SLA-based task scheduling in cloud computing using randomized PSO algorithm*

Swati Lipsa[1,*,†], Ranjan Kumar Dash[1,†]

[1]School of Computer Sciences, Odisha University of Technology and Research, Bhubaneswar, Odisha - 751029

Abstract

In SLA-based cloud computing environments, efficient task scheduling is critical for optimizing resource utilization and meeting performance objectives outlined in Service Level Agreements (SLAs). The existing literature highlights several limitations in this domain, including the complexity of scheduling tasks across heterogeneous Virtual Machines (VMs) while satisfying various constraints such as profit maximization or makespan minimization. These limitations underscore the need for innovative scheduling algorithms capable of addressing the dynamic nature of cloud environments. These challenges motivated us to develop a randomized Particle Swarm Optimization (PSO) algorithm designed specifically for the task scheduling problem in SLA-based cloud computing environments. This algorithm aims to efficiently allocate diverse tasks to available VMs while adhering to the critical constraints defined by the SLAs. The PSO algorithm employs randomized search strategies to efficiently explore the solution space and determine the optimal task for VM assignments. Further, to assess the effectiveness of the proposed scheduling algorithm, comparative analyses are performed against existing scheduling approaches such as Shortest Job First (SJF) and First Come First Serve (FCFS). These comparisons evaluate the performance, efficiency, and scalability of the randomized PSO algorithm in terms of profit maximization and makespan minimization. The outputs of this comparison provide valuable insights into the efficacy of the proposed algorithm in SLA-based cloud computing environments. Therefore, the findings of this study offer practical implications for enhancing the reliability, scalability, and performance of cloud-based services while ensuring compliance with SLA commitments and customer expectations.

Keywords

Task scheduling, SLA, Cloud Computing, Profit maximization, Makespan minimization, Particle swarm optimization, Genetic algorithm

1. Introduction

Cloud computing has transformed the face of modern computing by offering on-demand resources that are flexible and scalable, reshaping how businesses operate and individuals interact with technology. Cloud computing fundamentally provides computing services like storage, computational capabilities, software, and applications via the internet, thereby eliminating the need for on-premises infrastructure and empowering users with unparalleled accessibility and agility [1]. However, despite the myriad benefits it offers, cloud computing also presents a host of challenges that must be addressed to fully harness its potential. These challenges encompass various aspects, including security concerns, data privacy issues, performance bottlenecks, and reliability constraints, among others. Among the most pressing issues faced by cloud computing providers and users alike is the establishment and maintenance of Service Level Agreements (SLAs).

SLAs serve as contractual agreements between cloud service providers (CSP) and consumers, defining the terms and conditions under which services will be delivered, encompassing performance metrics, assurances of availability, subscription plans, penalties for non-compliance, and recourse mechanisms in case of service interruptions or breakdowns. In essence, SLAs encapsulate the expectations and obligations of both parties, serving as the cornerstone to ensure service quality, reliability, and accountability in the cloud computing ecosystem[2]. The need for robust SLAs in cloud computing stems from the

Symposium on Computing and Intelligent Systems (SCI), May 10, 2024, New Delhi, India

*Corresponding author.

†These authors contributed equally.

slipsait@outr.ac.in (S. Lipsa); rkdash@outr.ac.in (R. K. Dash)

0000-0002-4352-3229 (S. Lipsa); 0000-0003-3482-465X (R. K. Dash)

inherent complexities and uncertainties associated with distributed computing environments. As cloud infrastructures encompass diverse hardware and software components distributed across geographically dispersed data centers, ensuring consistent performance and availability becomes a daunting task. SLAs provide a structured framework for delineating performance objectives, establishing benchmarks, and enforcing compliance standards, thereby fostering trust and transparency among stakeholders [3][4].

Nevertheless, despite the best efforts to uphold SLAs, instances of service disruptions or deviations from agreed-upon performance levels can occur, triggering penalties or financial consequences as outlined in the SLA terms. These penalties serve as a mechanism for persuading cloud service providers to adhere to their commitments and prioritize service reliability and responsiveness.

Task scheduling algorithms play a pivotal role in mitigating SLA violations and optimizing resource utilization in cloud computing environments. By intelligently allocating computing resources and orchestrating task execution based on dynamic workload patterns, task scheduling algorithms can preemptively identify potential bottlenecks, mitigate contention for shared resources, and optimize task completion times, thereby enhancing overall system performance and meeting SLA requirements. Thus, this study proposes a randomized PSO algorithm for task scheduling while meeting constraints such as profit maximization and makespan minimization in an SLA-based cloud computing environment.

The rest of the paper is organized as follows. Section 2 presents an overview of the existing literature. Section 3 describes the proposed SLA-based task scheduling framework. Section 4 discusses the result analysis in detail, followed by conclusion in Section 5.

