ISSN: 0067-2904

A Deadline-Budget Constrained Task Admission Control for a Software-as-a-Service Provider in Cloud Computing

Richa Jain, Neelam Sharma
Department of Computer Science, Banasthali Vidyapith, Niwai, Rajasthan, India

Received: 21/3/2022 Accepted: 10/11/2022 Published: 30/10/2023

Abstract
Cloud computing is a pay-as-you-go model that provides users with on-demand access to services or computing resources. It is a challenging issue to maximize the service provider's profit and, on the other hand, meet the Quality of Service (QoS) requirements of users. Therefore, this paper proposes an admission control heuristic (ACH) approach that selects or rejects the requests based on budget, deadline, and penalty cost, i.e., those given by the user. Then a service level agreement (SLA) is created for each selected request. The proposed work uses Particle Swarm Optimization (PSO) and the Salp Swarm Algorithm (SSA) to schedule the selected requests under budget and deadline constraints. Performances of PSO and SSA with and without ACH are evaluated and compared using CloudSim. The simulation results prove that admission control maximizes profit by minimizing the number of task rejections and SLA violations. It also improves resource utilization while balancing makespan.

Keywords: Cloud Computing, Admission Control, Salp Swarm Algorithm; Task Scheduling, Service Layer Agreement.

1. Introduction

In cloud computing, the Software as a Service (SaaS) provider rents shareable computing resources from an Infrastructure as a Service (IaaS) provider and provides these resources to users' applications. Typically, SaaS providers want to increase their profit while users are willing to complete their applications under deadline and budget constraints. In a real scenario, a service level agreement (SLA) is signed between the user and SaaS provider that includes the user's quality of service (QoS) requirements. However, SLA also incorporates the penalty cost, which is given to the user if SLA is violated [1]. As a result, it decreases the SaaS provider's profit. Furthermore, as the number of SLA violations (SLAV) increases, it also reduces service reliability. Therefore, the main goal of SaaS providers is to execute users' applications while satisfying deadline and budget constraints and maximizing their profit. To achieve this, the main concern in this paper is to reduce the rate of SLAV. If a user submits an infeasible task with a low budget or a short deadline, it affects already accepted and upcoming tasks. So, there is a need for an admission control mechanism that identifies whether a request should be accepted or rejected to reduce resource overload.

