observed that many authors proposed various scheduling algorithms using metaheuristic and nature inspired algorithms but still scheduling in cloud computing can be considered as a NP-Hard problem due to its dynamic nature of heterogeneous and diversified requests and in order to provision virtual resources and while optimizing metrics is still a challenge. Specifically, in this research we identified that authors haven't discussed about trust on cloud provider and their parameters in terms of availability, success rate of tasks and turnaround efficiency which are important for the cloud provider perspective. Therefore, to maintain QoS and thereby to get trust from users above mentioned parameters are necessary. Therefore, we have taken these parameters for consideration of our work and we carefully evaluated priorities of both tasks, then evaluated priorities of VMs based on electricity cost at their datacenters. To model this algorithm we used whale optimization (Jia et al., 2021) to tackle with Scheduling problem in cloud computing.

3. Multi objective trust aware task scheduling in cloud computing

3.1. Problem definition and system architecture

In this section, we carefully defined the problem and system architecture in a detailed manner. For this to happen, we assume that by considering set of *i* tasks, set of *j* virtual resources, *k* physical hosts, *l* datacenters. After assumption, we have defined the problem in such a way that *i* tasks are mapped to *j* virtual resources which are placed in *k* physical hosts in turn they are placed in *l* datacenters while minimizing makespan, energy consumption, Availability, Success rate of tasks, turnaround efficiency by considering priorities of tasks and priorities of VMs by using electricity unit cost at datacenters.

The above Fig. 1 represents proposed system architecture used in this research work. Initially all cloud users submits their requests on to cloud admin console. A broker is a software lies between cloud users and Cloud provider and on behalf of users broker submits requests made by the cloud users to task manager on behalf of all users. In this architecture, after submitting requests to task manager, there is a mechanism that evaluates priorities of all incoming tasks based on their task capacity and appropriate processing capacity of a VM. After that we carefully evaluates priorities of VMs based on their electricity cost at their corresponding location. After evaluating priorities of both tasks and VMs these are to be submitted to MOTSWAO scheduler modeled using whale optimization in which it maps tasks to VMs precisely based on their priorities. When scheduler is mapping tasks to appropriate VMs in Datacenters initially it sends it requests task execution queue and as a deadline constraint is imposed in our scheduler VMs are allocated with tasks after pending tasks completion and it will be automatically tracked by a resource manager where it can track all the resource consumption and it gives status to scheduler based on that scheduler will map tasks accordingly to VMs in Datacenters and in this work, we added an event logger and SLA monitor through which we are calculating trust metrics i.e. Avail-

Table 1
Parameters addressed by various authors in existing scheduling algorithms.

Authors	Technique Used	Simulation environment	Parameters
(Dubey and Sharma, 2021)	CR-PSO	Cloudsim	Makespan, Total cost, energy consumption
(Alsaidy et al., 2020)	LJFP-PSO, MCT-PSO	MATLAB	Makespan, total execution time, energy consumption.
(Beegom and Rajasree, 2019)	Integer PSO	Cloudsim	Cost, makespan
(Nabi, 2022)	Adaptive PSO	Cloudsim	Makespan, Average resource utilization ratio, throughput.
(Agarwal and Srivastava, 2019)	PSO	Cloudsim	Makespan, resource utilization.
(Zhou, 2018)	MPSO	Cloudsim	Total cost
(Panwar, 2019)	TOPSIS-PSO	Cloudsim	Processing Cost, resource utilization, makespan
(Ebadifard and Babamir, 2018)	Load balancing PSO	Cloudsim	Makespan, resource utilization
(Pang, 2019)	EDA-GA	Cloudsim	Task completion time, Load balance of tasks
(Abualigah and Alkhrabsheh, 2022)	MVO-GA	MATLAB 2017	Task transfer time, Estimated completion time.
(Agarwal and Srivastava, 2018)	PSOGA	Cloudsim	makespan
(Senthil Kumar and Venkatesan, 2019)	GA-ACO	Cloudsim	Task Completion time, Response time, throughput.
(Elaziz et al., 2021)	HGSWC	MATLAB	makespan
(Prasanna Kumar and Kousalya, 2020)	CSA	Cloudsim	makespan
(Duan, 2018)	Incremental GA	Cloudsim	Makespan, computation time
(Senthil Kumar and Venkatesan, 2019)	HGPSO	Cloudsim	Availability, Scalability, Completion time
(Abdullah et al., 2019)	MOPSO-IS	Cloudsim	Total task time
(Zhang, 2018)	RC-GA	Cloudsim	Task execution, Power consumption
(Aggarwal, 2020)	SA-FFOA	Cloudsim	Makespan, Cost, execution time
(Pirozmand, 2021)	GAECS	MATLAB 2014a	Makespan, energy consumption.
(Srichandan et al., 2018)	MHBFA	MATLAB 2013a	Makespan, energy consumption.
(Sharma and Jain, 2019)	EACO	Cloudsim	Makespan, execution cost.
(Ajmal, 2021)	HAGA	Cloudsim	Execution time, Overall datacenter cost
(Amer, 2022)	ELHHO	Cloudsim	Cost, throughput, Resource utilization, Degree of Balance
(Chen, 2020)	IWC	MATLAB2018b	Total Cost
(Mangalampalli et al., 2022)	CSO	Cloudsim	Makespan, Total power cost in datacenters, energy consumption, migration time.
(Mangalampalli et al., 2022)	WOA	Cloudsim	Makespan, Total power cost, energy consumption
(Ali, 2021)	Multilevel trust model	MATLAB	Makespan, energy consumption, packet delivery ratio, trust value
(Karthika and Muthukumaran, 2022)	ADS-PAYG	MATLAB	Response time, accuracy, CPU Utilization
(Soleymani, 2021 (2021).)	Fuzzy based trust model	Cloudsim	Service satisfaction, SLA Violation

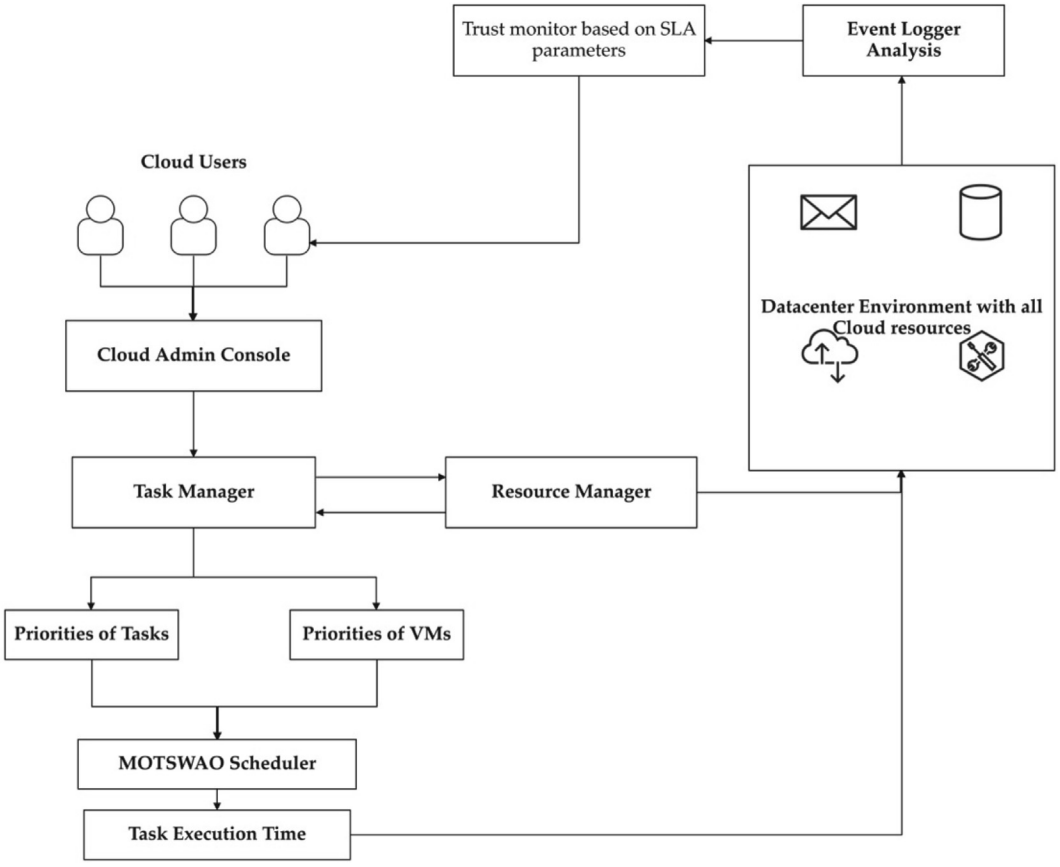

Fig. 1. Proposed System architecture.

ability, success rate, turnaround efficiency and thereby minimizing makespan, energy consumption in cloud paradigm. After describing proposed system architecture, we identified notations to be used in proposed system architecture. Notations used in mathematical modeling are shown as below in Table 2.

3.2. MOTSWAO mathematical modeling

Initially in mathematical modeling, we evaluated current load on all virtual resources considered in our work. It is evaluated using below Eq. (1).

Table 2
Notations used in mathematical modeling for System architecture.

Notation	Meaning
t_i	Number of considered tasks
vm_j	Number of considered VMs
H_k	Number of considered Hosts
DC_l	Number of considered Datacenters
lo_{vm_j}	Load on virtual machines
lo_{H_k}	Load on hosts resided in datacenters
pr_{vm_j}	Processing capacity of VMs
t_i^{prio}	Priorities of tasks
vm_j^{prio}	Priorities of VMs
m^i	Makespan
ene^{con}	Energy consumption
dl_i^t	Deadlines for considered tasks
ex_i^t	Execution time for considered tasks
fin_i^t	Finish time for considered tasks
$av(vm_j)$	Availability of virtual resources
$sr(vm_j)$	Success rate of virtual resources
$te(vm_j)$	Turnaround efficiency of a virtual resource
$trust_{csp}$	Trust of a cloud service provider

Table 3
Configuration Settings used for simulation.

Name	Quantity
No. of Tasks	100–1000
Task Length	800,000
Memory of Physical Host	16 GB
Storage capacity of Physical Host	1 TB
Bandwidth Capacity	100 Mbps
Number of Virtual Machines	30
Memory capacity of virtual machine	1 GB
Bandwidth capacity of Virtual network	10 Mbps
No. of Processing elements	1050 MIPS
Name of Hypervisor	Xen
Type of Hypervisor	Monolithic
Operating System	Linux
No. of Datacenters	5

$$lo_{vm_j} = \sum lo_j \tag{1}$$

Where lo_j indicates load on all considered j virtual resources. After calculating load on all VMs, we need to identify load on hosts and these VMs are placed in hosts. Therefore, load on hosts are identified as below using Eq. (2).

$$lo_{H_k} = \frac{lo_{vm_j}}{\Sigma H_k} \tag{2}$$

Where lo_{H_k} identified as load on overall hosts (see Fig. 2).