2. Related work

The existing literature offers a range of operational models and methods for creating and managing SLAs in the context of cloud computing. Furthermore, we delve into various task-scheduling approaches within the framework of SLA-oriented cloud computing. The study[5] introduces an approach that employs artificial neural networks (ANNs) to schedule tasks within cloud data centers, emphasizing the goal of improving energy efficiency. The authors in[6] introduce a genetic algorithm (GA)-based task scheduling method designed for the allocation and execution of diverse tasks within a cloud computing framework. The objective is to reduce both task completion time and execution costs while maximizing the utilization of available resources. The work carried out in[7] proposes an Energy and Performance-Efficient Task Scheduling Algorithm (EPETS) to address the challenge of task scheduling in virtualized cloud environments. By leveraging virtualization technology, the approach aims to balance the workload across different types of resources while minimizing energy consumption and maximizing performance. The paper[8] proposes a job scheduling algorithm that takes into account both the cost and the quality of service (QoS). This algorithm assesses the cost-related QoS parameters of virtual resources from the unified resource layer in real-time. Its goal is to improve user satisfaction and optimize the profits of commercial cloud providers. The study[9] addresses the challenges of managing workloads efficiently in edge-cloud environments by proposing cost-aware automatic scaling and workload-aware replica management solutions, aiming to optimize resource utilization, minimize costs, and enhance performance. The work[10] introduces a cost-based energy-efficient scheduling technique for dynamic voltage frequency scaling (DVFS) systems in cloud computing environments. This technique aims to optimize resource utilization, reduce energy usage, and lower operational costs by prioritizing tasks based on energy consumption and cost considerations.

The paper[11] presents a cost-effective and reliability-aware job scheduling algorithm for cloud computing systems. This algorithm prioritizes job assignments based on service cost and reliability considerations, so as to enhance resource utilization and ensure efficient execution of workloads in cloud environments. The researchers in[12] develop an intelligent cloud task scheduler using Deep Reinforcement Learning (DRL), where the scheduler relies on learning directly from experience without pre-existing knowledge to make optimal scheduling decisions. They frame task scheduling as a dynamic optimization challenge with limitations and utilize the deep deterministic policy gradients (DDPG) network to determine the best task assignment solution while adhering to performance and cost constraints. The authors also propose a correlation-aware state representation approach to capture inherent demand characteristics and design a dual reward model to learn the optimal task allocation strategy. The paper[13] utilizes deep reinforcement learning (DRL) techniques to make real-time decisions about job allocation across various cloud resources to improve resource utilization and reduce overall costs in hybrid cloud deployments. The study[14] presents a combined heuristic approach designed to optimize task scheduling in fog-cloud computing scenarios, with a focus on QoS demands and cost-effectiveness. The key goal of cost reduction is pursued by implementing a unique model called Hybrid Flamingo Search with a Genetic Algorithm (HFSGA) for improved task scheduling. The authors in[15] introduce a job scheduler driven by Deep Reinforcement Learning (DRL) to manage the real-time dispatch of jobs. Its primary objective is to organize user requests effectively to ensure quality of service (QoS) for end-users while also substantially cutting down on the expenses associated with executing jobs on virtual instances. The method proposed relies on a Deep Q-learning Network (DQN) model and seeks to diminish Virtual Machine (VM) costs while upholding high QoS for incoming jobs in cloud frameworks.

The paper[16] suggests a novel approach named Ant Grey Wolf Optimization (AGWO), which integrates the principles of Ant Colony Optimization (ACO) and Grey Wolf Optimization (GWO). This hybrid method aims to accelerate and enhance the quality of solution discovery in ACO. AGWO primarily focuses on improving the efficiency and results of cloud-fog computing frameworks by optimally scheduling IoT task requests. The work carried out in[17] proposed a heuristic approach for task scheduling in cloud computing. The authors in[18] introduce an innovative Grey Wolf Optimizer-based Task Scheduling (GWOTS) algorithm, which is designed to assign tasks to available resources. This meta-heuristic technique serves as a solution to optimization challenges. The study[19] presents a preemptive approach based on Deep Reinforcement Learning (DRL). This approach enhances the training of the scheduling policy through efficient preemptive mechanisms for jobs, to minimize the cost of job execution while ensuring the expected response time in a cloud setting is met. The paper[20] proposes an SLA-aware task-scheduling algorithm based on the Whale Optimization Algorithm for cloud computing systems. By prioritizing task scheduling based on SLA requirements and leveraging WOA for optimization, the algorithm aims to enhance resource utilization, meet SLAs, and improve the user experience in cloud environments. The work[21] presents a methodology for service placement and scheduling on the continuum from edge to cloud computing, prioritizing cost awareness. The approach can adjust to dynamic environments while maintaining optimal performance.

However, the above-discussed literature faces certain limitations such as the diverse nature of data centers and the varying resource requirements of user applications, inherent trade-offs between profit maximization, makespan minimization, and SLA compliance, resource heterogeneity by having VMs with varying capacities and performance characteristics and maximizing profit or minimizing makespan without appropriate alignment with energy efficiency goals in cloud data centers. So to address these limitations, we propose an SLA-based task scheduling algorithm using a randomized PSO technique, with a focus on designing scalable, and efficient task scheduling algorithms that can optimize resource utilization, minimize makespan, and ensure SLA compliance in cloud environments.