*Email: jainricha86@gmail.com

2. Literature Review

Many studies propose an admission control model. Some of them base their decision on the current load [2, 3]. Leontiou et al. [4] proposed an adaptive admission control for the web application. Their objective is to prevent overloading. They employed a queuing model with an adaptive feedback control system that adapted the admitted load to balance for changes in the system's capacity. He et al. [5] developed an admission control model for aggregate flow. They admitted or rejected tasks based on the bandwidth of the aggregate flow. It is computed using network calculus. They improved resource allocation and optimized QoS parameters. Konstanteli et al. [6] proposed a probabilistic admission control that was modeled on GAMS. The proposed work focuses on the optimum allocation of services on virtualized machines and reduces physical resources. Carvalho et al. [7] developed an admission control model and capacity planning method for IaaS providers. It decreases total infrastructure costs and optimizes service level objectives with good accuracy. A lot of research exists that considers the user's QoS constraints like deadline and budget to decide whether a task should be accepted or not. Reig et al. [8] incorporated machine learning techniques to develop a self-adjusting predictor. It predicts the resource requirement by using the previous execution results. The decision to admit is taken based on deadline constraints. A profit-oriented admission control framework (ActiveSLA) is proposed in [9]. First, it calculates the probability of finishing execution before the deadline. Then, it selects/rejects the new query based on that probability. This decision is taken with profit optimization in mind. A deadline-aware admission control mechanism (PYTHIA) is proposed in [10]. PYTHIA admits the tasks if their deadlines can be met with the estimated resources; otherwise, it rejects the tasks. Yuan et al. [11] proposed a revenue-based admission control method that admits/rejects requests based on revenue, priority, and response time. Further, they proposed cost-aware scheduling to reduce costs and enhance the throughput of CDC providers. The literature [12, 13] included both user QoS constraints and workflow constraints. Hoang et al. [14] proposed an admission control algorithm considering both the deadline and budget. But they did not consider penalty costs, which is a significant parameter when calculating total profit. This paper proposes an admission control and scheduling mechanism that rejects the task if it cannot be completed within the deadline [if the deadline is hard] and budget or if the SaaS provider is not going to earn any profit [if the deadline is soft]. Khojasteh et al. [15] presented two algorithms for task admission control by incorporating a filtering coefficient and a full-rate task acceptance threshold value. In 1st algorithm, they calculated the estimated average utilization. Then, they reject the task if it is greater than the utilization threshold, whereas in the 2nd algorithm, the incoming task is selected or rejected based on current utilization. Malawski et al. [16] proposed a dynamic provisioning and dynamic scheduling (DPDS) algorithm for resource provisioning and scheduling. Furthermore, they expand DPDS by including an admission control procedure that admits a new workflow based on the remaining budget and deadline while minimizing workflow cost. Leontiou et al. [17] proposed an autonomous mechanism for admission control to ensure performance stability. They used the Kalman filter for incoming load prediction. Yuan et al. [11] proposed a revenue-based admission control algorithm that admits a new request based on its revenue, priority, and response time. They selected higher-priority requests first to maximize revenue. Zheng et al. [13] proposed a BDC (Budget Deadline Constraint) plan to accept or reject workflow requests. They proposed a heuristic called Budget-constrained Heterogeneous Earliest Finish Time (BHEFT) based on the well-known list scheduling heuristic HEFT. However, total execution costs are not addressed in this paper. Hoang et al. [14] proposed an admission control and task scheduling algorithm based on the meta-heuristic algorithms ACO and PSO. They focused on finding the lowest-cost VMs and improving the SAAS provider's profit. Wu et al. [18] proposed an admission control and scheduling algorithm to improve the profitability of SAAS service providers. They also analyzed the impact of variations in QoS parameters on performance. The decision to admit a task is taken based on the deadline, budget, and penalty rate. They did not focus on other user-driven QoS constraints such as makespan, reliability, security, etc. Choudhary et al. [19] proposed an access control (PbTAC) model based on task priority. This model secures the information by applying rule policies and scheduling the tasks. It also optimizes storage and computation costs. Huang et al. [20] proposed an algorithm (ACCRA) for admission control and resource allocation in mobile edge computing. The objective is to enhance the system's utility. First, they divide the problem into three subproblems, and then ACCRA finds optimal solutions for these subproblems. Sathya et al. [21] proposed a framework for EMSA to preserve the privacy of the stored data in the eHealth cloud-assisted environment. The EMSA algorithm is a hybrid of the Successive Approximation Iterative Proximate algorithm and the Euclidean L3P Distance algorithm, performing role-based key generation.

3. Problem Definition

3.1 System Architecture

Figure 1 represents the system architecture of admission control and scheduling in cloud computing. It consists of three main components: cloud users, the SaaS layer, and the cloud environment.

- Cloud User: A cloud user submits task requests to the SaaS provider with some QoS constraints, i.e., deadline, deadline type, budget, and penalty ratio. The deadline type can be hard or soft. A soft deadline indicates that the user can wait after a missed deadline. A compensation amount is given to the user from the SaaS provider for this delay. In contrast, a hard deadline indicates that the task must be completed before its deadline.
- SaaS Layer:- User requests are received at this layer, and then the SaaS provider first checks whether this request is feasible or not. A request/task is feasible if it can be completed within the given deadline and budget. After rejecting infeasible requests, a SLA is created between the user and the SaaS provider. Then all selected tasks are scheduled using a task scheduling algorithm. SLA consists of four constraints in this work, i.e., deadline, budget, deadline type, and penalty.
- Cloud environment:- An IaaS provider offers a virtually infinite pool of resources. These computing resources are referred to as "virtual machines" (VMs) and are charged on a per-use basis. A SaaS provider leases software as a service to users on demand by renting these resources from a specific IaaS provider.