Mapping of all incoming tasks to appropriate virtual resources depends on priorities of tasks and VMs in our research. Therefore to calculate priorities of tasks depends on size of tasks, processing capacities of VMs. It can be identified as below using Eq. (3).

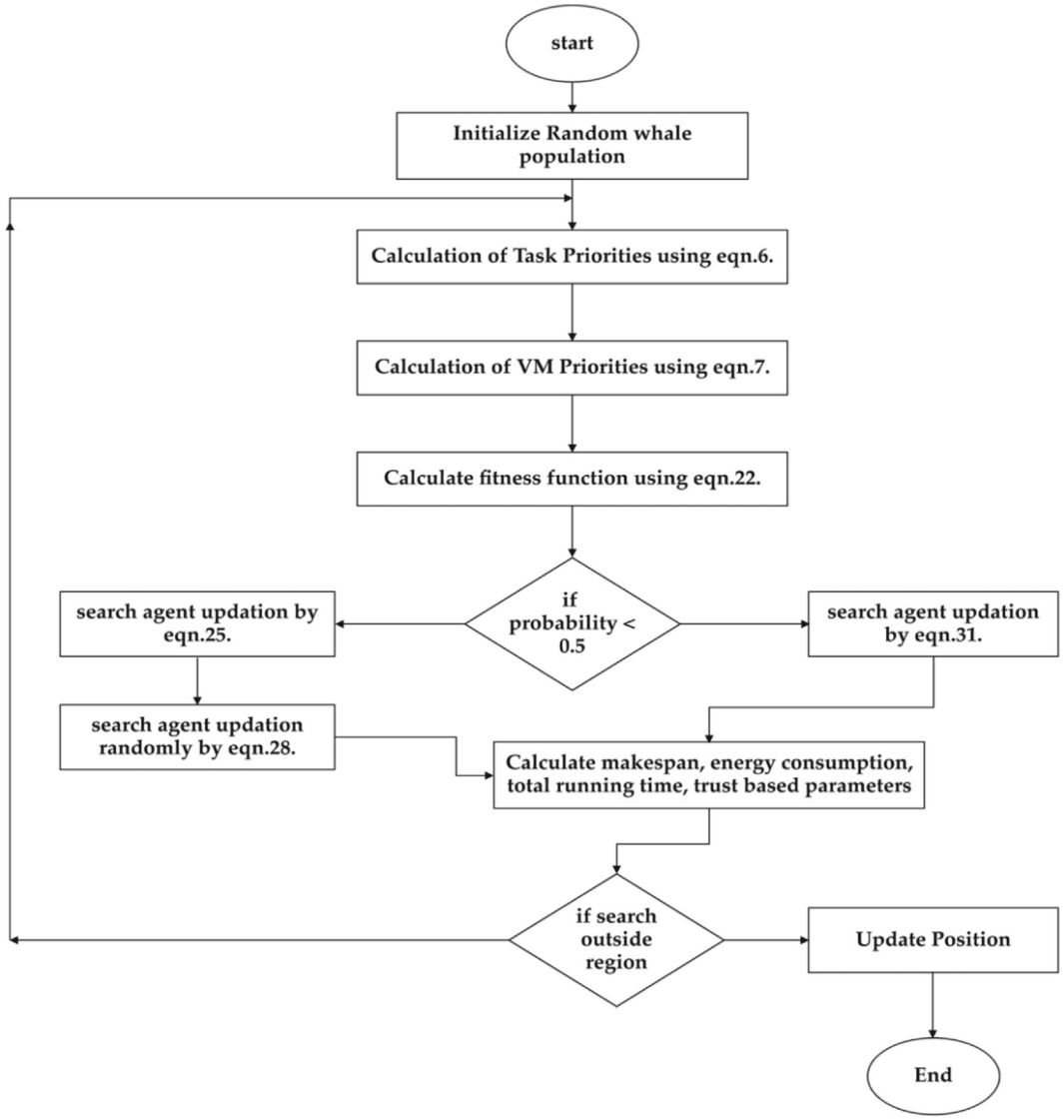

Fig. 2. Flow of proposed MOTSWAO algorithm.

$$pr_{vm} = pr^{no} * pr^{mips} \tag{3}$$

From Eq. (3). we identified process to calculate to processing capacity of a VM. Therefore, to calculate overall processing capacity of all VMs can be identified by using below Eq. (4).

$$tot_{vm}^{pro} = \Sigma pr_{vmj} \tag{4}$$

We identified total processing capacity of VMs by using Eq. (4) and now to calculate priorities of incoming tasks it is important to know that size of task which are varies in cloud computing environment. It is identified using below Eq. (5).

$$t_i^{size} = t_i^{MIPS} * t_i^{p} \tag{5}$$

Now after calculation of size of a task, priority of tasks are identified using below Eq. (6).

$$t_i^{prio} = \frac{t_i^{size}}{pr_{vm}} \tag{6}$$

Where pr_{vm} indicates processing capacity of a VM, t_i^{size} indicates size of tasks.

In this research, we calculated priorities of VMs are calculated based on electricity unit cost. It identified as ratio of highest unit electricity cost in all datacenters to unit electricity cost at that particular datacenter. It identified as below using Eq. (7).

$$vm_j^{prio} = \frac{unitcost_{elec}^{high}}{unitcost_{elec}^{DC}} \tag{7}$$

From Eqs. (6) and (7), we calculate priorities of tasks and VMs and then fed to MOTSWAO scheduler and based collected priorities it maps highest prioritized tasks should maps to highest prioritized VMs which should incur less electricity unit cost charges. While mapping tasks to appropriate VMs based on our scheduler we are identifying trust parameters based on SLA parameters i.e. Availability of VMs, Success rate, turnaround efficiency and thereby minimizing makespan, energy consumption.

Trust of a cloud service provider depends on quality of service given to the customers based on SLA made between customer and provider. Trust will be improved when SLA parameters i.e. Availability, Success rate and turn around efficiency is improved. Therefore, we calculated Availability of a VM using Eq. (8).

$$av(vm_j) = \frac{av_i}{t_i} \tag{8}$$

Table 4
Calculation of makespan using various workloads.

Algorithm No. of tasks	ACO	GA	PSO	MOTSWAO
S01	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	786.7 1.92 774.2	742.8 1.24 732.5	698.2 1.26 687.2	587.3 1.35 572.9
500	954.5 2.01 935.6	1356.9 2.25 1328.7	1135.8 1.09 1123.4	792.9 0.34 783.7
1000	1456.8 1.78 1443.2	1783.2 1.87 1743.8	1856.9 2.87 1843.9	987.23 0.78 956.2
S02	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	965.24 1.23 958.23	945.87 2.87 932.45	896.77 1.87 867.58	783.45 1.34 745.7
500	1309.5 1.98 1298.5	1367.56 3.78 1308.45	1476.21 0.54 1421.56	1049.23 1.88 987.32
1000	1783.6 0.87 1776.7	1798.78 2.76 1754.23	1829.56 1.43 1792.43	1338.5 2.12 1299.8
S03	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	872.4 2.24 867.5	784.67 1.56 732.78	856.52 1.77 834.21	672.56 1.09 632.78
500	943.56 1.67 932.78	1321.78 2.23 1286.5	957.67 0.98 932.78	745.8 0.89 734.34
1000	1432.5 2.67 1411.8	1567.89 1.89 1498.76	1378.6 1.67 1324.5	1012.8 0.76 985.67
S04	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	697.78 0.21 685.24	734.78 1.88 712.56	685.43 5.32 643.23	587.32 2.34 534.22
500	789.45 0.89 775.34	893.67 1.21 832.14	799.23 4.12 743.12	832.45 1.99 724.58
1000	1423.53 2.56 1401.87	1567.21 0.99 1521.8	1343.22 4.25 1312.56	1243.8 2.89 1198.5
S05	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	1576.2 4.22 1499.8	1523.78 1.45 1512.6	1867.45 2.85 1812.6	1043.2 1.78 988.56
500	1884.4 3.12 1856.2	2498.4 1.24 2410.6	2145.43 3.45 2056.8	1563.45 2.56 1521.67
1000	2984.5 2.07 2867.8	3278.6 0.98 3208.8	2876.21 3.96 2756.9	1956.45 1.98 1923.21
S06	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	865.13 2.34 832.11	734.67 1.56 713.78	756.12 2.31 742.12	623.45 2.87 609.12
500	948.56 1.87 912.67	1156.12 2.87 1112.89	912.21 1.56 878.21	823.67 1.92 792.45
1000	1346.9 1.98 1311.23	2024.34 1.67 1994.23	1987.24 2.12 1945.1	1156.7 2.12 1112.2

Where av_i is number of tasks accepted by a VM out of t_i tasks. Therefore from Eq. (8). Availability of a virtual resource defined as ratio of number of tasks accepted by a virtual resource to the number of tasks. It means that VM should be operational and available.

Another parameter we chosen which effects trust is success rate of a virtual resource depends on successful requests over submitted requests over a period of time and it calculated using Eq. (9).

$$sr(vm_j) = \frac{s_i}{av_i} \tag{9}$$

From Eq. (9). s_i indicates successful executed requests on virtual resource and av_i indicates number of submitted requests over a period of time.

After calculating success rate from Eq. (9). turnaround efficiency of a virtual resource identified using Eq. (10). It is defined as ratio of estimated turnaround time provided by cloud service provider in SLA to the actual turnaround time by executing a request on a virtual resource.

$$te(vm_j) = \frac{estim_T}{actual_T} \tag{10}$$

From Eqs. (8)–(10). We calculated trust as follows in Eq. (11).

$$trust_{CSP} = Y1 * av + Y2 * sr * Y3 * te \tag{11}$$

Where $Y = \{Y1, Y2, Y3\}$ are weights for above trust parameters in Eqs. (8)–(10). These weights are positive weights and they are calculated based on co variance technique mentioned in (Singh and Chatterjee, 2017). The weights are lies between 0 and 1. i.e. $Y \in [0, 1]$. These weights may change from user to user and time to time i.e. may be for a user it may change for a user from time to time. Therefore, in above Eq. (11) for availability weight is given as $Y1 = 0.5$, for success rate $Y2 = 0.2$, for turnaround efficiency $Y3 = 0.1$. The above Eq. (11). calculates $trust_{CSP}$ using above weights.