3. SLA-based task scheduling framework

3.1. SLA-based task scheduling problem formulation

Let $VM_1, VM_2, \cdots VM_m$ are the m number of VMs provided by the Cloud Service Provider (CSP). The available tasks be $T_1, T_2, \cdots T_n$.

3.1.1. Allocation of a task to VM

Allocation of a task to VM is defined by the following binary variable:

$$A(T_i, VM_j) = \begin{cases} 1 & \text{if } T_i \text{ is allocated to } VM_j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

3.1.2. Gain cost

Gain cost is the defined cost in the agreement between the user and CSP to accomplish the execution of the submitted tasks. The gain cost of a particular VM can be calculated in terms of the execution time of the total number of tasks processed by that VM i.e.

$$G(VM_j) = \sum_{i=1}^{n} E(T_i) \times A(T_i, VM_j) \tag{2}$$

The total gain cost considering all the VMs can be expressed as

$$G = \sum_{j=1}^{m} \sum_{i=1}^{n} E(T_i) \times A(T_i, VM_j) \tag{3}$$

3.1.3. Penalty cost

Penalty cost refers to the cost that the CSP incurs and pays to the user when it fails to perform the tasks. The tasks which are not allocated to any VMs can be defined by the Boolean variable as

$$B(T_i) = \begin{cases} 1 & \text{if } T_i \text{ is not allocated to any VM} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The overall penalty cost for all the VMs can be expressed in a manner similar to the gain cost as

$$P = \sum_{j=1}^{m} \sum_{i=1}^{n} S(T_i) \times B(T_i) \tag{5}$$

Where, S() is the size of task. The overall benefit or loss can be calculated from the Eq.(3) and Eq.(5) as

$$Profit = G - P \tag{6}$$

When profit is negative, it represents a loss. In order to maximize the profit, the average utilization of VMs should be maximum.

3.1.4. Average utilization of VMs

The utilization of a VM for a specific period of time t can be defined as the ratio of its execution time to the total time t. Thus,

$$\gamma(VM_j) = \frac{E_T}{t} \tag{7}$$

$\gamma(VM_j)$=0 represents VM_j is idle for the time period t.

3.1.5. Makespan

The makespan of a set of m number of allocated tasks $T_1, T_2, \cdots T_m$ executed by different VMs is the maximum execution time of these VMs.

$$M = \max\{E_T(VM_1), E_T(VM_2), \ldots, E_T(VM_m)\} \tag{8}$$

Where E_T is the execution time.

If $n > m$, the remaining tasks will be allocated to the VMs and hence the overall makespan becomes

$$\begin{aligned} M =& \max\{E_T(VM_1), E_T(VM_2), \ldots, E_T(VM_m)\} \\ &+ \max\{E_T(VM_1), E_T(VM_2), \ldots, E_T(VM_{i-m})\} \\ &\ldots \\ &+ \max\{E_T(VM_1), E_T(VM_2), \ldots, E_T(VM_{n-m})\} \end{aligned} \tag{9}$$

The profit can be maximized by maximizing the utilization of VMs or minimizing the makespan. Since VMs have distinct configurations, the optimal assignment of tasks to these VMs leads to the highest possible profit for the CSP.

3.2. Randomized PSO algorithm for task scheduling

The task scheduling problem involves the allocation of different tasks to available VMs to satisfy some constraints like maximizing the profit or minimizing the overall makespan of the task execution. In this context, the conventional PSO algorithm might not be effective because the fitness function or the objective function is not a mathematical function but rather an allocation problem. Further, also the governing equations typically used in PSO to update the velocity and position may not be applicable. Hence, in this work, a modified version of PSO is presented in which each particle is assigned a random allocation of tasks to the VMs. The velocity of each particle is determined by replacing a task from the list of tasks allocated earlier. The position of the particle is updated by assigning it to its current velocity.

3.2.1. Position

Each particle is assigned a random allocation of tasks to the different VMs. The position is represented by an array (1xm). It is initialized as

$$position = [unique(random(0, n-1), m-1)] \tag{10}$$

Where, unique() generates m random numbers each time from 0 to n-1

3.2.2. Velocity and the fitness function

The velocity of each particle is defined by replacing a single task from the earlier allocated tasks.
The fitness function to evaluate each particle is presented in Algorithm 1. In this algorithm, MIPS() is the MIPS of VMs, and max() finds the maximum elements from an array.
Each step of the randomized PSO algorithm is presented in Algorithm 2. In this algorithm, FUTILITY is a number greater than any number generated by the random number generator.

3.2.3. Time complexity and convergence of the proposed algorithm

The infinite while loop stops its execution when the condition gp=gBest is satisfied i.e. the current computed gBest and previously computed gBest do not change their values. Let ι be the number of iterations for which the infinite while loop will execute. The inner for loop will execute up to n_p times. Hence, the worst running time of the proposed algorithm is O($\iota \times n_p$).

