Peer-to-Peer Networking and Applications
https://doi.org/10.1007/s12083-024-01800-0

End to end delay aware service function chain scheduling in network function virtualization enabled networks

Sudha Dubba[1] · Balaprakasa Rao Killi[1]

Received: 1 December 2023 / Accepted: 24 August 2024

Abstract
Network function virtualization is a key enabling technology for the customization of network services in next-generation networks to support diverse applications. Most enterprise and network services contain specific network functions that are stitched together in a predefined sequence to form a service function chain. The deployment and scheduling of a service function chain onto the substrate network play a vital role in deciding the efficiency of resource utilization and the performance of network management. For a delay-sensitive network service request traversing a service function chain, the end-to-end packet delay is a crucial parameter that indicates the deployment performance. Transmission, propagation, processing, edge queueing, and virtualization delays all impact the order in which virtual network functions execute. Service level agreement violations and incorrect schedules are produced when the controller does not take edge queueing and virtualization delays into account. In this work, we propose a service function chain scheduling problem for the optimization of the end-to-end delay while considering transmission, propagation, queueing, virtualization, and processing delays. Then, we propose a scheduling approach based on the earliest finish times of the physical machines to minimize the end-to-end delay of the service function chain. The performance of the proposed service function chain scheduling approach using the earliest finish time is evaluated in terms of end-to-end delay, service level agreement violation ratio, resource utilization, and acceptance ratio. We compare our proposed algorithm with four existing approaches from the literature. Simulation results show that our proposed approach outperforms existing approaches in terms of end-to-end delay, service level agreement violation ratio, resource utilization, and acceptance ratio.

Keywords Network function virtualization · Service function chain scheduling · End-to-end delay · Service level agreement violation ratio

1 Introduction

Traditional network functions such as firewall (FW), intrusion detection system (IDS), network address translator (NAT), gateway, virtual private network (VPN), traffic monitor (TM), etc. are executed on hardware middleboxes. Current traditional networks have specific location-dependent services, laborious design and testing time to carry network functions to the market. Hence, they are incapable of supporting the scalability and flexibility

requirements of communication services for accomplishing the growing demands and upcoming business opportunities. Network function virtualization (NFV) is an architectural concept that leverages IT virtualization to virtualize entire classes of network capabilities into building blocks that can be chained together. It is a disruptive and significant paradigm in network service provisioning. NFV provides flexibility and elasticity to the composition and deployment of network services by detaching network functional components from built-in hardware. NFV enables to run the virtualized form of these network functions, known as virtual network functions (VNFs), on commer-

Balaprakasa Rao Killi contributed equally to this work

This article is part of the Topical Collection: *1- Track on Networking and Applications*
Guest editor: Vojislav B. Misic

✉ Sudha Dubba
ds720078@student.nitw.ac.in

Balaprakasa Rao Killi
bsprao@nitw.ac.in

[1] Department of Computer Science and Engineering, National Institute of Technology, Warangal 506003, Telangana, India

cial off-the-shelf servers. Further, NFV can reduce capital expenditure (CAPEX) and operational expenditure (OPEX) in the network. Most enterprise and network services consist of specific network functions that are stitched together in a predefined sequence to form a service function chain (SFC). Since an SFC request involves a number of VNFs, achieving an NFV environment raises two concerns namely: define and implement an SFC request and efficient mapping and scheduling VNFs of a given service. The European Telecommunications Standards Institute (ETSI) is collaborating with network operators and equipment vendors to foster NFV technologies and is currently working on how to implement SFC requests. However, the aspect of mapping and scheduling within the realm of VNFs has been subject to limited investigation, despite its increasing importance. With the constant growth and evolution of networks alongside the changing service requirements of users, relying on manual allocation of VNFs to specific physical machines is impractical for network operators. Hence, there is a need to explore more efficient and automated approaches to VNF mapping and scheduling in order to keep up with the dynamic nature of modern networks. The deployment and scheduling of the SFC onto the substrate network plays a vital role in determining the efficiency of resource utilization and performance of network management. E2E delay of a delay-sensitive network service request traversing a service function chain is the main parameter indicating deployment performance. Many of the research works have explored the VNF deployment problem for the optimization of E2E delay of the network service. Few works in the literature [1, 2] formulated the E2E delay of an SFC as a summation of packet transmission and propagation delays on each physical link without considering packet processing delays. However, each packet from different traffic flows passing through a physical machine usually requires different amounts of CPU processing time on the server. Each network service traverses through a variety of VNFs, thus, the processing of certain SFC requests can create different types of performance issues. Some SFC requests have large headers that can slow down CPU processing, while others have large packet payloads that demand longer transmission times. Moreover, the packet arrival process at a physical machine correlates with packet processing and transmission at its preceding physical machines.