After evaluating trust parameters, we need to evaluate our remaining parameters i.e. makespan, energy consumption. To evaluate makespan, execution time of a task need to be identified and moreover that a deadline constraint is imposed in our work i.e. dl_i^t deadline of tasks. Here execution time of task calculated using below Eq. (12).

$$ex_i^t = \frac{ex^t}{pr_{vm}} \tag{12}$$

We have mentioned earlier that we imposed a deadline constraint in our work i.e. when a user submitted a request or task it should immediately assigned with a virtual resource or otherwise it should wait until it assigned with a VM for finish time of current execution of a task. For that, we calculated finish time of a task using below Eq. (13).

$$fin_i^t = \Sigma vm_j + ex_i^t \tag{13}$$

We have already assumed that every task should finish its execution before finish time completes so as to maintain SLA thereby trust should be improved as per our assumptions. Therefore, finish time of a task should always be less than or equal to deadline of a task. It is calculated using following Eq. (14).

$$fin_i^t \leq dl_i^t \tag{14}$$

After identifying how execution time, finish time and deadline constraints makespan is to be evaluated as a schedulers efficiency need to be primarily identified by how much makespan is needed for tasks in cloud computing when a scheduler generates schedules. Generally makespan is defined as execution time of a task corresponding to a VM. Therefore, when makespan is to be minimized effectiveness of scheduler will be improved. It is calculated using Eq. (15).

$$m^i = \max(fin_i^t(vm_j)) \tag{15}$$

$$\min fin_i^t(t_i vm_j) = \sum_{i=1} \sum_{j=1} \zeta_{ij} fin_i^t(t_i vm_j) \tag{16}$$

Where ζ_{ij} indicates a value as 1 if a task t_i is assigned to a vm_j otherwise it is to be kept as 0.

From above Eqs. (15) and (16) we carefully evaluated makespan which is an important perspective for any cloud paradigm while

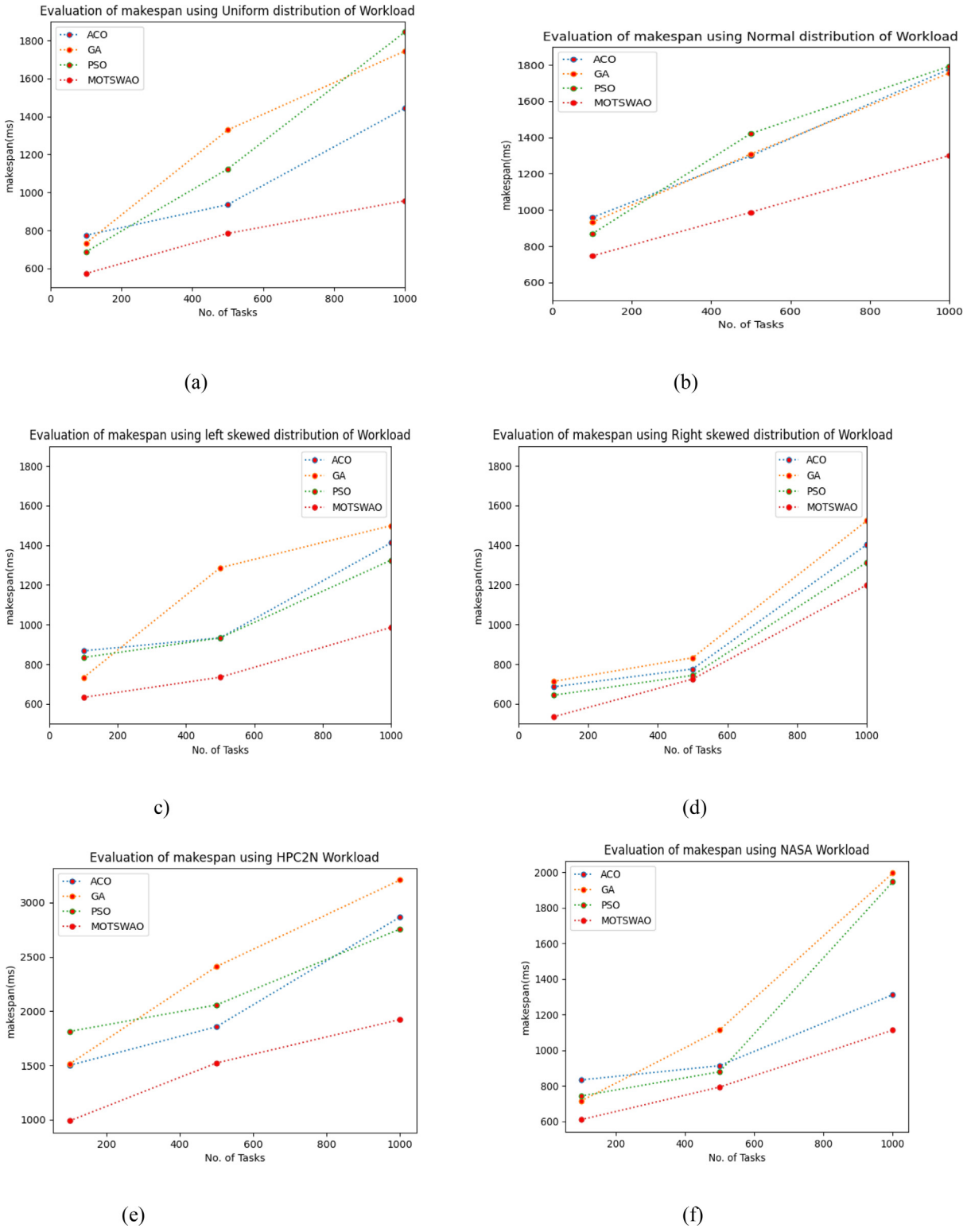

Fig. 3. Calculation of makespan (**a**) calculation of makespan using uniform distribution; (**b**)calculation of makespan using Normal distribution; (c)calculation of makespan using left skewed distribution; (d) calculation of makespan using right skewed distribution; (e) calculation of makespan using HPC2N work logs; (f) calculation of makespan using NASA work logs.

developing a scheduler but minimization of consumption of energy is also an important parameter which gives advantage to cloud consumer and cloud service provider. Energy consumption of a cloud paradigm mainly depends on consumption of energy at idle time, computation time. It is calculated using below Eq. (17).

$$vm_j = \begin{cases} \tau_j \\ \rho_j \end{cases} \tag{17}$$

Where τ_j indicates active state of a vm and ρ_j indicates idle state of a vm. Therefore consumption of energy for all vm s are calculated using below Eq. (18)

Table 5
Calculation of Energy Consumption using various workloads.

Algorithm				
No. of tasks	**ACO**	**GA**	**PSO**	**MOTSWAO**
S01	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	72.4 2.45 67.45	75.72 1.21 71.32	63.45 1.23 60.78	45.78 0.34 40.38
500	104.4 1.21 93.24	98.21 2.18 94.56	103.11 2.35 91.32	57.32 0.26 51.21
1000	143.8 0.89 123.5	171.32 3.21 167.32	148.76 2.12 135.78	120.67 0.56 112.34
S02	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	94.12 1.24 92.14	92.67 1.98 89.67	87.24 1.32 84.23	68.98 0.43 53.45
500	108.32 2.01 102.8	127.67 2.56 96.45	109.35 2.67 97.89	87.46 0.24 79.34
1000	178.56 3.21 148.9	176.32 2.57 167.24	134.78 1.24 131.54	123.45 0.67 67.56
S03	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	79.46 1.38 73.23	67.56 2.45 62.16	74.75 1.34 69.32	60.32 0.32 56.34
500	94.78 2.24 85.23	123.32 1.98 109.34	113.45 2.34 108.21	78.34 0.56 70.67
1000	165.3 3.23 145.34	153.45 1.89 148.67	153.21 1.87 126.32	112.65 0.97 92.45
S04	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	69.27 2.12 67.56	67.45 1.99 59.56	62.67 1.78 60.57	47.78 0.98 44.32
500	86.89 5.23 78.46	132.17 1.78 121.31	113.45 1.55 109.23	82.26 0.76 70.88
1000	139.12 1.89 134.24	178.65 1.67 165.78	128.45 1.21 112.78	97.12 0.34 90.21
S05	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	86.24 1.43 81.32	90.78 1.58 88.34	82.87 2.71 79.67	64.56 1.21 57.21
500	112.32 1.89 107.21	117.8 1.92 112.31	104.67 1.88 101.46	82.12 0.67 79.11
1000	134.21 1.78 123.5	132.67 1.54 128.99	132.78 2.12 121.67	100.57 0.12 98.45
S06	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	82.15 1.32 76.43	89.75 2.78 74.36	78.46 2.16 71.34	51.99 0.76 48.98
500	92.77 2.67 86.56	109.45 1.88 98.23	95.44 2.78 91.45	78.56 0.68 67.32
1000	134.14 3.57 117.67	130.56 0.99 124.67	127.67 2.54 108.56	89.67 0.34 80.88

$$ene_j^{conv} = fin_i^t * \tau_j + (m^i - fin_i^t) * \rho_j \tag{18}$$

$$act_{con}^{ene} = \left(ene_{\{max\}}^{con} - ene_{\{max\}}^{con}\right) * re^{ut} + ene_{\{max\}}^{con} \tag{19}$$

Therefore overall consumption of energy calculated using Eq. (20) and it is calculated as below.

$$ene_{con} = \sum ene_j^{conv} + act_{con}^{ene} \tag{20}$$

We calculated total running time of all tasks using below Eq. (21).

$$Tot_{running} = \sum ex_i^t \tag{21}$$

3.3. Multi objective optimization fitness function for MOTSWAO scheduler

Now we evaluated and modeled our scheduler by using mathematical modeling from Eqs. (1)–(20) and now to optimize parameters we used an optimization model i.e. whale optimization and before using that algorithm we defined a fitness function carefully which is a Multi Objective optimization model and it is identified using below Eq. (22).

$$f(x) = \zeta_1 * m^i + \zeta_2 * ene_{con} + \zeta_3 * av(vm_j) + \zeta_4 * sr(vm_j) + \zeta_5 * te(vm_j) + \zeta_6 * Tot_{running} \tag{22}$$

$$\zeta_1 + \zeta_2 + \zeta_3 + \zeta_4 + \zeta_5 + \zeta_6 = 1 \tag{23}$$

By using above Eqs. (22) and (23) we evaluate parameters mentioned in our work. The above fitness function need to minimize makespan, energy consumption, Total running time while improving availability, success rate, turnaround efficiency while following deadline constraint and thereby trust value will be optimized in turn user can take a decision to choose a particular cloud provider based on these optimized parameters. Therefore, to optimize above mentioned parameters we used a nature inspired algorithm to model our proposed scheduling algorithm. In next section, we briefly discussed about proposed multi-objective trust aware task scheduling algorithm modeled by whale optimization.

3.4. Multi objective trust aware task scheduling algorithm using whale optimization

This section discusses about proposed algorithm through which scheduler built up in this manuscript. The proposed approach developed using Whale Optimization algorithm (Jia et al., 2021) which is a metaheuristic and nature inspired approach which works based on humpback whales. This algorithm initially generates random population of whales as agent population $\vec{M} = (1, 2, \cdots ..K)$ and then out of this population best search agent identified as m^*. Then, after best search identification, fitness of that agent calculated using Eq. (21). After calculating fitness function position of target have to be identified and if current target is best solution then update it as best solution and it is calculated using below Eq. (24).

$$\overrightarrow{Be} = |\vec{N}.\overrightarrow{M^*(x)} - \vec{M}| \tag{24}$$

In above equation $\vec{N}$ represents coefficient vector, $\vec{M}$ represents Position vector, $\overrightarrow{M^*}$ represents best solution of position vector of present iteration x.