Algorithm 1 fitness(position)

```
for i = 0 to m-1 do
    E[i]=T[position[i]]/MIPS(VM[i])
end for
ms=max(E[i])
return ms
```

Algorithm 2 Randomized_PSO(T,VM,n_p)

```
gBest=FUTILITY
gp=0
for each p do
    p.position=[unique(random(0,n-1), m-1)]
    p.velocity=[[0] for i in range(m)]
end for
for each particle p, $1 \leq i \leq n_p$ do
    p[i].pBest = Fitness(p)
end for
Find $gBest = \arg\min_{1 \leq i \leq n_p}(p[i].pBest)$
while !termination do
    for i = 1 to $n_p$ do
        Alloc=p[i].position
        p[i].velocity= [replace(T[x],T[y]) in Alloc for T[x] $\in$ Alloc and T[y]$\notin$Alloc
        p[i].position=p[i].velocity
        if $Fitness(p[i]) < p[i].PBest$ then
            p[i].PBest=Fitness(p[i])
        end if
        if $p[i].PBest < GBest$ then
            GBest= p[i].PBest
        end if
    end for
    if gp==gBest then
        break
    end if
    gp=gBest
end while
```

4. Results and discussion

The proposed algorithm is simulated in Cloudsim. The different VMs used in this work are in line with the VMs of Cloudsim. Twenty number of particles are taken while simulating the randomized PSO algorithm.

4.1. Convergence of the randomized PSO

To demonstrate the convergence of the randomized PSO, various combinations of workload and VMs are employed. The workload contains the tasks generated randomly with sizes ranging from 2 million instructions to 3 million instructions. The results obtained are presented in Table 1. This table entails the randomized PSO to converge each time within a satisfactory number of iterations.

Table 1
Convergence of randomized PSO

Sl. no.	# of tasks	# of VM	Epoch #	Makespan
1	100	32	47	3599.9
2	200	32“	10	3800
3	500	32	201	3599.9
4	100	64	11	5999.9
5	200	64	20	5800
6	500	64	91	5600
7	1000	32	100	5400
8	1000	64	173	5600
9	1000	128	333	6000
10	2000	128	568	5800

Table 2
Comparison of the proposed algorithm

Algorithm	Task id	Task size	VM	MIPS(VM)	E_T	MS
FCFS	1	200000	1	250	800	
	2	300000	2	400	750	1600
	3	100000	3	500	200	
	4	400000	4	250	1600	
	3	100000	1	250	400	
SJF	1	200000	2	400	500	1200
	5	200000	3	500	400	
	2	300000	4	250	1200	
	3	100000	1	250	400	
Proposed	1	200000	4	250	800	800
	5	200000	2	400	500	
	2	300000	3	500	600	

4.2. Comparison

The proposed scheduling algorithm is compared against some standard algorithms like shortest job first (SJF) and first come first serve (FCFS). In order to effectively compare these algorithms, we consider a scenario where five tasks are scheduled across four VMs. The results are presented in Table 2. This table shows a substantial reduction of the makespan (MS) while executing the four tasks.

The randomized PSO is compared against the genetic algorithm in terms of the makespan. While using GA, each chromosome represents one solution to the problem, i.e., allocation of tasks to the VMs. Two point crossover method is used for the generation of the offspring. The fitness function as presented earlier is used to evaluate each chromosome. The size of the population is 100. The comparison is made for different workloads such as 100, 200, 300, 400, and 500 tasks respectively. The results of these comparisons are depicted in Figure 1. Randomized PSO outperforms GA for each workload in terms of minimizing the makespan.

4.3. SLA based scheduling

The service level agreement(SLA) can be defined in terms of three levels (0, 1, and 2)[6]. These levels are defined in Table 3. To ensure SLA, a gain cost of 3 per unit processing time (as per Cloudsim) and a penalty cost of 1 per million instructions are considered in this work. While the number of virtual machines (VMs) may vary, the workload is represented by 128 randomly generated jobs, each with a size ranging from 2 to 3 million instructions. For level 1, the CSP provides the maximum available VMs to execute the tasks. The least number of VMs are made available to the user for level 1. Level