In addition to transmission, propagation, and processing delays, edge queuing and virtualization delays also affect the scheduling of VNFs and the order in which VNFs are executed on those physical machines. Service level agreement (SLA) violations and incorrect schedules are produced when the controller does not take edge queuing and virtualization delays into account. Motivated by this, we formulated an end-to-end delay-aware SFC deployment and scheduling problem as an optimization problem while considering edge queuing delay, virtualization delay, processing delay, transmission delay, and propagation delay. Further, we proposed a scheduling algorithm based on the earliest finish time (SEFT) that specifically addresses the challenge of efficient end to end delay minimization in NFV-enabled networks. Unlike existing solutions, SEFT integrates various delays into the scheduling decision, providing a holistic optimization framework. By considering all the aforementioned delays, the proposed approach minimizes the total time taken for data to traverse the SFC, resulting in lower SLA violations and average end-to-end delays compared to existing approaches that consider only a subset of these delays. This is particularly important in scenarios with varying data sizes and a high number of SFC requests, where precise scheduling is critical to maintaining SLA compliance. Compared to other scheduling algorithms, SEFT's unique combination of VNF ranking and earliest finish time (EFT) based PM selection offers a significant improvement in reducing end-to-end delays. The detailed calculation of upward ranks ensures that the scheduling process considers both individual VNF requirements and overall SFC dependencies.

The contributions of this paper are summarized as follows:

- SFC mapping and scheduling problem for minimizing end-to-end delay is formulated as an integer non-linear programming problem (INLP).
- Transmission delay of virtual links, processing delay of VNFs at the physical machines, propagation delay, virtualization delay of physical machines, and queueing delay of physical edges are considered to formulate the end-to-end delay.
- An SFC scheduling approach based on the earliest finish time (SEFT) of physical machines is proposed.
- The impact of the end to end delay formulation on the SLA violations ratio is analysed while considering the delays such as transmission delay, processing delay, propagation delay, virtualization delay, edge queueing delay.
- The performance of the proposed approach is compared with the existing algorithms in terms of average end-to-end delay, service level agreement (SLA) violation ratio, resource utilization, and acceptance ratio.

The outline of the paper is as follows: Section 2 presents a detailed literature review on SFC deployment and scheduling. The system model and problem formulation are presented in section 3 and section 4, respectively. The proposed heuristic based on earliest finish time of VNFs for SFC scheduling problem is presented in Section 5. The performance of the proposed heuristic is evaluated by considering two benchmark network topologies and compared with existing approaches in Section 6. The concluding remarks are presented in section 7.

2 Related work

2.1 VNF placement

Various VNF placement approaches are proposed for the optimization of energy, cost, delay and resource utilization of SFCs in the NFV-enabled networks. Some integrate ensemble learning models with matching theory [3], while others formulate the problem as an integer linear program to minimize resource utilization and power consumption [4]. Deep reinforcement learning-based VNF placement approaches aim to improve performance degradation and migration costs [5], and strategies like privacy-preserving SFC deployment across multiple domains focus on resource utilization and privacy [6].

Furthermore, efforts have been made to minimize energy consumption and improve resource utilization using matching theory [7]. VNF placement scheme addressing latency and availability in mobile edge computing networks is proposed in [8], alongside an approach for joint optimization of bandwidth consumption and link utilization [9]. Additionally, demand-aware network function placement utilizing game theory and multi-layer neural network models for prediction are explored in [10] and [11]. Several strategies aim to minimize resource utilization and operational costs while meeting QoS requirements [12] and [13]. The SFC placement problem is formulated as an integer linear program to optimize the end to end delay while considering the propagation delay [14]. The authors of [15] proposed a VNF placement approach to minimize the end to end delay while taking queuing delay into consideration. The SFC deployment problem is formulated as an integer linear problem, and a heuristic is proposed for parallel SFC deployment in [16]. These studies provide valuable insights into optimizing VNF placement, but there remains a need for approaches that comprehensively address end-to-end delay considerations, including transmission, propagation, processing, queueing, and virtualization delays.