After identification of best solution using Eq. (24), position of next best solution for search agent identified using below Eq. (25).

$$\vec{M}(x+1) = \overrightarrow{M^*(x)} - \vec{S}.\overrightarrow{Be} \tag{25}$$

Where $\vec{S}$, $\vec{N}$ are coefficient vectors and those coefficient vectors are calculated as below.

$$\vec{S} = 2\vec{p}.\vec{q} - \vec{p} \tag{26}$$

$$\vec{N} = 2.\vec{q} \tag{27}$$

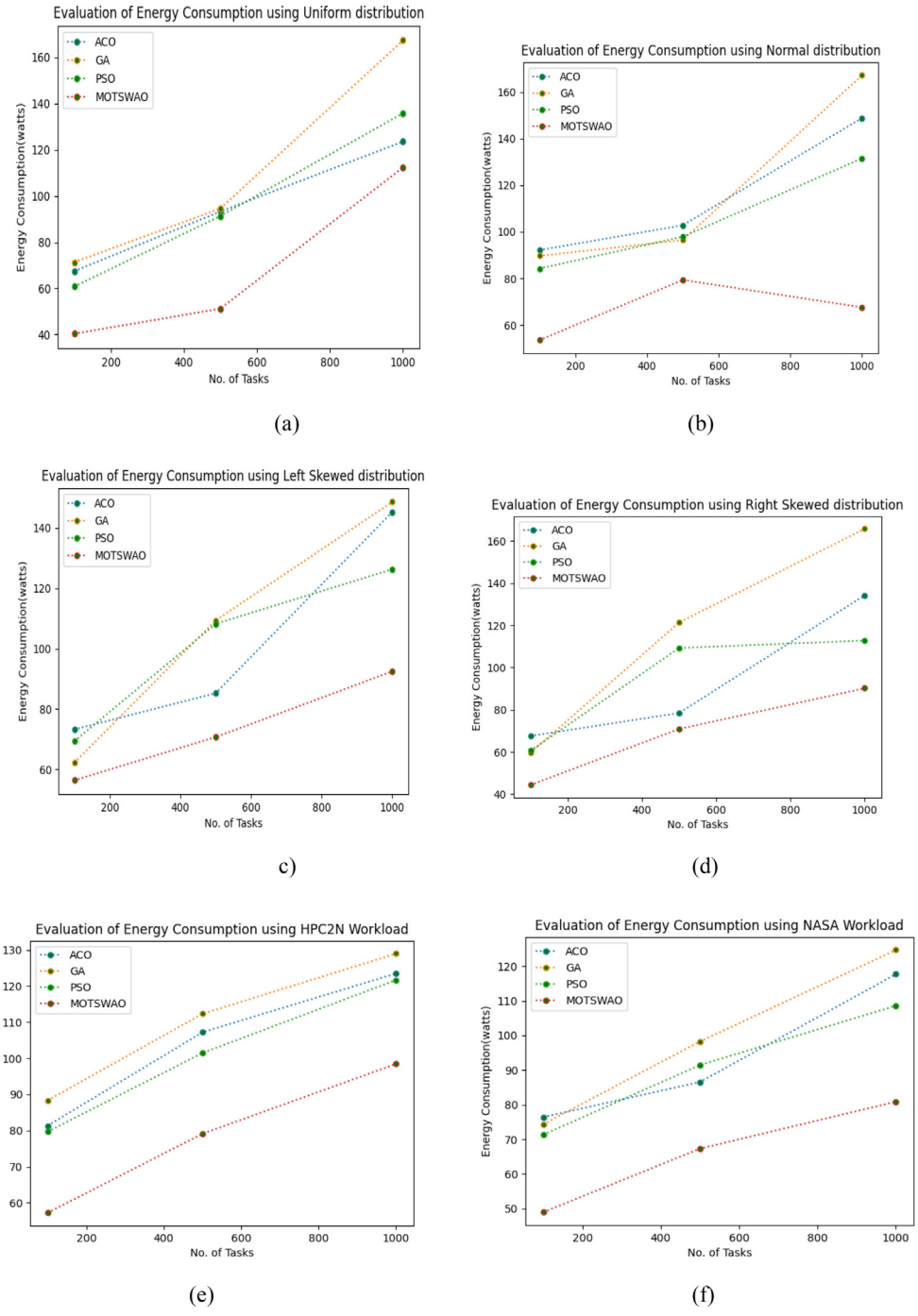

Fig. 4. Calculation of Energy Consumption (**a**) calculation of Energy Consumption using uniform distribution; (**b**)calculation of Energy Consumption using Normal distribution; (c)calculation of Energy Consumption using left skewed distribution; (d) calculation of Energy Consumption using right skewed distribution; (e) calculation of Energy Consumption using HPC2N work logs; (f) calculation of Energy Consumption using NASA work logs.

From Eqs. (26) and (27) $\overrightarrow{p}$ and $\overrightarrow{q}$ are decreases from 2 to 0 and those are random vectors lies in between 0 and 1. After best search agent identification coefficient vectors looking for next position and looking around problem search space and looking for best search agent by exploiting search space then exploitation of search space starts and it mainly divided into two steps i.e. shrinking and spiral updation phases in which coefficient vectors changed to a new range i.e. [-1,1].

In spiral updation, agent position calculated using Eq. (28).

$$\overrightarrow{M}(x+1) = \overrightarrow{Be}^{f}.b^{zr}.\cos(2\Pi r) + \overrightarrow{M^{*}(x)} \tag{28}$$

Where z represents constant and r represents value in between [-1,1]. Then from Eq. (27) $\overrightarrow{Be}^{f}$ calculated as below.

$$\overrightarrow{Be}^{f} = |\overrightarrow{M^{*}(x)} - \overrightarrow{M(x)}| \tag{29}$$

From Eq. (29) $\overrightarrow{M(x)}$ represents current vector of a solution and $\overrightarrow{M^{*}(x)}$ best solution of a corresponding vector. After exploitation phase either by using shrinking or spiral updation then exploration phase starts based on a random search agent solutions can be identified in this phase.

$$\overrightarrow{Be} = |\overrightarrow{N}.\overrightarrow{M_{rd}} - \overrightarrow{M}| \tag{30}$$

Table 6
Calculation of Availability using various workloads.

Algorithm				
No. of tasks	**ACO**	**GA**	**PSO**	**MOTSWAO**
S01	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	85.62 1.23 79.43	92.1 0.74 86.54	89.67 1.89 83.45	94.56 0.88 90.36
500	78.12 2.14 69.43	85.32 0.93 79.34	79.88 1.45 73.87	92.88 0.45 89.98
1000	68.54 1.12 54.78	74.67 0.45 63.46	81.88 1.64 78.46	97.32 0.67 93.45
S02	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	82.77 1.67 78.67	91.78 0.99 87.56	92.78 1.32 88.34	95.35 0.96 91.34
500	78.56 1.54 70.35	88.34 0.67 80.23	86.67 1.01 78.12	93.21 0.76 88.34
1000	67.34 1.21 62.16	76.78 0.58 73.45	79.99 1.87 72.15	96.16 0.45 93.56
S03	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	67.56 1.76 63.88	77.56 1.67 72.44	69.88 2.34 63.54	97.88 0.23 95.21
500	75.44 1.28 70.13	84.78 0.87 79.35	82.67 1.88 75.45	96.78 0.55 91.33
1000	80.67 1.88 79.45	74.32 2.59 68.67	78.23 1.26 71.58	98.87 0.71 93.99
S04	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	64.78 1.94 58.44	73.89 1.67 69.99	78.44 2.87 69.98	98.65 0.56 96.13
500	75.67 2.89 69.65	77.67 2.31 70.65	81.34 1.98 74.67	97.41 0.47 95.42
1000	81.67 2.78 72.78	84.89 1.92 80.67	89.98 1.56 80.78	98.79 0.87 96.37
S05	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	54.55 2.78 52.17	67.87 1.53 64.35	71.67 1.89 68.67	91.78 0.87 89.21
500	68.78 3.18 62.86	63.34 1.92 58.11	74.56 2.78 70.38	96.46 0.91 93.48
1000	73.17 3.29 69.87	70.56 2.88 65.46	80.67 3.32 75.81	98.12 0.45 97.56
S06	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	52.19 2.89 49.88	56.78 1.34 52.18	69.16 2.88 63.78	90.58 0.59 87.57
500	65.67 3.56 61.87	64.21 2.85 60.21	71.76 2.56 69.43	97.32 0.78 92.34
1000	76.43 2.99 71.34	78.16 3.18 72.67	79.38 3.13 74.98	98.35 0.87 94.32

After identification of best solution in exploration phase, next iterated solution calculated using below Eq. (31).

$$\vec{M}(x+1) = \vec{M_{rd}} - \vec{S}.\vec{Be} \tag{31}$$

After successful completion of both exploitation and exploration phases termination phase begins as if agent moves out of region and then $\vec{M^*}$ is updated and it continues until best solution identified.

Proposed MOTSWAO Scheduling algorithm

Input: Tasks represented as $t_i = \{t_1, t_2, t_3, \cdots\cdots, t_i\}$, virtual resources represented as $vm_j = \{vm_1, vm_2, vm_3 \cdots\cdots., vm_j\}$, hosts represented as $H_k = \{H_1, H_2, H_3, \cdots.., H_k\}$, Datacenters represented as $DC_l = \{DC_1, DC_2, DC_3, \cdots., DC_l\}$

Output: schedules generated by MOTSWAO scheduler while improving $av(vm_j)$, $sr(vm_j)$, $te(vm_j)$ and minimizing m^i, ene_{con}, $Tot_{running}$ and generates trust value.