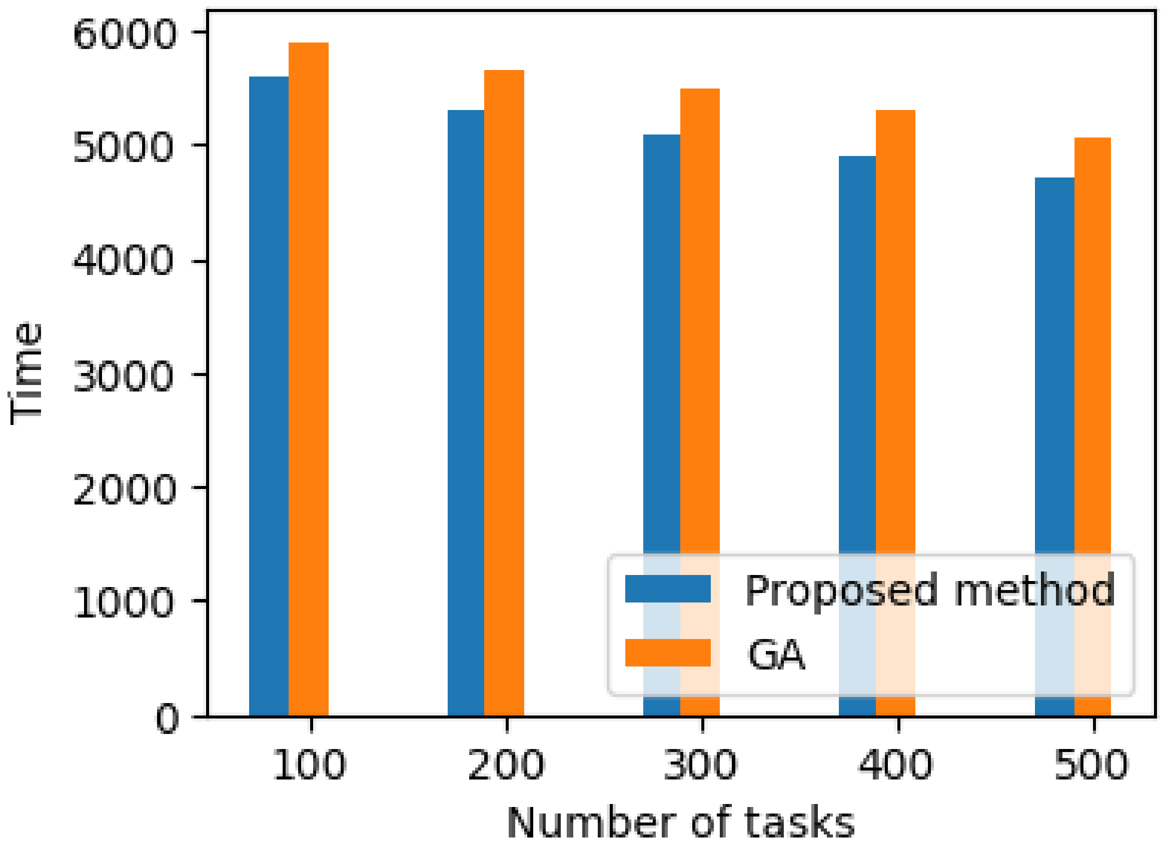

Figure 1: Comparison of randomized PSO and GA over different workloads

Table 3
Levels of SLA

Level	Performance in %	Budget in%
1	100	0
2	0	100
3	x	y

3 provides a computing environment of a mixed type of performance-based and budget-based. For level 1, 128 VMs are allocated to execute the tasks and the net profit is nearly 15000 (Figure 2). For 32 VMs, the randomized PSO algorithm is executed four times, and the total makespan is nearly 5000 time units (profit of 15000). If the budget is taken into consideration, the profit is calculated accordingly. For e.g. suppose the user requires the budget to be 6000, then only 2000 processing time is allocated, and thereby only 32 tasks are executed (Figure 2). To study the SLA for level 3, the tasks are mixed types i.e. x% tasks are performance-based while y% are budget-based. The figure depicts the profit of CSP for executing 128 tasks of 30% performance-based and 70% budget-based with the help of varying VMs (Figure 2).

The utilization rate of VMs is shown in Figure 3. The comparison of profit, penalty cost and gain cost for level 1, 2 and 3 are shown in Figure 4 for processing of 128 number of tasks. This figure entails the level 1 to be more profit making than other two levels. However, if the user needs processing of a huge number of tasks, the CSP may not allocate the VMs sufficiently enough to avoid the penalty cost. Thus, CSP may provide level 3 service even though some penalty cost may be incurred. In the level 3 service, users may be debarred from processing most of its tasks if a deadline is considered or they have to wait