2.2 SFC placement and routing

In the realm of SFC placement and routing, a variety of methodologies have been proposed to optimize various aspects such as packet loss, jitter, delay, and deployment cost. The authors of [17] introduced a joint VNF placement and routing approach employing the lagrange relaxation method to enhance packet loss, jitter, and delay in NFV-enabled software-defined networks. Complementarily, a deep reinforcement learning-based model focusing on SFC deployment cost and delay optimization is presented in [18]. Furthermore, Xu et al. [19] formulated the SFC placement problem as an integer linear programming problem, while focusing on enhancing VNF availability through a layered graph approach. Similarly, [20] introduced a model to minimize the end-to-end delay by considering both propagation and processing delays, a framework particularly suited for SDN environments [21]. A VNF placement and routing model while considering the transmission delay and processing delays is proposed in [22]. To Address the dynamically varying traffic demands, reinforcement learning-based approaches for SFC deployment are explored in [23], while strategies based on Q-learning aimed at improving reliability is introduced in [24]. Additionally, [25] delved into optimizing end-to-end latency and deployment costs in SFC placement. In [26, 27], a VNF placement and chaining approach is proposed to optimize SFC acceptance rates and provider profits. Meanwhile, Zhang et al.. proposed a dynamic planning approach for SFC deployment in fog cloud computing environments, aiming to enhance resource utilization, throughput, and latency [28]. The authors of [29] addressed parallel SFC deployment in distributed networks, emphasizing efficient resource utilization. Furthermore, [30] proposed a delay-constrained SFC mapping problem to maximize service provider profits while meeting QoS requirements. Moreover, [31] formulated an SFC deployment model aimed at providing edge intelligence using distributed deep reinforcement learning. In a different context, [32]explored VNF placement in content distribution networks to minimize redundant resource usage. Furthermore, [33] explored SFC deployment optimization in elastic optic networks, formulating it as an integer quadratic program to optimize computing and spectrum resources. [34] addressed VNF placement and routing problem, aiming to maximize the number of accepted requests using a 0-1 integer linear programming model. An ant colony optimization-inspired dynamic VNF placement approach is proposed in [35] that focuses on minimizing both delay and cost factors in the network.

2.3 VNF/SFC scheduling

In the domain of VNF/SFC scheduling, various approaches have been proposed to optimize delay, availability, reliability, and resource utilization. Riera et al. pioneered VNF scheduling, formulating the joint VNF mapping and scheduling problem as a flexible job-shop problem [37]. Subsequent studies, such as [40], addressed VNF scheduling through mixed integer linear programming and evolutionary algorithms to minimize SFC schedule latency by considering transmission and processing delays. The authors of [36] proposed a VNF scheduling model using the evolutionary algorithm while considering transmission delay. In [41], the VNF scheduling problem is formulated as an integer nonlinear programming problem. The authors introduced a heuristic to optimize the latency and resiliency of the SFC schedule, taking into account the transmission and processing delays of VNFs. A tabu search based VNF mapping and scheduling framework while considering the processing delay is proposed in [38]. Qu et al. proposed a scheduling approach using parallel processing for the NFV-enabled SDNs [42]. This work focuses on improving the reliability and latency of SFCs. Control scheduling methods, as proposed in [43], aim to minimize QoS degradation duration by considering varying reconfiguration delays. Moreover, models for SFC deployment and flow scheduling in geo-distributed data centers [44] and NFV-enabled space-air-ground-integrated networks [45] have been proposed, considering additional costs and delays due to VNF re-instantiation and migration for optimizing total deployment cost and end-to-end delay. The authors of [46] introduced a comprehensive approach targeting network utilization, cost reduction, and end-to-end delay optimization. Subsequently, [47] formulated a mixed integer linear problem to minimize the makespan of all SFCs while meeting end-to-end delay requirements. Moreover, Gu et al. presented a fairness-aware flow scheduling framework, addressing equity concerns in VNF scheduling [48], while Cao et al. proposed an SFC deployment and scheduling model tailored for 6G networks, focusing on selecting servers with abundant resources to embed the SFC [49]. Additionally, dynamic SFC scheduling models, such as the one proposed in [50], aim to maximize the number of accepted SFCs through integer linear programming. Meanwhile, [51] tackled flow completion time reduction by considering transmission and processing delays.