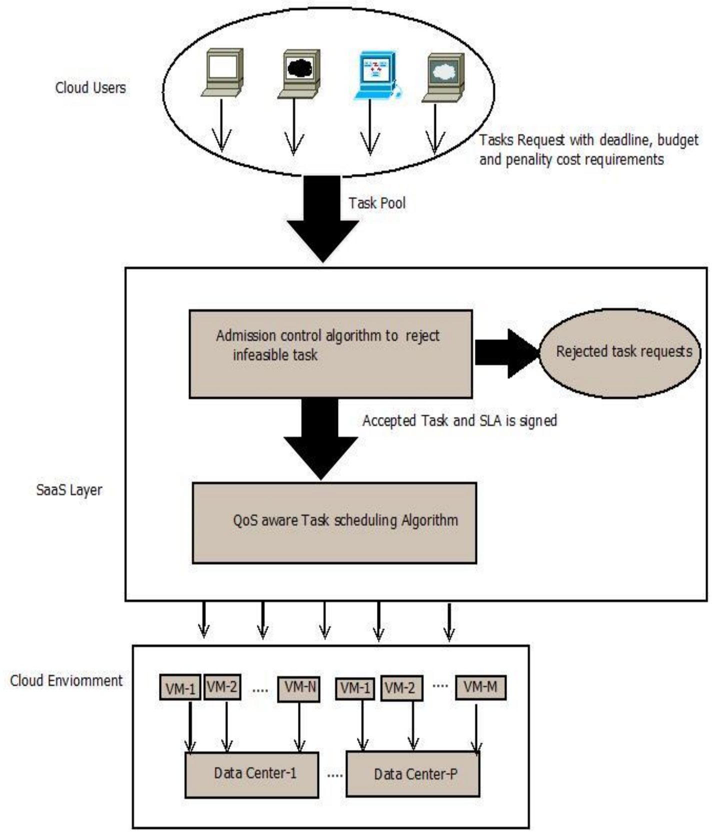
Cloud Users
Tasks Request with deadline, budget and penality cost requirements
Task Pool
SaaS Layer
Admission control algorithm to reject infeasible task
Rejected task requests
Accepted Task and SLA is signed
QoS aware Task scheduling Algorithm
Cloud Enviomment
VM-1
VM-2
....
VM-N
VM-1
VM-2
....
VM-M
Data Center-1
....
Data Center-P

Figure 1: System Architecture

3.2 Problem Formulation

3.2.1. User's Task

This work uses the CloudSim3.0 framework to analyze the performance of a proposed heuristic and uses workload traces from Planet Lab to make this simulation applicable. These data are part of the CoMon project, which can be accessed at [https://github.com/beloglazov/planetlab-workload-traces]. A user sent a task with some QoS constraints, i.e., deadline, deadline type, budget, and penalty. Based on these constraints, the admission control mechanism decides whether to accept or reject the task. If a request is accepted, a SLA is created and signed by both the user and the SAAS provider.

3.2.2. Resources

This project includes a single data center with virtual machines that are charged based on their usage. This resource model is similar to the one offered by Amazon EC2 [22]. Here, we consider m instances of six types of VM, which are heterogeneous and represented as VM=,{Vm-1.,,Vm-2……..,Vm-m.}. Price, bandwidth, RAM, and number of CPU cores of different VM types are shown in Table 1. These are adopted from the General Purpose instance group.

Table 10: Instance type based on Amazon EC2

VM TYPE	VCPU	Bandwidth (Mbps)	RAM(Gib)	Price ($)
a1.medium	1	10000	2	0.0255
a1.large	2	10000	4	.0510
m4.large	2	450	8	.19
m4.xlarge	4	750	16	.38
t2.small	1	600	2	.032
t2.medium	2	650	4	.0644

3.2.3 Mathematical Model

This section introduces the mathematical equations that are used in this work. A set of n tasks denoted by $T = \{t_1, t_{2,\dots\dots,} t_n\}$. dl_i, dt_i, bt_i and p_i represent deadline, deadline type, budget and penalty cost of i^{th} task. Deadline is calculated as given in [23]. In this work, the deadline factor (β) is taken as 5 (average value). It is assumed that 20% of the total number of tasks have a hard deadline. In this work, budget bt_i is calculated as follows:

$$bt_i = \frac{minT_i + maxT_i}{2} \quad , \; i \in n \tag{1}$$

Here, minTi and maxTi are the minimum and maximum execution times of a task t_i.
Completion Time of i^{th} task on j^{th} Vm is represented as $CompT_{ij}$. It can be calculated as:

$$CompT_{ij} = StartT_i + RunT_{ij} \tag{2}$$

Where $StartT_i$ is the start time when task i has started its execution and $RunT_{ij}$ is the total execution time of task i on j^{th} Vm.
$cost_{ij}$ is the total execution cost when i^{th} task runs on j^{th} Vm. It is calculated as:

$$\text{cost}_{\text{ij}} = RunT_{ij} * \text{price}_{\text{j}} \tag{3}$$

Where price_j represents the unit price of the j^{th} Vm.
So, Total Profit of SaaS provider can be calculated as:

$$Tprofit = \sum_{i=0}^{n} bt_i - \sum_{i=0}^{n} cost_{ij} - \sum_{i=0}^{n} p_i \tag{4}$$