1.Start
2. Give input i.e. set of tasks t_i, set of VMs vm_j, set of physical hosts H_k, Datacenters DC_l
3.Initialize whale population randomly.
4.Task priorities evaluation using Eq. (6).
5.VM priorities evaluation using Eq. (7).
6.Calculate fitness function using Eq. (22).
7.if $(probability < 0.5)$
8.if $\left(|\vec{s}| < 1\right)$
9.Calculation of search agent's position using Eq. (25).
10.elseif $\left(|\vec{s}| < 1\right)$
11.Calculation of search agent's position using Eq. (31).
12.endif
13.endif

(*continued*)

Proposed MOTSWAO Scheduling algorithm

14.if $(probability \geq 0.5)$
15.Calculation of search agent's position using Eq. (28).
16.endif
17.calculate $av(vm_j)$, $sr(vm_j)$, $te(vm_j)$, m^i, ene_{con}, $Tot_{running}$
18.calculate $trust_{csp}$ using Eq. (11).
19. if current trust is increased then update trust by calculating $trust_{current} = trust_{current} + trust_{csp}$
20. else
21. calculate trust decreased by calculating $trust_{current} = -2^a * trust_{csp}$
22.endif
23.if agent is outside search explorated region
24.update $\vec{M^*}$
25.endif
26.end

In the above algorithm, Initially a random generated whale population spread over problem space. After that, Task priorities are calculated using Eq. (6), VM priorities are calculated using Eq. (7) respectively. In the next step, calculate fitness function using Eq. (22) and if probability of a search agent is less than 0.5 and coefficient vector is less than 1 calculate search agent position using Eqs. (25) and (31). Otherwise if probability of search agent is greater than or equal to 0.5 calculate search agent's position using Eq. (28). After evaluating all calculate availability of a virtual resource, success rate of a virtual resource, turnaround efficiency of a virtual resource, makespan, energy consumption. After calculating parameters mentioned, we finally calculate trust value of a cloud service provider based on above mentioned parameters by calculating trust based on additive increase or in exponential decrease manner. Check whether parameters are optimized or not and check whether it is the best solution or not among all solutions. If it generates best

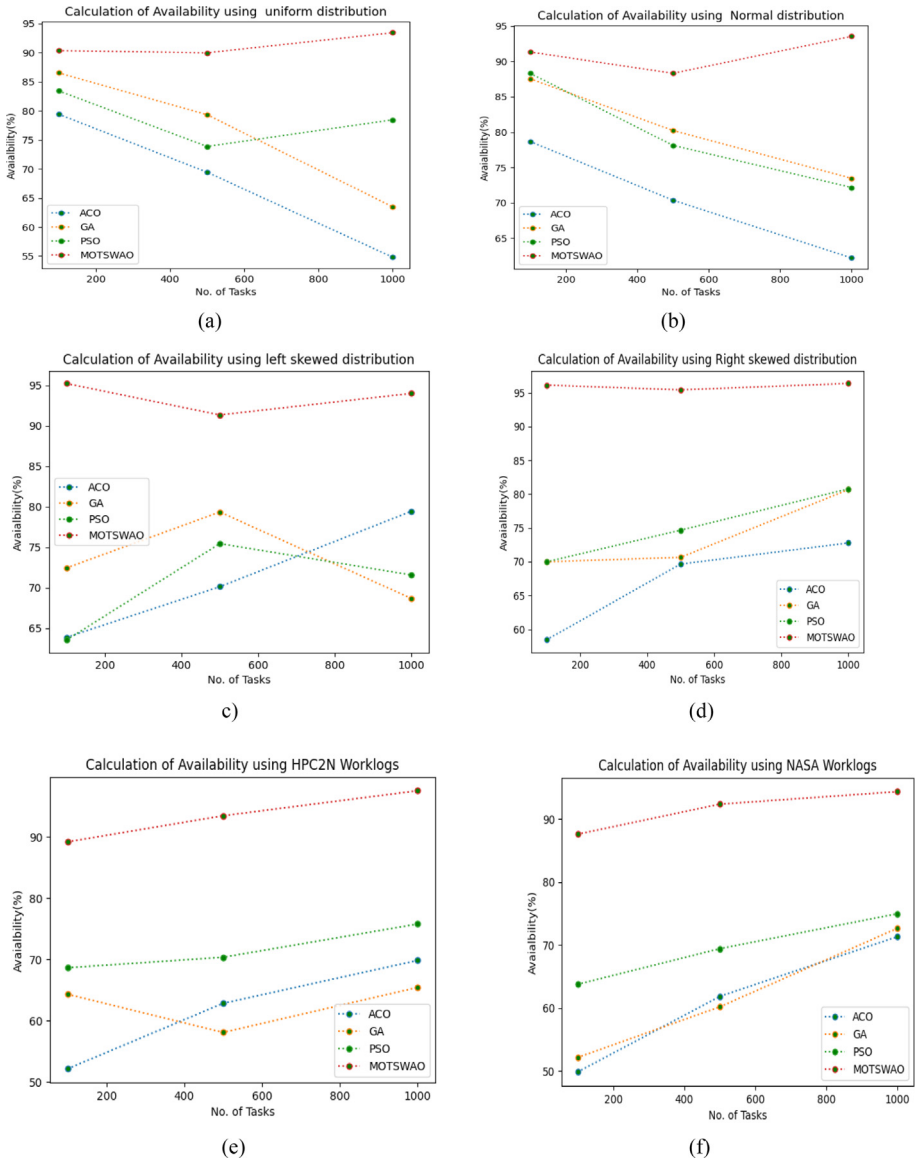

Fig. 5. Calculation of Availability using various workloads (**a**) calculation of Availability using uniform distribution; (**b**)calculation of Availability using Normal distribution; (c)calculation of Availability using left skewed distribution; (d) calculation of Availability using right skewed distribution; (e) calculation of Availability using HPC2N work logs; (f) calculation of Availability using NASA work logs.

schedules and improved the parameters then update it as best solutions and if not continue the same process until it arrives at best solution.

4. Simulation and results

This section clearly discusses about simulations conducted by authors in a detailed manner. Entire extensive simulations are conducted on a very popular simulator named Cloudsim (Calheiros, 2011) in which entire cloud environment can be simulated with the support of java programming. For our research work, we considered workload by fabricating datasets with different distributions and by using real time worklogs of HPC2N (HPC2N, 2016), NASA (NASA). After generating workload, we evaluated SLA based trust parameters, makespan, energy consumption. We compared our proposed approach with baseline approaches i.e. ACO, GA, PSO algorithms with above considered workload.

4.1. Configuration settings for simulation and simulation setup

This subsection presents necessary configuration settings for simulation and as already mentioned simulation conducted in Cloudsim (Calheiros, 2011) for our experimentation. This simulator developed at University of Melbourne. Our physical host configuration setup consists of 16 GB RAM, i7 Processor, windows operating system environment supports virtualization. We generated workload by fabricating datasets as uniform, Normal, left skewed, right skewed distributions and those are indicated as S01,S02,S03,S04 respectively. After generating fabricated workload from datasets with different distributions, we also used real time workload traces from HPC2N (HPC2N, 2016), NASA (NASA) computing clusters and these workloads are indicated as S05,S06 respectively. S01 is uniform distribution workload which consists of all equal distribution of tasks. S02 is Normal distribution workload which consists of high number of medium tasks and less number of small, high tasks. S03 is left skewed distribution workload consists high number of

Table 7
Calculation of Success rate using various workloads.

Algorithm				
No. of tasks	**ACO**	**GA**	**PSO**	**MOTSWAO**
S01	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	83.67 1.88 76.58	90.7 0.23 83.21	86.96 2.76 80.54	95.98 0.75 91.45
500	73.18 2.47 65.72	78.67 1.87 72.46	72.67 1.88 68.34	94.57 0.41 90.67
1000	63.87 2.99 51.76	75.98 0.99 65.98	78.23 1.56 71.75	98.54 0.35 92.67
S02	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	65.32 1.24 62.17	72.78 1.89 66.57	78.87 2.97 70.12	96.43 0.56 91.89
500	71.78 2.65 68.98	79.45 1.98 70.52	85.12 1.35 80.23	95.12 0.87 92.12
1000	78.67 1.28 75.88	80.48 2.24 74.36	77.86 1.78 71.34	97.69 0.19 90.31
S03	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	69.78 2.89 61.79	73.88 3.76 70.58	75.18 2.12 69.19	94.59 0.44 93.18
500	76.87 2.77 71.26	81.09 3.12 73.43	80.79 3.34 73.67	97.46 0.98 92.43
1000	82.45 3.05 80.57	70.21 1.88 69.18	77.45 2.18 72.36	97.65 0.56 90.77
S04	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	53.78 1.79 43.56	71.67 4.23 61.54	68.45 1.87 62.23	98.65 1.12 92.19
500	69.99 3.18 62.14	65.87 3.67 60.23	70.87 3.88 66.37	95.34 1.56 93.58
1000	78.33 3.99 69.54	80.12 4.56 75.14	82.45 2.76 77.88	98.76 1.24 94.99
S05	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	52.31 3.12 48.78	63.55 1.34 57.44	67.09 2.87 63.87	95.12 0.24 94.39
500	59.88 2.78 51.32	56.23 2.87 49.88	71.98 3.66 68.13	97.18 0.87 96.21
1000	71.45 2.85 68.44	76.12 3.45 70.99	82.18 2.57 78.36	96.35 2.33 94.11
S06	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	58.17 1.35 52.67	55.32 2.98 49.67	62.17 1.43 58.45	94.21 0.97 90.78
500	65.32 2.77 60.08	60.15 3.02 55.32	70.57 2.98 64.36	96.34 1.87 93.67
1000	72.38 3.69 68.87	72.10 4.13 68.99	65.99 1.78 60.85	99.76 0.84 95.43

small tasks and less number of large tasks. Finally S04 is right skewed distribution workload consists of less number of small tasks and high number of large tasks. We used a standard simulation settings for our work and it is taken from (Madni, 2019). The below Table 3 represents configuration settings used for simulation.

4.2. Calculation of makespan

In our research work, we calculated makespan for our MOTSWAO scheduler and it is needed to calculate makespan while designing a scheduler as effectiveness of any scheduler depends on with generated makespan. Minimization of makespan improves performance of scheduler. Therefore, we calculated makespan by giving 100 to 1000 tasks and ran simulation for 50 iterations. We used settings in above Table 3 for simulation and given workloads from S01,S02,S03,S04,S05,S06 respectively. We compared our proposed MOTSWAO approach with baseline algorithms i.e. ACO, GA, PSO approaches. The below Table 4 indicates generated makespan for different workloads given as input to MOTSWAO scheduler and from below Fig. 3 it is clearly evident that our proposed MOTSWAO scheduler outperforms baseline approaches for generated makespan for 50 iterations for the considered tasks.

4.3. Calculation of energy consumption

After calculation of makespan, we calculated Energy Consumption for our MOTSWAO scheduler and it is needed to calculate Energy Consumption while designing a scheduler as it is an important parameter from both the perspectives of Cloud Consumer and Service Provider. Minimization of Energy consumption benefits cloud provider in generating huge power bills and benefits to cloud consumer benefiting in receiving cloud services for a cheaper cost and benefits to environment while minimizing energy consumption. Therefore, we calculated Energy Consumption by giving 100 to 1000 tasks and ran simulation for 50 iterations. We used settings in above Table 3 for simulation and given workloads from S01,S02,S03,S04,S05,S06 respectively. We compared our proposed MOTSWAO approach with baseline algorithms i.e. ACO, GA, PSO approaches. The below Table 5 indicates generated Energy Consumption for different workloads given as input to MOTSWAO scheduler and from below Fig. 4 it is clearly evident that our proposed MOTSWAO scheduler outperforms baseline approaches for generated Energy Consumption for 50 iterations for the considered tasks.

4.4. Calculation of Availability

After calculating makespan, Energy Consumption for our MOTSWAO scheduler, we calculated SLA based parameters to improve trust and it is needed to calculate availability as it is an important parameter from both the perspectives of Cloud Consumer and Service Provider. Improvement of availability percentage benefits cloud provider in view of quality of service which improves reliability and thereby trust on Cloud provider will be improved. Therefore, we calculated Availability of virtual resources by giving 100 to 1000 tasks and ran simulation for 50 iterations. We used settings in above Table 3 for simulation and given workloads from S01,S02, S03,S04,S05,S06 respectively. We compared our proposed MOTSWAO approach with baseline algorithms i.e. ACO, GA, PSO approaches. The below Table 6 indicates generated Availability for different workloads given as input to MOTSWAO scheduler and from below Fig. 5 it is clearly evident that our proposed MOTSWAO scheduler outperforms baseline approaches for generated availability for 50 iterations for the considered tasks.