Further advancements include cost-aware VNF placement and scheduling frameworks for dynamic SFCs in public cloud networks [52], energy-efficient scheduling models for parallel applications [53], and adaptive SFC scheduling models targeting network performance optimization and management overhead reduction [54].

While many studies address aspects of scheduling, there remains a gap in optimizing end-to-end delay comprehensively while considering various delay types. This underscores the necessity for holistic scheduling approaches that address propagation, processing, queueing, and virtualization delays for enhanced network performance and efficiency. Table 1 represents the comparison of the related works.

3 System model

In this section, we present the physical network model, SFC model, and decision variables used in problem formulation.

3.1 Physical network model

An NFV-enabled physical network hosts multiple network service functions on a common physical infrastructure, which is represented as a graph G =(P,E), where P denotes set of physical machines (PMs) and E denotes set of physical links connecting PMs. Multiple VNFs can be instantiated in each PM to support multiple VNF types. The resource capacity of each PM $j \in P$ is represented by $c_{p_j} = c_{p_j}^{cpu}$. Each physical link $e_{jj'}$ is represented as a tuple $\langle p_s, p_d, B_{e_{jj'}}, \lambda_{e_{jj'}}\rangle$, where $p_s, p_d \in P$ are end points of the edge, $B_{e_{jj'}}$ is bandwidth capacity of the edge, and $\lambda_{e_{jj'}}$ is the latency of the physical edge $e_{jj'}$.

3.2 Service function chain model

Let S be the set of SFC requests in the system, and $q \in S$ be the q^{th} SFC request. Each SFC request $q \in S$ is represented by a tuple $\langle v_{entry}, V_q, v_{exit}, B_q, D_q\rangle$. v_{entry} and v_{exit} represent the source and destination of the SFC request. $V_q = \{v_1, v_2, v_3..v_n\}$ is the predefined set of VNFs through which SFC request q has to pass through. B_q represents the bandwidth demand of the SFC request q. D_q represents the number of data bits to be transmitted from source to destination of the SFC request q.

3.3 Decision variables

We define three binary decision variables, $x_{i,q}^{j}$, $w_{ii',q}^{jj'}$, and $p_{q,q'}^{i,i'}$ to capture the relationship between the SFC request

Table 1 Comparison of related works

Reference	Transmission Delay	Propagation Delay	Queueing Delay	Processing Delay	Virtualization Delay
Qu et al. [36]	✓	✗	✗	✓	✗
Yang et al. [22]	✓	✗	✗	✓	✗
Riera et al. [37]	✗	✗	✗	✓	✗
Mijumbi et al. [38]	✗	✗	✗	✓	✗
Li et al. [39]	✓	✗	✗	✓	✗
Mohammad et al. [14]	✗	✓	✗	✗	✗
Ye et al. [15]	✗	✗	✓	✗	✗
Our work	✓	✓	✓	✓	✓

and the physical network. We introduce the binary decision variable $x^j_{i,q}$ to the placement state of VNF i on the PM j. It is set to one if VNF i of SFC q is mapped onto PM j, otherwise, it is set to zero. The decision variables used in the proposed model are summarized in Table 2.

$$x^j_{i,q} = \begin{cases} 1 & \text{if VNF } i \text{ of SFC } q \text{ is mapped onto PM } j \\ 0 & \text{Otherwise} \end{cases} \tag{1}$$

$w^{jj\prime}_{ii\prime,q}$ is the binary decision variable that defines the placement state of the virtual link $l^q_{ii\prime}$ placed on the physical link $e_{jj\prime}$.It is set to one if the virtual link $l^q_{ii\prime}$ is placed on the physical link $e_{jj\prime}$; otherwise, it is set to zero.

$$w^{jj\prime}_{ii\prime,q} = \begin{cases} 1 & \text{if virtual link } l^q_{ii\prime} \text{ of SFC} \\ & \text{is placed on physical link } e_{jj\prime} \\ 0 & \text{Otherwise} \end{cases} \tag{2}$$