If SLA is violated then some penalty p_i is given by provider to the user. It decreases the profit of the provider. This study aims to reduce the SLA violation rate (SlaVR) to increase service provider profit. SlaVR is calculated as:

$$\text{SlaVR} = \frac{Total\ No.of\ SLA\ \text{violation}}{Total\ No.of\ Task} * 100 \tag{5}$$

A task t_i can be scheduled on j^{th} Vm only when its deadline (dl_i) is less than or equal to $(CompT_{ij})$ and its budget (bt_i) is greater than execution cost $(cost_{ij})$.
The objective of this work is to maximize the total profit of SaaS providers. Then, Deadline-Budget Constrained Task Admission Control problem can be formulated as follows:
$\boldsymbol{Maximize\ Tprofit}$
subject to

$$dl_i \le CompT_{ij}\ (if\ hard\ deadline) \tag{6}$$
$$bt_i > cost_{ij} \tag{7}$$
$$i \in \{1,2,\dots\dots n\}(set\ of\ n\ tasks)$$
$$\text{j} \in \{1,2,\dots\dots m\}(set\ of\ m\ Vms)$$

Constraint (6) assures that the deadline of the admitted task must be less than its completion time. Constraint (7) ensures that the budget of the admitted task must be less than its execution cost. After the admission control process, tasks are scheduled in such a way that profit is maximized while satisfying deadline and budget constraints.

4. The Proposed Algorithm

Algorithm 1 shows the pseudo-code of the proposed admission control heuristic. We'll look at two resource queues here.1) A time-oriented queue (consisting of fast VMS) 2) Budget-oriented queue (consisting of cheap VMs) (line 1). Tasks are arranged in ascending order to schedule the task with the lower deadline first (line 2). Initially, lines 3–6 schedule m tasks. First, a VM from ToQ is taken, and the check_constraint function (Algorithm 2) checks whether tasks can be scheduled on that VM under deadline and budget constraints. The check_constraint function works as follows:

1) If both constraints are satisfied, then schedule the task.
2) If budget is violated, take a Vm from BoQ; If it satisfies both constraints, then schedule; otherwise, reject the task.
3) If a deadline is violated and it is a hard deadline, reject the task; otherwise, schedule the task and calculate the delay.

Lines (7-10) schedule the remaining tasks. Line 8 finds that the Vm has a minimum load, and then line 9 calls the check_constraint functions to check constraints. After rejecting infeasible requests, line 11 creates an SLA for accepted task requests. Finally, line 12 executes these selected tasks using efficient task scheduling algorithms. In this work, we are using SSA and PSO.

Algorithm 1: Pseudo code of proposed Admission Control Heuristic (ACH) and scheduling.

```
    Input: User's tasks (Number of tasks is m)
    Output: Tasks are scheduled while optimizing QoS constraints.
    Steps:
1.  Take two resource queues i.e. time oriented (ToQ) and budget oriented(BoQ)
2.  Sort all tasks the non-decreasing order of deadline.
3.  for each tasks $t_i$(i=1 to m)
4.    take a $Vm_i$ from ToQ
5.    Call Check_Constraints(i,j)
6.  end for
7.  for each tasks $t_i$ (i=m+1 to n)
8.    find vm having minimum execution time
9.    Call Check_Constraints(i,vm)
10. end for
11. Create SLA for selected tasks
12. Do task scheduling for optimizing QoS constraints.
```

Algorithm 2: Pseudo code of checking constraints for admission control

```
Check_Constraints(i,j)
    Input:   Task i and vm
    Output: Task is selected /rejected
Steps:

  1.  if $dl_i$<=$compT_i$ and $bt_i$ >$cost_{ij}$
  2.     schedule $t_i$
  3.  else if($dl_i$>$compT_i$)
             if($dl_i$ type is HARD)
                then reject $t_i$
             else schedule $t_i$ and calculate delay
  4.   if($bt_i$ <cost)
          take $Vm_k$ from ToB
          if $dl_i$<=$compT_k$ and $bt_i$ >$cost_{ik}$
             then schedule $t_i$
             k++
          else reject $t_i$
```