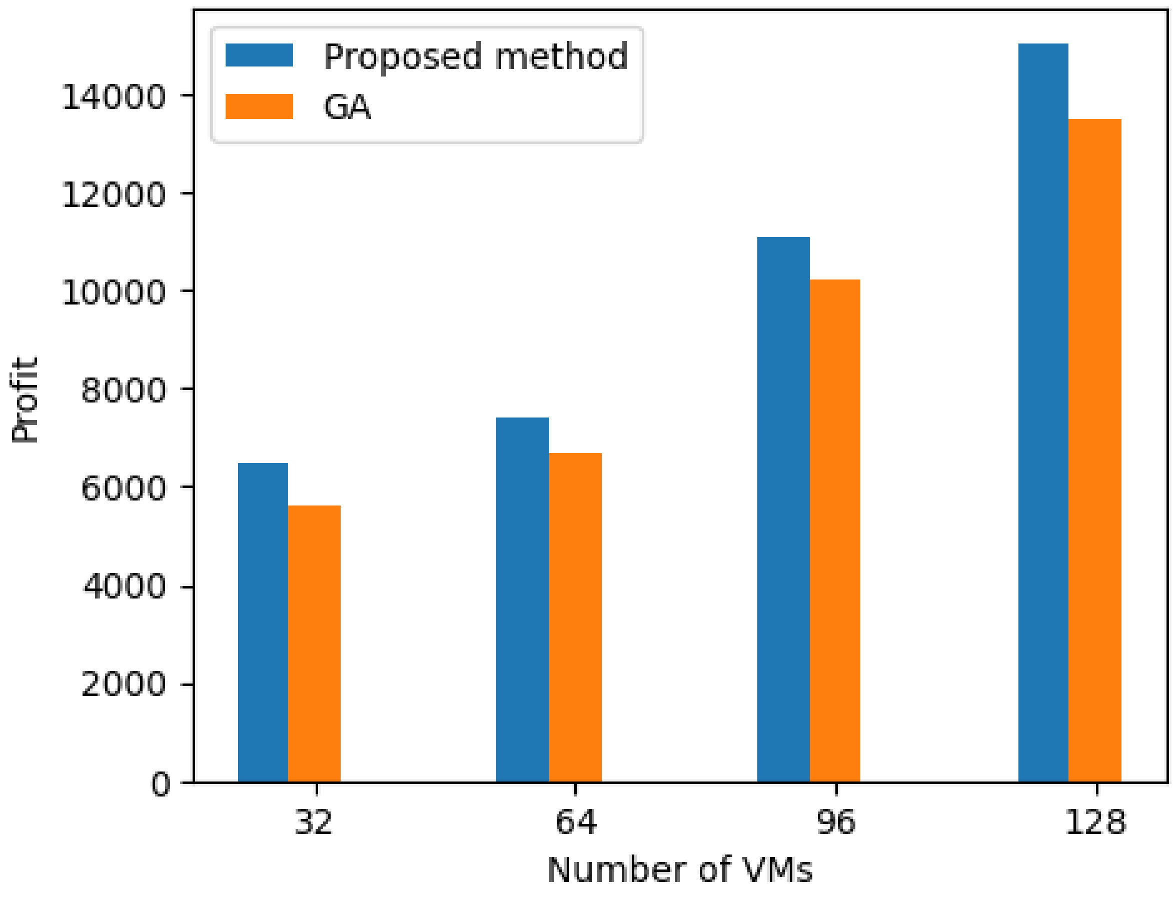

Figure 2: SLA based scheduling for maximum profit

for a longer time when no deadline is fixed. The penalty cost is nominal to gain cost for this case. As compared to level 2, level 3 provides more profit to CSP while deploying a feasible number of VMs.

5. Conclusion

The necessity for efficient task scheduling in SLA-based cloud computing environments arises from the increasing complexity and scale of cloud infrastructures, where optimal resource allocation is paramount to meeting performance objectives and ensuring customer satisfaction. Task scheduling involves the allocation of diverse tasks to available VMs while adhering to various constraints, such as maximizing profit or minimizing the overall makespan of task execution. Taking into account the significance of this challenge, we have developed a randomized PSO algorithm in this paper. The rationale behind employing the developed algorithm lies in its ability to effectively explore the solution space and discover optimal task-to-VM assignments while satisfying critical constraints and objectives defined by SLAs. To evaluate the effectiveness of the proposed scheduling algorithm, a comparative analysis was performed against existing scheduling approaches such as SJF and FCFS. The comparison aims to assess the performance, efficiency, and scalability of the randomized PSO algorithm in terms of profit maximization and makespan minimization. The outcomes of this comparison provide valuable insights into the efficacy of the proposed algorithm in addressing the complexities inherent in SLA-based cloud computing environments.