4.5. Calculation of Success rate

After calculating makespan, Energy Consumption for our MOTSWAO scheduler, we calculated SLA based parameters to improve trust and it is needed to calculate success rate as it is an important parameter from both the perspectives of Cloud Consumer and Service Provider. Improvement of success rate percentage benefits cloud provider in view of quality of service which improves relia-

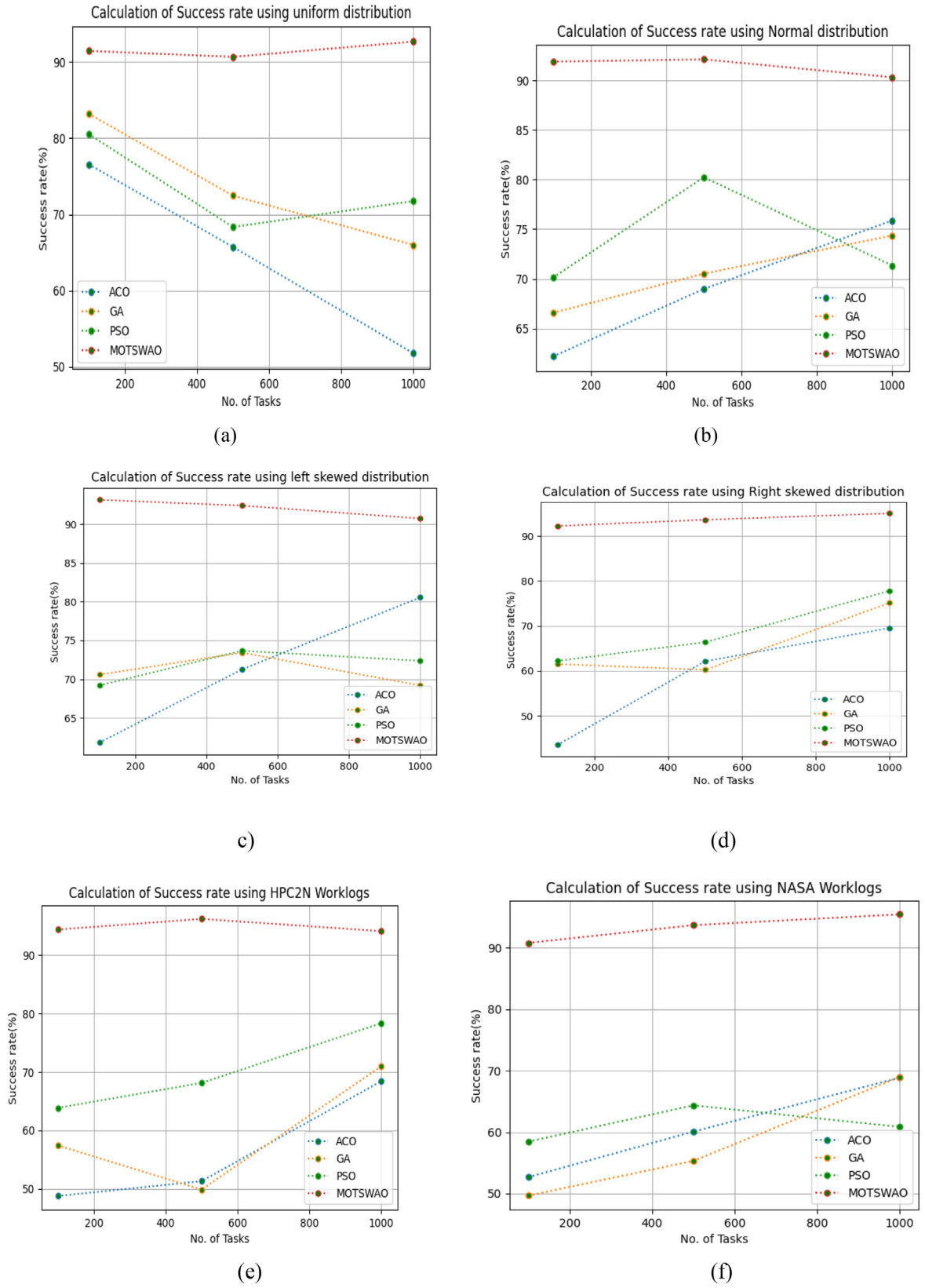

Fig. 6. Calculation of Success rate using various workloads (**a**) calculation of Success rate using uniform distribution; (**b**)calculation of Success rate using Normal distribution; (c)calculation of Success rate using left skewed distribution; (d) calculation of Success rate using right skewed distribution; (e) calculation of Success rate using HPC2N work logs; (f) calculation of Success rate using NASA work logs.

bility and thereby trust on Cloud provider will be improved. Therefore, we calculated success rate of virtual resources by giving 100 to 1000 tasks and ran simulation for 50 iterations. We used settings in above Table 3 for simulation and given workloads from S01,S02,S03,S04,S05,S06 respectively. We compared our proposed MOTSWAO approach with baseline algorithms i.e. ACO, GA, PSO

Table 8
Calculation of Turnaround efficiency using various workloads.

Algorithm				
No. of tasks	**ACO**	**GA**	**PSO**	**MOTSWAO**
S01	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	54.36 2.67 47.98	82.1 2.78 80.87	76.44 3.11 70.99	94.56 0.64 89.77
500	67.99 1.96 59.88	71.75 3.57 67.98	64.36 2.15 61.23	97.68 0.53 92.88
1000	59.96 3.12 52.18	65.34 3.24 61.78	71.33 2.99 68.77	99.65 0.87 90.32
S02	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	62.67 1.81 59.88	67.99 2.87 63.34	71.54 2.35 67.77	95.78 2.56 91.67
500	68.32 3.16 61.67	70.88 1.57 65.62	77.56 2.78 70.12	96.12 3.32 93.34
1000	72.45 2.77 69.53	78.43 2.87 71.66	83.67 3.36 78.24	98.78 1.78 95.66
S03	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	64.57 1.12 59.34	69.02 1.77 67.86	74.9 3.99 70.35	95.78 0.89 94.36
500	71.35 1.88 65.66	73.56 2.09 70.34	79.12 2.88 72.35	98.12 0.12 97.88
1000	80.39 1.34 76.88	82.88 1.12 76.88	85.32 3.09 81.67	99.24 0.32 98.11
S04	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	52.63 1.97 49.88	73.88 3.99 67.92	71.21 1.23 69.71	96.58 2.36 91.87
500	67.87 3.43 61.32	80.77 2.86 72.98	74.99 4.27 67.34	93.25 2.99 92.99
1000	76.45 2.87 71.36	85.48 3.78 79.46	85.37 3.87 81.77	99.09 1.98 93.78
S05	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	48.36 3.98 43.56	56.77 1.01 51.23	63.45 2.43 57.36	97.56 0.36 95.87
500	57.45 2.96 49.67	58.78 1.12 55.67	76.77 4.24 70.15	99.32 0.62 94.56
1000	70.77 2.32 63.87	59.35 1.04 51.12	85.21 3.87 76.65	98.78 0.88 98.89
S06	$\vec{z}$ λ **best**	$\vec{\vec{z}}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	64.76 1.32 60.37	58.32 3.13 53.67	69.41 1.67 63.84	96.88 0.34 91.37
500	72.19 3.58 64.24	62.65 3.53 58.36	75.36 1.88 69.77	98.56 2.77 95.41
1000	77.86 3.45 71.23	74.67 2.45 72.33	80.23 2.33 77.36	98.99 0.56 96.37

approaches. The below Table 7 indicates generated success rate for different workloads given as input to MOTSWAO scheduler and from below Fig. 6 it is clearly evident that our proposed MOTSWAO scheduler outperforms baseline approaches for generated success rate for 50 iterations for the considered tasks.

4.6. Calculation of turnaround efficiency

After calculating makespan, Energy Consumption for our MOTSWAO scheduler, we calculated SLA based parameters to improve trust and it is needed to calculate turnaround efficiency as it is an important parameter from both the perspectives of Cloud Consumer and Service Provider. Improvement of turnaround efficiency percentage benefits cloud provider in view of quality of service which improves reliability and thereby trust on Cloud provider will be improved. Therefore, we calculated turnaround efficiency of virtual resources by giving 100 to 1000 tasks and ran simulation for 50 iterations. We used settings in above Table 3 for simulation and given workloads from S01,S02,S03,S04,S05,S06 respectively. We compared our proposed MOTSWAO approach with baseline algorithms i.e. ACO, GA, PSO approaches. The below Table 8 indicates generated turnaround efficiency for different workloads given as input to MOTSWAO scheduler and from below Fig. 7 it is clearly evident that our proposed MOTSWAO scheduler outperforms baseline approaches for generated turnaround efficiency for 50 iterations for the considered tasks.

4.7. Calculation of Total running time

After calculating makespan, Energy Consumption for our MOTSWAO scheduler, we calculated SLA based parameters to improve trust and it is needed to calculate total running time as it is an important parameter from both the perspectives of Cloud Consumer and Service Provider. Minimization of Total running time benefits cloud provider and user will also tend to choose service provider for customer selection and thereby quality of service will increase and in turn trust improves on cloud service provider. Therefore, we calculated total running time of tasks by giving 100 to 1000 tasks and ran simulation for 50 iterations. We used settings in above Table 3 for simulation and given workloads from S01,S02,S03,S04,S05,S06 respectively. We compared our proposed MOTSWAO approach with baseline algorithms i.e. ACO, GA, PSO approaches. The below Table 9 indicates generated turnaround efficiency for different workloads given as input to MOTSWAO scheduler and from below Fig. 8 it is clearly evident that our proposed MOTSWAO scheduler outperforms baseline approaches for generated turnaround efficiency for 50 iterations for the considered tasks.