5. Result and discussion

To evaluate the performance of the proposed heuristic (ACH), we use the CloudSim 3.0 framework [24]. Task scheduling is done using existing metaheuristics, i.e., PSO [25] and SSA [26]. To prove the effectiveness of the ACH, we compare the performance of PSO, SSA, APSO (PSO with admission control), and ASSA (SSA with admission control). All algorithms are run in the same simulation environment for a fair comparison. The population size is 50, and the maximum number of iterations is 1000 for all algorithms. Experiments are conducted for 1052 user tasks and varying VMS from 50 to 250. Each experiment is run five times independently. The population size is 50, and the maximum number of iterations is 1000 for all algorithms. Experiments are conducted for 1052 user tasks and varying VMS from 50 to 250. Each experiment is run five times independently. It can be observed from Figure 2 that if ACH is combined with SSA and PSO, then it increases profit. In the results, ASSA achieves a profit approximately 33% higher than SSA, whereas APSO achieves about 19% higher than SSA. ACH prevents the admission of infeasible tasks, which decreases the number of SLA violations and the number of task rejections. Figure 3 compares ASSA with SSA and APSO with PSO and shows that ASSA and APSO reduce SLA violations. Comparison results prove that the number of SLA violations is reduced in ASSA by 17% and in APSO by 30%. Figure 3 shows the comparison of ASSA with SSA and APSO with PSO in terms of task rejections. Simulation results prove that ASSA reduces the number of task rejections by 18% and APSO reduces it by 36%. For ASSA and APSO, rejected tasks are calculated by adding the rejected tasks in ACP and in the task scheduling process. Figure 5 illustrates that ASSA and APSO balance makespan also. Finally, resource utilization is compared for SSA and PSO with ASSA and APSO, respectively. Comparison results show that ACH effectively enhances resource utilization. It can be seen in the results that it increased by 8% in ASSA and by 8.5% in APSO.

Figure 3 and Figure 4 compare ASSA with SSA and APSO with PSO and show that ASSA and APSO reduce SLA violations and task rejections, respectively. For ASSA and APSO, rejected tasks are calculated by adding the rejected tasks in ACP and in the task scheduling process. Finally, Figure 6 proves that resource utilization is increased by adding ACH.

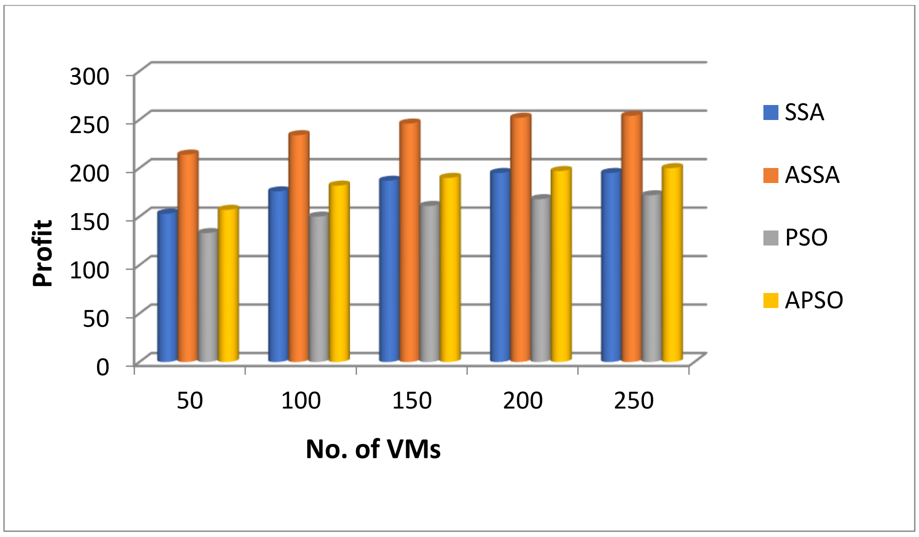

Figure 2: Comparison of Profit for SSA, ASSA, PSO and APSO.