$p^{i,i\prime}_{q,q\prime}$ is the binary decision variable that defines the processing order of two VNFs scheduled on the same PM. It is set to one if the VNF i of SFC q is processed before the VNF $i\prime$ of SFC $q\prime$ on PM j.

$$p^{i,i\prime}_{q,q\prime} = \begin{cases} 1 & \text{if VNF } i \text{ of SFC } q \text{ gets processed} \\ & \text{before the VNF } i\prime \text{ of SFC } q\prime \\ 0 & \text{Otherwise} \end{cases} \tag{3}$$

Table 2 Summary of Notations

Notations	Description
i	VNF
j	Physical machine
$e_{jj\prime}$	Physical edge connecting PMs j and $j\prime$
q	SFC request
c^{cpu}_j	Processing capacity of PM
$B_{e_{jj\prime}}$	Bandwidth capacity of physical edge
$\lambda_{e_{jj\prime}}$	Allowed latency of physical edge $e_{jj\prime}$
v_{entry},v_{exit}	source and destinations of SFC
B_q	Bandwidth demand of SFC q
D_q	Data size to be processed by SFC q
λ_q	Arrival rate of SFC requests in network
d^{cpu}_i	Processing capacity demand of SFC
$b^{q,l_{ii\prime}}$	Bandwidth demand of SFC
v	speed of light
H_j	number of VNFs co located
$U^{bw}_{ii\prime}$	link utilization
λ_j	cumulative traffic load of the PM
$x^j_{i,q}$	1 if VNF i of SFC q mapped is onto PM j
$w^{jj\prime}_{ii\prime,q}$	1 if virtual link $l^q_{ii\prime}$ is placed on physical link $e_{jj\prime}$
$p^{i,i\prime}_{q,q\prime}$	1 if VNF i of SFC q gets processed before the VNF $i\prime$ of SFC $q\prime$

4 Problem formulation

The objective is to minimize the end-to-end delay of SFCs while satisfying placement, capacity, scheduling, flow, and domain constraints.

4.1 Objective function

For each SFC request, we evaluate the end-to-end delay T_{End} by taking the following delays into consideration: transmission delay T_t, queueing delay T_q, propagation delay of physical links T_{prop}, virtualization delay T_{virt}, and processing delay T_{proc}.

$$T_{End} = T_t + T_{queue} + T_{prop} + T_{virt} + T_{proc} \tag{4}$$

4.1.1 Transmission delay

Data packets belonging to an SFC experience transmission delays as they transit through a link connecting two VNFs located on different physical machines. The transmission delay (T_t) of an SFC request is the sum of transmitting data on all edges. The transmission delay of SFC q on virtual link $l_{ii\prime}$ can be formulated as below [36]:

$$T_t = \sum_{l^q_{ii\prime} \in L} \frac{D_q}{b^{q,l_{ii\prime}}} \qquad \forall l \in L \tag{5}$$

Where D_q is the number of data bits to be transmitted from the source to destination of SFC q. $b^{q,l_{ii\prime}}$ represents the bandwidth demand of the virtual link $l^q_{ii\prime}$.

where D_{e_i} is the inherent propagation delay of the physical edge e_i.

4.1.2 Propagation delay

The propagation delay is directly proportional to the length of the physical link and is independent of the traffic load [14].

$$T_{prop} = \frac{dist(j, j\prime)}{v} \tag{6}$$

where $dist(j, j\prime)$ is the length of the physical edge connecting physical machines j and $j\prime$, and v is the propagation speed or speed of light.

4.1.3 Virtualization delay

The virtualization overhead of a physical machine is a function of co-located VNFs H and the cumulative traffic load λ [55]. The virtualization delay of a physical machine is as follows:

$$f(H,\lambda) = 0.338 * H * \lambda^{12.15} + 0.51 * \lambda; \quad \lambda = \sum_{i=1}^{V} \lambda_i \quad (7)$$

However, the above expression is nonlinear, which can be piecewise linearized as follows:

$$H_j = \sum_{q\in S}\sum_{i\in V_q} x^j_{i,q}, \qquad \forall j \in P \quad (8)$$

$$\lambda_j = \sum_{q\in S}\sum_{i\in V_q} x^j_{i,q} \cdot \lambda^{q,i}, \qquad \forall j \in P \quad (9)$$