4.8. Discussion of results

This subsection clearly discusses simulated results analysis and improvement of parameters which we mentioned in subsections 4.2,4.3,4.4,5.4,4.6,4.7 We already mentioned that we used Cloudsim (Calheiros, 2011) for extensive simulations. Our entire simulations carried out with different datasets fabricated and with real time worklogs and they are mentioned as S01,S02,S03,S04,S05 and S06 respectively. We compared our proposed MOTSWAO scheduler with existing baseline algorithms i.e. ACO, GA, PSO by running simulation with 50 iterations. We already presented results for our measured parameters in above sub sections of 4.2, 4.3,4.4,4.5,4.6,4.7. From results, we draw inferences in such a way that how far our proposed MOTSWAO improved parameters and generated schedules effectively. We presented improvement of makespan over baseline approaches in Table 10 for different workloads we used and in Table 11 we presented improvement of energy consumption over baseline approaches for different workloads and in Table 12 we presented improvement of Availability over baseline approaches for different workloads and in Table 13 we presented improvement of Success rate over baseline approaches for different workloads, in Table 14 we presented improvement of turnaround efficiency over baseline approaches for different workloads and finally in Table 15 we presented improvement of Total running time of tasks over baseline

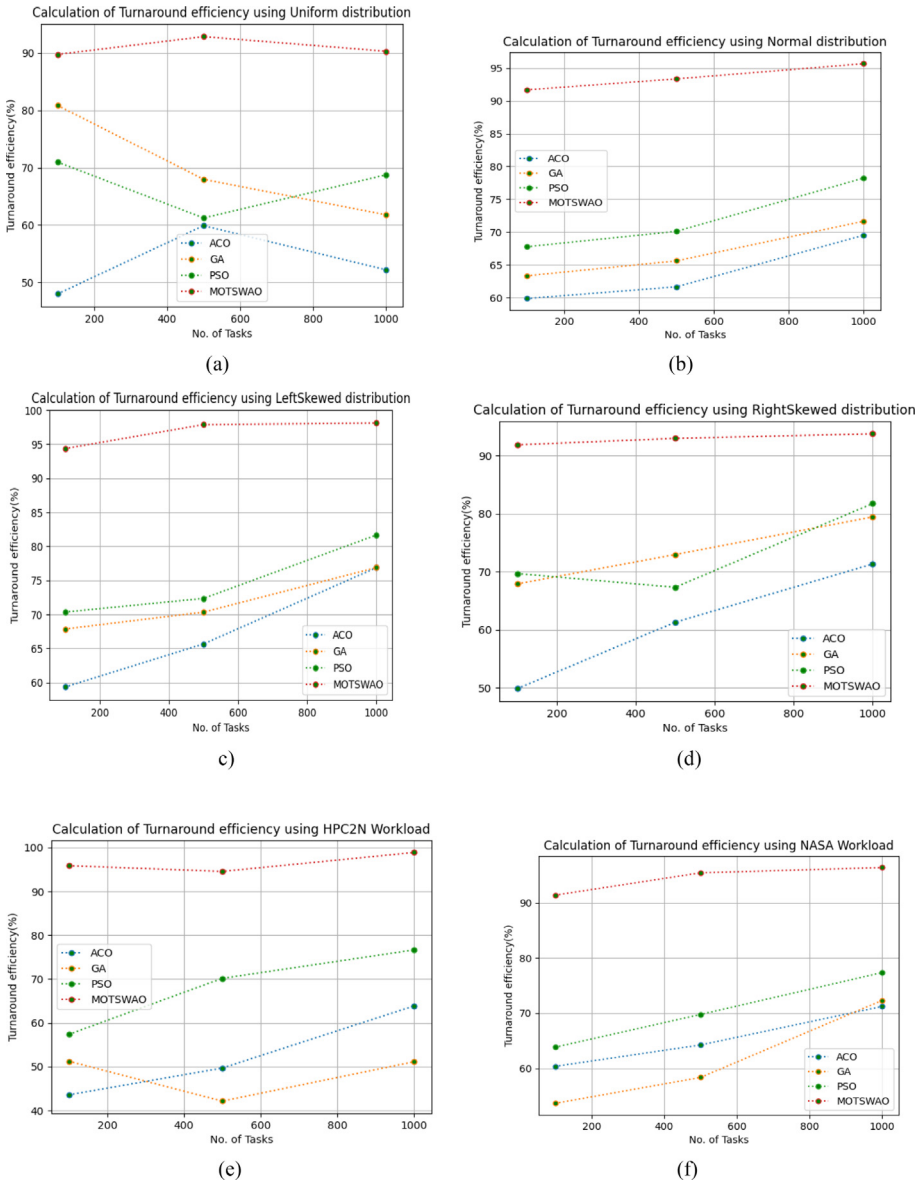

Fig. 7. Calculation of Turnaround efficiency using various workloads (**a**) calculation of Turnaround efficiency using uniform distribution; (**b**)calculation of Turnaround efficiency using Normal distribution; (c)calculation of Turnaround efficiency using left skewed distribution; (d) calculation of v using right skewed distribution; (e) calculation of Turnaround efficiency using HPC2N work logs; (f) calculation of Turnaround efficiency using NASA work logs.

Table 9
Calculation of Total Running time using various workloads.

Algorithm				
No. of tasks	**ACO**	**GA**	**PSO**	**MOTSWAO**
S01	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	723.7 1.35 714.7	757.9 1.57 726.78	687.34 1.53 643.67	524.9 1.12 512.76
500	921.8 1.68 912.36	1287.6 2.43 1226.7	1153.9 1.13 1109.2	754.7 0.28 739.8
1000	1489.9 1.88 1424.5	1757.5 1.35 1721.34	1843.2 1.76 1812.35	976.31 0.53 921.35
S02	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	947.87 1.37 921.47	938.34 2.54 909.17	887.42 1.65 845.77	776.31 1.25 724.36
500	1334.5 1.65 1309.87	1352.76 3.72 1323.41	1435.7 0.26 1415.78	1058.12 1.86 975.37
1000	1747.9 0.63 1721.12	1756.32 2.12 1709.241	1834.7 1.12 1732.51	1321.46 2.16 1309.87
S03	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	856.21 2.08 837.8	724.7 1.09 707.87	827.18 1.32 809.44	664.14 1.03 635.65
500	926.76 1.07 899.65	1309.56 2.76 1267.88	924.53 0.84 912.88	739.4 0.34 712.67
1000	1414.76 2.37 1392.37	1486.32 1.25 1414.54	1356.7 1.65 1321.6	1007.4 0.26 927.87
S04	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	686.73 0.18 624.56	721.98 1.26 709.85	678.17 2.19 632.75	598.34 2.67 523.87

(*continued*)

Algorithm				
No. of tasks	**ACO**	**GA**	**PSO**	**MOTSWAO**
500	734.54 0.35 712.87	825.36 1.16 808.26	776.12 1.36 731.88	721.53 2.56 699.36
1000	1436.72 1.87 1412.47	1534.65 0.76 1498.32	1321.21 3.12 1298.67	1216.79 1.36 1206.8
S05	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	1512.8 4.07 1476.3	1541.34 1.21 1522.7	1832.31 1.76 1809.12	1057.9 1.36 979.37
500	1856.3 2.99 1789.1	2432.6 1.67 2398.9	2132.75 2.57 2087.14	1524.67 2.14 1498.15
1000	2999.6 1.75 2934.22	3185.7 0.83 3098.3	2673.45 2.13 2546.76	1927.36 1.09 1899.31
S06	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**	$\vec{z}$ λ **best**
100	824.72 1.77 799.81	729.86 1.18 718.21	747.35 1.58 724.32	621.67 1.35 595.12
500	935.31 1.16 909.88	1124.34 1.43 1088.35	909.18 1.12 865.16	809.25 1.58 756.29
1000	1324.3 1.06 1295.87	2008.31 1.21 1954.76	1926.31 1.86 1909.31	1056.9 2.09 965.87

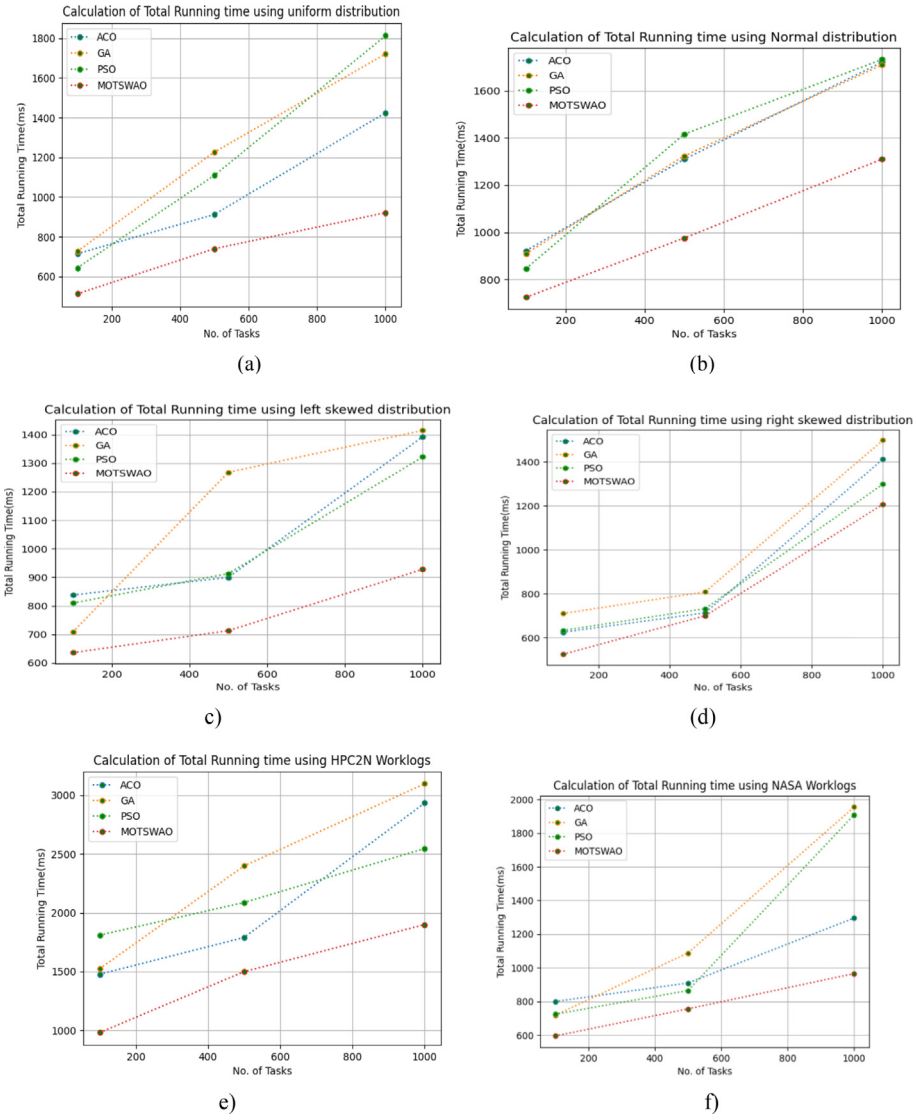

Fig. 8. Calculation of Total running time using various workloads (**a**) calculation of Total running time using uniform distribution; (**b**)calculation of Total running time using Normal distribution; (c) calculation of Total running time using left skewed distribution; (d) calculation of Total running time using right skewed distribution; (e) calculation of Total running time using HPC2N work logs; (f) calculation of Total running time using NASA work logs.

approaches for different workloads. Finally from all these results we can say that our proposed MOTSWAO scheduler improves basic parameters i.e. makespan, energy consumption, Total running time and as well as SLA based trust parameters and thereby improving quality of service and trust improvement for satisfaction of cloud user.

Table 10
Improvement in makespan over existing algorithms.