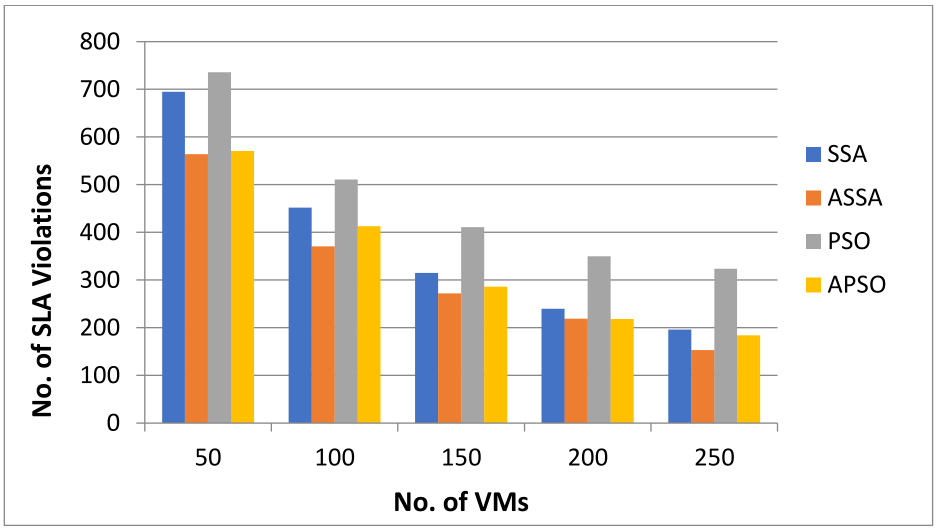

Figure 3: Comparison of Number of SLA Violations for SSA, ASSA, PSO and APSO

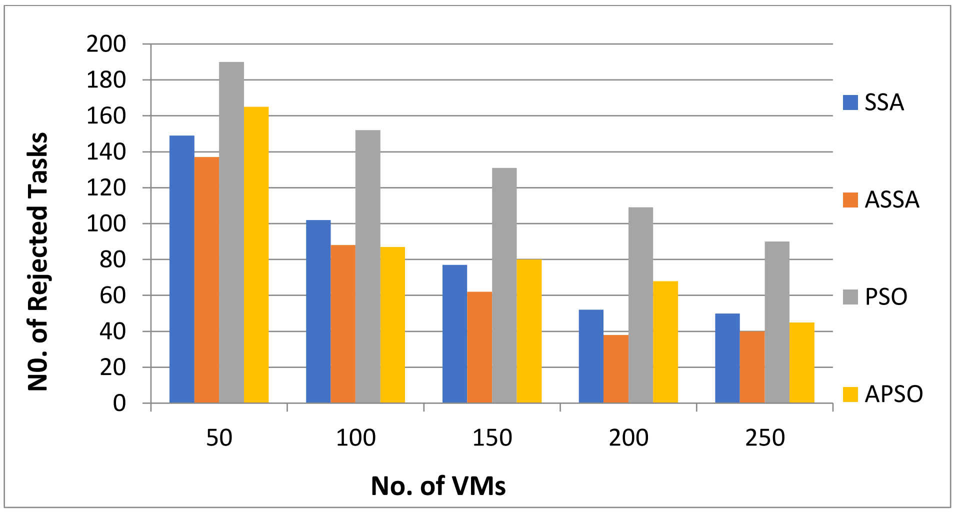

Figure 4: Comparison of Number of Task Rejections for SSA, ASSA, PSO and APSO

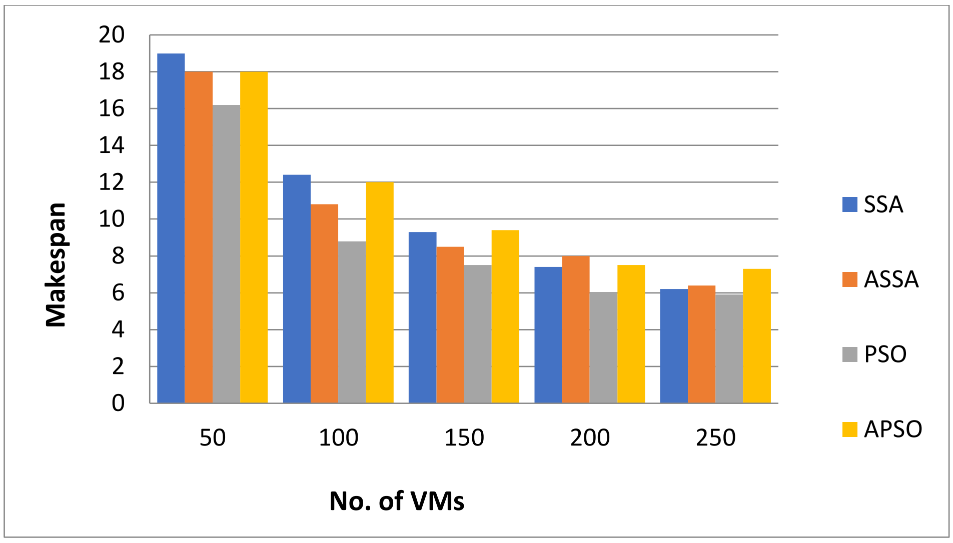

Figure 5: Comparison of Makespan for SSA, ASSA, PSO and APSO

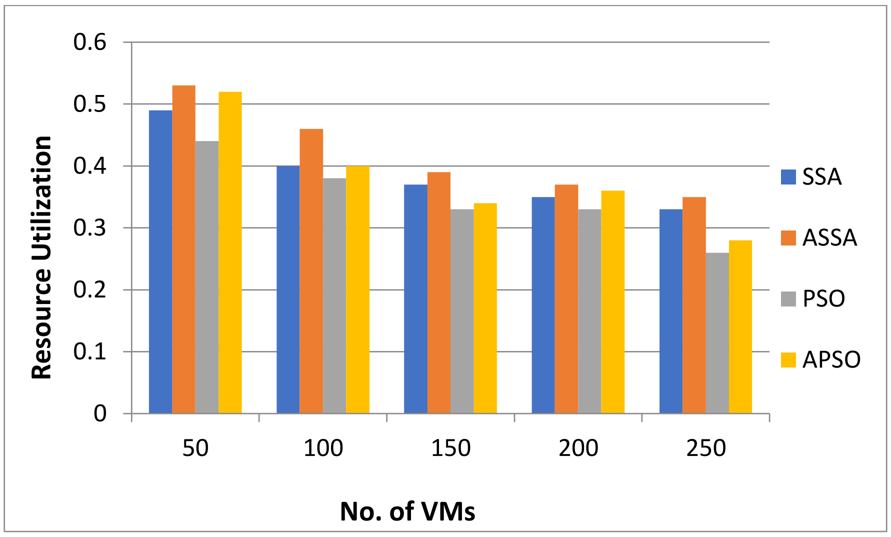

Figure 6: Comparison of Resource Utilization for SSA, ASSA, PSO and APSO

6. Conclusion and future work

This paper describes a budget deadline-based admission control heuristic (ACP) that is implemented at the SaaS layer. It guarantees to reduce SLA violations and task rejections by rejecting infeasible tasks before SLA establishment. The proposed work also incorporates penalty costs that affect the profit of the provider. Finally, a SLA is created for each selected task, and they are scheduled in such a way that budget and deadline constraints are met. According to simulation results, the proposed ACP and scheduling can significantly increase the profit of SaaS provider resource utilization while also satisfying the user by meeting SLA parameters and balancing time frames. In future work, we would extend the work by improving task scheduling that optimizes QoS parameters like throughput, load balancing, and energy consumption.

Conflict of interest

The authors declare that they have no conflicts of interest.

References

[1] Yeo, Chee Shin, and Rajkumar Buyya, "Service level agreement based allocation of cluster resources: Handling penalty to enhance utility," In *2005 IEEE International Conference on Cluster Computing*, pp. 1-10. IEEE, 2005.