Where H_j is the number of VNFs co-located on the physical machine j. λ_j is the cumulative traffic load of the physical machine j. The final equation for calculating the virtualization delay is transformed as below:

$$T_{virt} = \sum_{j\in P} 0.338 * H_j * \lambda_j^{12.15} + 0.51 * \lambda_j \quad (10)$$

4.1.4 Processing delay

The utilization of a physical machine is the ratio between processing demand and available processing capacity of the physical machine. It is defined as below:

$$U^{\text{cpu}}_j = \frac{\sum_{q\in S}\sum_{i\in V_q} x^j_{i,q} \cdot d^{cpu}_i}{c^{cpu}_j} \quad (11)$$

For an SFC request q, the arrival rate and processing delay of a VNF follow a Poisson and exponential distribution, respectively. Hence, we can adopt the M/M/1 queueing model for each VNF. Therefore, the processing delay [36] of a VNF node i of the SFC request is evaluated as below:

$$T^i_{proc} = \frac{1}{c^{cpu}_j\left(1 - U^{\text{cpu}}_j\right)} \quad (12)$$

The processing delay T_{proc} of a service function chain is the sum of processing delays of all VNFs.

$$T_{proc} = \sum_{i\in V_q} T^i_{proc} \quad (13)$$

4.1.5 Queueing delay

As the SFC transfers traffic across various physical machines, it experiences queueing delay which is determined by the utilization of each link. Link utilization can be formulated as the ratio between total traffic demand and link capacity [15].

$$U^{\text{bw}}_{ii'} = \frac{\sum_{q\in S}\sum_{l^q_{ii'}\in L} w^{jj'}_{ii',q} \cdot d^{q,l_{ii'}}}{B_{e_{jj'}}} \quad (14)$$

The queueing delay of the service function chain at the VNF i is formulated as below :

$$T_{queue} = \frac{1}{B_{e_{jj'}}\left(1 - U^{\text{bw}}_{ii'}\right)} \quad (15)$$

4.2 Constraints

To successfully schedule and deploy an SFC in a physical network, the following constraints need to be satisfied.

4.2.1 Capacity constraints

It is essential that the resource requirements of SFCs deployed on the physical network do not exceed the resources available on the physical machines and links. This is captured using node capacity and link capacity constraints:

$$\sum_{i\in V_q} x^j_{i,q} \cdot d^{cpu}_i \le c^{cpu}_j, \quad \forall j \in P \quad (16)$$

$$\sum_{l^q_{ii'}\in L}\sum_{q\in S} B_q \times w^{jj'}_{ii',q} \le B_{e_{jj'}} \quad \forall (j, j\prime) \in E \quad (17)$$

Equation 16 ensures that the required processing capacity of the VNFs deployed on the physical machine does not exceed the total available processing capacity of the PM. Equation 17 ensures that the total bandwidth requirements of virtual links of SFC requests do not exceed the available bandwidth limit of the physical links.

4.2.2 Flow constraint

For all the physical machines except for the source and destination nodes, the incoming flow has to be equal to the outgoing flow. It is captured using the following constraint:

$$\sum_{e_{jj'}\in E} w^{jj'}_{ii',q} - \sum_{e_{j'j}\in E} w^{j'j}_{ii',q} = x^j_{i,q} - (x^j_{i',q}) \quad (18)$$
$$\forall j \in P, l^q_{ii'} \in L, q \in S$$

4.2.3 VNF placement constraints

$$x^j_{i,q} \leq 1, \qquad \forall i \in V_q, q \in S \tag{19}$$

Equation 19 guarantees that each VNF is scheduled on at most one physical machine.