Algorithms			
Improvement of makespan			
	ACO	**GA**	**PSO**
S01	26 %	39.2 %	36.71 %
S02	24 %	24.08 %	25.69 %
S03	26.74 %	33.12 %	23.89 %
S04	14.15 %	19.86 %	8.9 %
S05	28.7 %	37.83 %	33.09 %
S06	17.74 %	34.21 %	29.49 %

Table 11
Improvement in Energy Consumption over existing algorithms.

Algorithms			
Improvement of Energy Consumption			
	ACO	**GA**	**PSO**
S01	28.24 %	38.79 %	29.16 %
S02	41.72 %	43.3 %	36.12 %
S03	27.76 %	31.45 %	27.77 %
S04	26.7 %	40.74 %	27.3 %
S05	24.76 %	28.78 %	22.47 %
S06	29.74 %	33.66 %	27.34 %

Table 12
Improvement in Availability over existing algorithms.

Algorithms			
Improvement of Availability			
	ACO	**GA**	**PSO**
S01	34.4 %	19.38 %	16.12 %
S02	29.39 %	13.2 %	14.52 %
S03	31.42 %	27.25 %	33.22 %
S04	43.34 %	30.09 %	27.72 %
S05	51.56 %	49.12 %	30.42 %
S06	49.77 %	48.2 %	31.73 %

Table 13
Improvement in Success rate over existing algorithms.

Algorithms			
Improvement of Success rate			
	ACO	**GA**	**PSO**
S01	41.6 %	23.97 %	24.54 %
S02	32.5 %	29.73 %	23.75 %
S03	29.38 %	29.63 %	28.40 %
S04	60.21 %	42.58 %	35.97 %
S05	68.92 %	59.68 %	35.33 %
S06	54.09 %	60.87 %	52.38 %

Table 14
Improvement in Turnaround efficiency over existing algorithms.

Algorithms			
Improvement of Turnaround efficiency			
	ACO	**GA**	**PSO**
S01	70.58 %	29.59 %	35.82 %
S02	46.88 %	30.49 %	29.85 %
S03	43.82 %	3176 %	29.4 %
S04	52.63 %	26.45 %	27.33 %
S05	84.18 %	83.1 %	41.71 %
S06	44.57 %	53.58 %	34.21 %

Table 15
Improvement in Total Running time over existing algorithms.

Algorithms			
Improvement of Total Running time			
	ACO	**GA**	**PSO**
S01	27.49 %	38.53 %	34.26 %
S02	23.6 %	23.32 %	23.28 %
S03	26.08 %	29.46 %	24.39 %
S04	10.85 %	19.7 %	9.57 %
S05	28.39 %	37.3 %	33.16 %
S06	22.64 %	32.74 %	26.6 %

5. Conclusion and future work

Task Scheduling is one of prominent challenge in Cloud Computing paradigm as provisioning and deprovisioning of virtual resources to corresponding user requests is highly dynamic as requests comes from heterogeneous resources and varies with time, size and processing capacity. Therefore ineffective task scheduler in cloud environment leads to degradation of cloud provider's quality of service and violates SLA thereby trust on Cloud service provider reduced. Therefore, it is important to maintain trust and reliability towards cloud consumer by cloud service provider. Therefore an effective scheduler need to be employed in Cloud Computing paradigm. In this manuscript we proposed a MOTSWAO scheduler which evaluates priorities for both tasks and VMs and schedules tasks appropriately to virtual resources. We developed this scheduler using whale optimization. We implemented entire extensive simulation in Cloudsim toolkit. We considered different fabricated and real time worklogs mentioned in our manuscript as S01,S02,S03,S04,S05,S06 respectively. Then our proposed MOTSWAO compared over existing algorithms ACO, PSO, GA. From simulation results, our proposed MOTSWAO proved that it outperforms existing approaches by improvising basic parameters i.e. makespan, energy consumption, Total running time and SLA based trust parameters i.e. Availability, Success rate, Turnaround efficiency. In future, we will employ a ML model into our scheduler to predict upcoming workloads and will check efficacy of scheduler for generating schedules and for optimization of parameters. ML model we want to employ in our future work is we want to use a deep reinforcement learning model which is based on rewards and feedback model through which we schedule workload effectively onto virtual resources and optimize parameters accordingly.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abdullah, Monir, Ebtsam A. Al-Muta'a, and Maher Al-Sanabani. 2019. "Integrated MOPSO algorithms for task scheduling in cloud computing." Journal of Intelligent & Fuzzy Systems 36.2, 1823-1836.

Abualigah, L., Alkhrabsheh, M., 2022. Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing. The Journal of Supercomputing 78 (1), 740–765.

Agarwal, M., Srivastava, G.M.S., 2018. Genetic algorithm-enabled particle swarm optimization (PSOGA)-based task scheduling in cloud computing environment. International Journal of Information Technology & Decision Making 17 (04), 1237–1267.

Agarwal, M., Srivastava, G.M.S., 2019. A PSO algorithm based task scheduling in cloud computing. International Journal of Applied Metaheuristic Computing (IJAMC) 10 (4), 1–17.

Aggarwal, A. et al., 2020. Self adaptive fruit fly algorithm for multiple workflow scheduling in cloud computing environment. Kybernetes.

Ajmal, M.S. et al., 2021. Hybrid ant genetic algorithm for efficient task scheduling in cloud data centers. Computers and Electrical Engineering 95, 107419.
Ali, A. et al., 2021. Multilevel central trust management approach for task scheduling on IoT-based mobile cloud computing. Sensors 22 (1), 108.
Alsaidy, Seema A., Amenah D. Abbood, and Mouayad A. Sahib. 2020. "Heuristic initialization of PSO task scheduling algorithm in cloud computing." Journal of King Saud University-Computer and Information Sciences.
Alsaidy, Seema A., Amenah D. Abbood, and Mouayad A. Sahib 2020. "Heuristic initialization of PSO task scheduling algorithm in cloud computing." Journal of King Saud University-Computer and Information Sciences.
Amer, D.A. et al., 2022. Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing. The Journal of Supercomputing 78 (2), 2793–2818.
Arunarani, A.R., Manjula, D., Sugumaran, V., 2019. Task scheduling techniques in cloud computing: A literature survey. Future Generation Computer Systems 91, 407–415.
Beegom, A.S., Rajasree, M.S., 2019. Integer-pso: a discrete pso algorithm for task scheduling in cloud computing systems. Evolutionary Intelligence 12 (2), 227–239.
Calheiros, R.N. et al., 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and experience 41 (1), 23–50.
Chen, X. et al., 2020. A WOA-based optimization approach for task scheduling in cloud computing systems. IEEE Systems Journal 14 (3), 3117–3128.
Duan, K. et al., 2018. Adaptive incremental genetic algorithm for task scheduling in cloud environments. Symmetry 10 (5), 168.
Dubey, K., Sharma, S.C., 2021. A novel multi-objective CR-PSO task scheduling algorithm with deadline constraint in cloud computing. Sustainable Computing: Informatics and Systems 32, 100605.
Ebadifard, F., Babamir, S.M., 2018. A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment. Concurrency and Computation: Practice and Experience 30 (12), e4368.
Abd Elaziz, Mohamed, and Ibrahim Attiya. "An improved Henry gas solubility optimization algorithm for task scheduling in cloud computing." Artificial Intelligence Review 54.5 (2021): 3599-3637.
Francis, Tina. "A Comparison of Cloud Execution Mechanisms Fog, Edge, and Clone Cloud Computing." International Journal of Electrical & Computer Engineering (2088-8708) 8.6 (2018).
HPC2N: the HPC2N Seth log. http://www.cs.huji.ac.il/labs/par allel/workload/l_hpc2n/.0 (2016).
Jia, LiWei, Li, K., Shi, X., 2021. Cloud computing task scheduling model based on improved whale optimization algorithm. Wireless Communications and Mobile Computing 2021.
Karthika, A., Muthukumaran, N., 2022. An ADS-PAYG approach using trust factor Against economic denial of sustainability attacks in cloud storage. Wireless Personal Communications 122 (1), 69–85.
Madni, S.H.H. et al., 2019. Hybrid gradient descent cuckoo search (HGDCS) algorithm for resource scheduling in IaaS cloud computing environment. Clust. Comput. 22 (1), 301–334.
Mangalampalli, S., Swain, S.K., Mangalampalli, V.K., 2022. Prioritized energy efficient task scheduling algorithm in cloud computing using whale optimization algorithm. Wireless Personal Communications 126 (3), 2231–2247.
Mangalampalli, S., Swain, S.K., Mangalampalli, V.K., 2022. Multi Objective Task Scheduling in Cloud Computing Using Cat Swarm Optimization Algorithm. Arabian Journal for Science and Engineering 47 (2), 1821–1830.
Nabi, S. et al., 2022. AdPSO: adaptive PSO-based task scheduling approach for cloud computing. Sensors 22 (3), 920.
NASA. https://www.cse.huji.ac.il/labs/parallel/workload/l_nasa_ ipsc/.
Pang, S. et al., 2019. An EDA-GA hybrid algorithm for multi-objective task scheduling in cloud computing. IEEE Access 7, 146379–146389.
Panwar, N. et al., 2019. TOPSIS–PSO inspired non-preemptive tasks scheduling algorithm in cloud environment. Cluster Computing 22 (4), 1379–1396.
Pirozmand, P. et al., 2021. Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing. Neural computing and applications 33 (19), 13075–13088.
Prasanna Kumar, K.R., Kousalya, K., 2020. Amelioration of task scheduling in cloud computing using crow search algorithm. Neural Computing and Applications 32 (10), 5901–5907.
Senthil Kumar, A.M., Venkatesan, M., 2019. Task scheduling in a cloud computing environment using HGPSO algorithm. Cluster Computing 22 (1), 2179–2185.
Senthil Kumar, A.M., Venkatesan, M., 2019. Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment. Wireless Personal Communications 107 (4), 1835–1848.
Sharma, N., Garg, P., 2022. Ant colony based optimization model for QoS-based task scheduling in cloud computing environment. Measurement: Sensors 24, 100531.
Sharma, S., Jain, R., 2019. EACO: an enhanced ant colony optimization algorithm for task scheduling in cloud computing. International Journal of Security and Its Applications 13 (4), 91–100.
Singh, A., Chatterjee, K., 2017. A multi-dimensional trust and reputation calculation model for cloud computing environments. 2017 ISEA Asia Security and Privacy (ISEASP).
Soleymani, Mona, et al. 2021. "Fuzzy Rule-Based Trust Management Model for the Security of Cloud Computing." Mathematical Problems in Engineering 2021.
Srichandan, S., Kumar, T.A., Bibhudatta, S., 2018. Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm. Future Computing and Informatics Journal 3 (2), 210–230.
Velliangiri, S. et al., 2021. Hybrid electro search with genetic algorithm for task scheduling in cloud computing. Ain Shams Engineering Journal 12 (1), 631–639.
Zhang, N. et al., 2018. A genetic algorithm-based task scheduling for cloud resource crowd-funding model. International Journal of Communication Systems 31 (1), e3394.
Zhou, Z. et al., 2018. A modified PSO algorithm for task scheduling optimization in cloud computing. Concurrency and Computation: Practice and Experience 30 (24), e4970.