[2] Cherkasova, Ludmila, and Peter Phaal, "Session-based admission control: A mechanism for peak load management of commercial web sites," *IEEE Transactions on computers* 51, no. 6 (2002): 669-685.

[3] Zhao, Yali, Rodrigo N. Calheiros, Athanasios V. Vasilakos, James Bailey, and Richard O. Sinnott, "Profit maximization and time minimization admission control and resource scheduling for cloud-based big data analytics-as-a-service platforms," In *International Conference on Web Services*, pp. 26-47. Springer, Cham, 2019.

[4] Leontiou, Nikolaos, Dimitrios Dechouniotis, and Spyros Denazis, "Adaptive admission control of distributed cloud services," In *2010 International Conference on Network and Service Management*, pp. 318-321. IEEE, 2010.

[5] He, Yunlong, Jun Huang, Qiang Duan, Zi Xiong, Juan Lv, and Yanbing Liu, "A novel admission control model in cloud computing," *arXiv preprint arXiv:1401.4716* (2014).

[6] Konstanteli, Kleopatra, Theodora Varvarigou, and Tommaso Cucinotta, "Probabilistic admission control for elastic cloud computing," In *2011 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, pp. 1-4. IEEE, 2011.

[7] Carvalho, Marcus, Francisco Brasileiro, Raquel Lopes, Giovanni Farias, Alessandro Fook, Joao Mafra, and Daniel Turull, "Multi-dimensional admission control and capacity planning for IaaS clouds with multiple service classes," In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pp. 160-169. IEEE, 2017.