4.2.4 SFC scheduling constraints

If two VNFs i and i' are requested by different SFCs q and q' and both i and i' are mapped onto the same physical machine j , i.e., $x^j_{i,q} = 1$ and $x^j_{i\prime,q\prime} = 1$, then the PM j should not process those two SFC requests at the same time. This is captured using the VNF scheduling constraint in Eq. 20:

$$1 \geq p^{i,i\prime}_{q,q\prime} + p^{i\prime,i}_{q\prime,q} \geq x^j_{i,q} + x^j_{i\prime,q\prime} - 1 \forall q, q\prime \in S; i, i\prime \in V_q \tag{20}$$

The SFC scheduling problem to minimize the end to end delay is formulated as below:

$$\min \ T_{End} \tag{21}$$
$$s.t \quad Eq.(16) - Eq.(20)$$

4.2.5 Domain constraints

$$x^j_{i,q} \in \{0, 1\} \forall i \in V_q, q \in S \tag{22}$$

$$w^{jj'}_{ii\cdot,q} \in \{0, 1\} \forall i \in V_q, q \in S \tag{23}$$

$$p^{i,i\prime}_{q,q\prime} \in \{0, 1\} \forall i \in V_q, q \in S \tag{24}$$

Equations 22, 23 and 24 ensure that the decision variables $x^j_{i,q}$, $w^{jj'}_{ii\cdot,q}$ and $p^{i,i\prime}_{q,q\prime}$ take values either zero or one.

Note that the M/M/1 queueing model is used to formulate the processing delay, which makes the problem non-linear. Minimum dominating set problem can be reduced to the SFC scheduling problem in polynomial time. Hence, the SFC scheduling problem is NP hard [56]. In the next section, a heuristic based on the earliest finish time is proposed to solve the SFC scheduling problem.

5 Proposed algorithm

This section first describes the high-level overview of the proposed scheduling approach. We then present step by step description of the proposed algorithms.

The proposed SFC Scheduling using the earliest finish time (SEFT) aims to optimize the mapping and scheduling of SFCs onto physical machines in a NFV-enabled network. By leveraging the earliest finish time of physical machines, the algorithm seeks to minimize end-to-end delay while efficiently allocating resources and reducing the waiting time for data transmission and processing. The algorithm consists of two main phases: VNF ranking and physical machine selection. In the VNF ranking phase, the algorithm computes ranks for each VNF based on their average processing time on physical machines and the average communication delay between the physical machines. These ranks are then used to determine the order in which the VNFs will be scheduled onto physical machines. In the physical machine selection phase, VNFs are sequentially mapped onto physical machines in a way that minimizes the earliest finish time (EFT). This involves calculating the EFT of each VNF on each physical machine and selecting the machine with the minimum EFT for each VNF. After scheduling a VNF onto a physical machine, the resource capacity of that physical machine is updated to reflect the allocated VNF.

The proposed SFC scheduling approach based on earliest finish time is presented in Algorithm 1. This approach is designed to operate efficiently by considering the available physical resources and the specific requirements of the SFC request. It begins by receiving two primary inputs: a set of physical machines denoted as P, and an SFC request q. It produces a scheduling solution for the given SFC as output. In the VNF ranking phase (Steps 1-4), each VNF within the SFC is evaluated to determine its rank or priority in the chain. This evaluation involves computing the "upward rank" for each VNF that essentially prioritizes the VNFs based on factors such as processing time and communication delays. Subsequently, in Step 4, the VNFs are sorted in descending order of their computed ranks, establishing a hierarchy that guides the scheduling process. The Physical Machine Selection phase (Steps 5-12) focuses on assigning VNFs to available physical machines in a manner that minimizes processing delays. Beginning with the highest-ranked VNF, the algorithm systematically evaluates the EFT value on each physical machine and selects the one with the minimum EFT value for assignment. This iterative process continues until all VNFs are successfully scheduled onto physical machines. Specifically, the first VNF with the highest rank is selected in step 6. In steps 7-9, EFT of VNF on each PM is calculated using the algorithm 3. The VNF with the maximum upward rank is placed onto the PM with the minimum EFT value in step 10. After scheduling the VNF onto the PM, the resource capacity of the physical machine is updated in step 11. Steps 5-12 are repeated until all VNFs are scheduled. Throughout the execution of the algorithm, the resource capacity of each physical machine is dynamically updated to ensure efficient utilization. This ensures that the scheduling solution adapts to the changing workload and resource availability within the network.

The algorithm for computing the upward rank of a VNF is presented in Algorithm 2. By recursively traversing the SFC starting from the exit VNF, it computes the upward rank that reflects the importance of each VNF in the chain. The upward rank is based on various factors such as the data size of the VNF, processing time, communication startup time, and data transfer rate between physical machines.