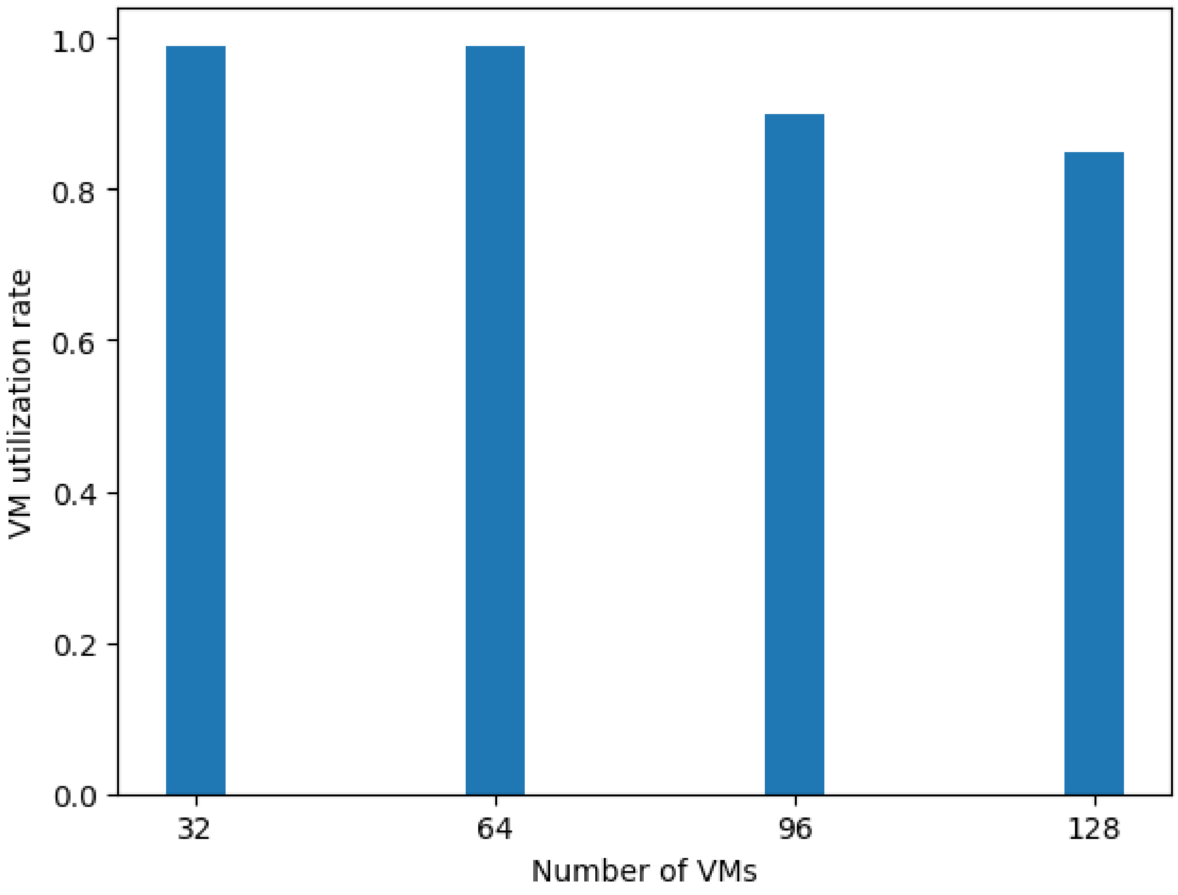

Figure 3: Utilization rate of VMs

References

[1] S. Sindhu, S. Mukherjee, Efficient task scheduling algorithms for cloud computing environment, in: International conference on high performance architecture and grid computing, Springer, 2011, pp. 79–83.

[2] S. Son, G. Jung, S. C. Jun, An sla-based cloud computing that facilitates resource allocation in the distributed data centers of a cloud provider, The Journal of Supercomputing 64 (2013) 606–637.

[3] A. Hussain, M. Aleem, M. A. Iqbal, M. A. Islam, Sla-ralba: cost-efficient and resource-aware load balancing algorithm for cloud computing, The Journal of Supercomputing 75 (2019) 6777–6803.

[4] M. Lavanya, B. Shanthi, S. Saravanan, Multi objective task scheduling algorithm based on sla and processing time suitable for cloud environment, Computer Communications 151 (2020) 183–195.

[5] M. Sharma, R. Garg, An artificial neural network based approach for energy efficient task scheduling in cloud data centers, Sustainable Computing: Informatics and Systems 26 (2020) 100373.

[6] A. Y. Hamed, M. H. Alkinani, Task scheduling optimization in cloud computing based on genetic algorithms, Computers, Materials & Continua 69 (2021) 3289–3301.

[7] M. Hussain, L.-F. Wei, A. Lakhan, S. Wali, S. Ali, A. Hussain, Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing, Sustainable Computing: Informatics and Systems 30 (2021) 100517.

[8] R. Rajavel, S. K. Ravichandran, P. Nagappan, S. Venu, Cost-enabled qos aware task scheduling in the cloud management system, Journal of Intelligent & Fuzzy Systems 41 (2021) 5607–5615.

[9] C. Li, J. Liu, B. Lu, Y. Luo, Cost-aware automatic scaling and workload-aware replica management for edge-cloud environment, Journal of Network and Computer Applications 180 (2021) 103017.