[8] Reig Ventura, Gemma, Javier Alonso López, and Jordi Guitart Fernández, "Deadline constrained prediction of job resource requirements to manage high-level SLAs for SaaS cloud providers," (2010).

[9] Xiong, Pengcheng, Yun Chi, Shenghuo Zhu, Junichi Tatemura, Calton Pu, and Hakan HacigümüŞ, "ActiveSLA: a profit-oriented admission control framework for database-as-a-service providers," In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, pp. 1-14. 2011.

[10] Dimopoulos, Stratos, Chandra Krintz, and Rich Wolski, "Pythia: Admission control for multi-framework, deadline-driven, big data workloads," In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pp. 488-495, 2017.

[11] Yuan, Haitao, Jing Bi, Wei Tan, and Bo Hu Li, "CAWSAC: Cost-aware workload scheduling and admission control for distributed cloud data centers," *IEEE Transactions on Automation Science and Engineering* 13, no. 2, pp. 976-985, 2015.

[12] Shi, Jiyuan, Junzhou Luo, Fang Dong, Jinghui Zhang, and Junxue Zhang, "Elastic resource provisioning for scientific workflow scheduling in cloud under budget and deadline constraints," *Cluster Computing* 19, no. 1, pp. 167-182, 2016.

[13] Zheng, Wei, and Rizos Sakellariou, "Budget-deadline constrained workflow planning for admission control," *Journal of grid computing* 11, no. 4, pp. 633-651, 2013.

[14] Hoang, Ha Nguyen, Son Le Van, Han Nguyen Maue, and Cuong Phan Nhat Bien, "Admission control and scheduling algorithms based on ACO and PSO heuristic for optimizing cost in cloud computing," In *Recent Developments in Intelligent Information and Database Systems*, pp. 15-28. Springer, Cham, 2016.

[15] Khojasteh, Haleh, and Jelena Mišić, "Task admission control policy in cloud server pools based on task arrival dynamics," *Wireless Communications and Mobile Computing* 16, no. 11, pp.1363-1376, 2016.

[16] Malawski, Maciej, Gideon Juve, Ewa Deelman, and Jarek Nabrzyski, "Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds," *Future Generation Computer Systems* 48, pp. 1-18, 2015.

[17] Leontiou, Nikolaos, Dimitrios Dechouniotis, and Spyros Denazis, "Adaptive admission control of distributed cloud services," In *2010 International Conference on Network and Service Management*, pp. 318-321. IEEE, 2010.

[18] Wu, Linlin, Saurabh Kumar Garg, and Rajkumar Buyya, “SLA-based admission control for a Software-as-a-Service provider in Cloud computing environments,” *Journal of Computer and System Sciences* 78, no. 5, pp. 1280-1299, 2012.

[19] Choudhary, Sandeep, and Nanhay Singh, “Analysis of Security-Based Access Control Models for Cloud Computing,” International Journal of Cloud Applications and Computing (IJCAC) 12, 1, pp. 1-19, 2022.

[20] Huang, Jiwei, et al, “Dynamic admission control and resource allocation for mobile edge computing enabled small cell network,” *IEEE Transactions on Vehicular Technology* 71, 2, pp. 1964-1973, 2021.

[21] Sathya, A., and S. Raja, “Privacy preservation-based access control intelligence for cloud data storage in smart healthcare infrastructure,” *Wireless Personal Communications* 118, 4, pp. 3595-3614, 2021.

[22] Amazon EC2 Pricing. https://aws.amazon.com/ec2/pricing/.

[23] Jain, Richa, and Neelam Sharma, “A Deadline-Constrained Time-Cost-Effective Salp Swarm Algorithm for Resource Optimization in Cloud Computing,” *International Journal of Applied Metaheuristic Computing (IJAMC)* 13, no. 1, pp. 1-21, 2022.

[24] Calheiros, Rodrigo N., Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya, “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” *Software: Practice and experience* 41, no. 1, pp. 23-50, 2011.

[25] Eberhart, Russell, and James Kennedy, “A new optimizer using particle swarm theory,” *MHS'95. Proceedings of the sixth international symposium on micro machine and human science*. IEEE, 1995.

[26] Mirjalili, Seyedali, et al, “Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems,” *Advances in engineering software* 114, pp. 163-191, 2017.