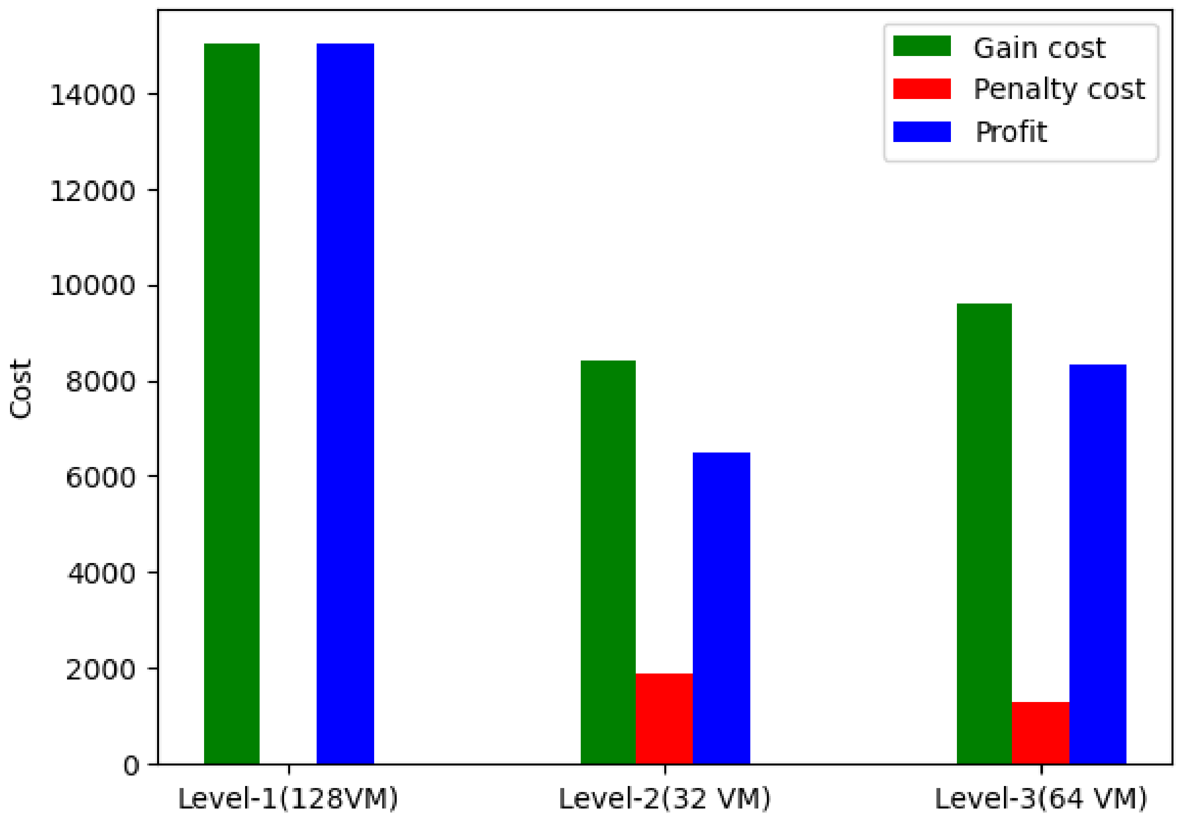

Figure 4: Comparison of profit for different SLA levels

[10] M. S. Ajmal, Z. Iqbal, F. Z. Khan, M. Bilal, R. M. Mehmood, Cost-based energy efficient scheduling technique for dynamic voltage and frequency scaling system in cloud computing, Sustainable Energy Technologies and Assessments 45 (2021) 101210.

[11] X. Tang, Y. Liu, Z. Zeng, B. Veeravalli, Service cost effective and reliability aware job scheduling algorithm on cloud computing systems, IEEE Transactions on Cloud Computing (2021).

[12] Z. Zhao, X. Shi, M. Shang, Performance and cost-aware task scheduling via deep reinforcement learning in cloud environment, in: International Conference on Service-Oriented Computing, Springer, 2022, pp. 600–615.

[13] L. Cheng, A. Kalapgar, A. Jain, Y. Wang, Y. Qin, Y. Li, C. Liu, Cost-aware real-time job scheduling for hybrid cloud using deep reinforcement learning, Neural Computing and Applications 34 (2022) 18579–18593.

[14] S. M. Hussain, G. R. Begh, Hybrid heuristic algorithm for cost-efficient qos aware task scheduling in fog–cloud environment, Journal of Computational Science 64 (2022) 101828.

[15] F. Cheng, Y. Huang, B. Tanpure, P. Sawalani, L. Cheng, C. Liu, Cost-aware job scheduling for cloud instances using deep reinforcement learning, Cluster Computing (2022) 1–13.

[16] M. S. Kumar, G. R. Karri, Agwo: Cost aware task scheduling in cloud fog environment using hybrid metaheuristic algorithm, Int. J. Exp. Res. Rev 33 (2023) 41–56.

[17] S. Lipsa, R. K. Dash, N. Ivković, K. Cengiz, Task scheduling in cloud computing: A priority-based heuristic approach, IEEE Access 11 (2023) 27111–27126.

[18] R. Ghafari, N. Mansouri, Cost-aware and energy-efficient task scheduling based on grey wolf optimizer, J. Mahani Math. Res 12 (2022) 257–288.

[19] L. Cheng, Y. Wang, F. Cheng, C. Liu, Z. Zhao, Y. Wang, A deep reinforcement learning-based preemptive approach for cost-aware cloud job scheduling, IEEE Transactions on Sustainable Computing (2023).

[20] S. Mangalampalli, S. K. Swain, G. R. Karri, S. Mishra, et al., Sla aware task-scheduling algorithm in cloud computing using whale optimization algorithm, Scientific Programming 2023 (2023).

[21] S. Rac, M. Brorsson, Cost-aware service placement and scheduling in the edge-cloud continuum, ACM Transactions on Architecture and Code Optimization (2024).