Algorithm 1 SEFT.

```
Input: SFC $q$
       PM set P
Output: Scheduling solution of SFC $q$
1: for each VNF $i \in V_q$ do                          ▷ VNF Ranking phase
2:     find upward rank $rank_u(i)$ using Algorithm 2
3: end for
4: Sort VNFs in the descending order of $Rank_u$
5: while all VNFs are not placed do                     ▷ PM Selection phase
6:     Pick first VNF from the sorted list of $Rank_u$
7:     for each PM $j \in$ P do
8:         Find $EFT(i, j)$ using Algorithm 3
9:     end for
10:    place VNF $i$ on physical machine $j$ with minimum $EFT(i, j)$
11:    update resource capacity of each PM $j$ when VNF $i$ is scheduled
       and deployed
12: end while
13: return Scheduling solution of SFC $q$.
```

Algorithm 2 takes the data size of VNF to be processed, VNF processing time matrix W, average communication startup time $\overline{S}$, and average data transfer rate $\overline{R}$ as input. W is a matrix of size $V \times P$ wherein the $(i, j)^{th}$ entry, i.e., $w_{i,j}$ represents the estimated execution time to finish processing of VNF i on PM j. R is a matrix of size $P \times P$ that comprises the data transfer rates between PMs. The communication startup costs of PMs are stored in P dimensional vector S. The algorithm returns an upward rank of VNF $rank_u(i)$ as output. If the VNF is an exit VNF V_{exit}, then the rank of the VNF is set to the average processing time of the VNF.

$$\text{rank}_u\left(v_{\text{exit}}\right) = \overline{w_{\text{exit}}}. \tag{25}$$

$$\overline{w_{exit}} = \frac{\sum_{j=1}^{P} w_{exit,j}}{|P|}. \tag{26}$$

where $\overline{w_{\text{exit}}}$ represents the average processing time of exit VNF. It is defined as the ratio between the sum of execution time of exit VNF on each physical machine and the number of physical machines. The upward rank of VNFs other than exit VNF is recursively defined in Eq. 27. It is the sum of the length of the critical path from the VNF to the exit VNF and the average processing time of VNF.

$$\text{rank}_u\left(i\right) = \overline{w_i} + \max_{k \in \text{succ}(i)} \left\{\overline{c_{i,k}} + \text{rank}_u\left(k\right)\right\} \tag{27}$$

where $succ(i)$ is the set of successors of VNF i, and $\overline{c_{i,k}}$ represents the average communication time to transfer the data between the PMs onto which those VNFs i and k are mapped.

The average communication time of a link (i, k) is defined as below:

$$\overline{c_{i,k}} = \overline{S} + \frac{Data_i}{\overline{R}} \tag{28}$$

The detailed step-by-step explanation of Algorithm 2 provides insights into how the upward rank of each VNF is computed. In steps 2-5, the algorithm begins by computing the total processing time for each VNF on every available physical machine. Subsequently, in step 6, the average processing time of $\overline{w_i}$ of each VNF i is calculated based on the processing times. In step 7, the average communication time $\overline{c_{i,k}}$ of VNF i is calculated using the average communication start-up time $\overline{S}$, data to be processed $Data_i$, and average data transfer rate between PMs $\overline{R}$. For the exit VNF V_{exit} as outlined in steps 8-10, its rank is simply set to its average processing time. This straightforward approach ensures that the exit VNF's rank is solely determined by its processing duration, given its lack of dependencies on successors. For VNFs other than the exit VNF, steps 11-20 detail the process of calculating their upward ranks. This involves considering each successor k of the VNF i and computing the total delay, which is the sum of the average communication time $\overline{c_{i,k}}$ and the upward rank of the successor k as shown in step 13. In steps 14-16, the successor with maximum total delay is determined. This total delay is the summation of the average communication time $\overline{c_{i,k}}$ and the upward rank of the successor k denoted as $rank_u(k)$.

The upward rank of VNF $rank_u(i)$ is computed in step 17. The algorithm for computing the EFT of VNF i on PM j is presented in Algorithm 3. It takes as input PM j, SFC q, data size of VNF $Data_i$, VNF processing time matrix W, communication startup time vector S, and data transfer rate matrix R. It produces as output earliest finish time value $EFT(i, j)$ of VNF i on PM j. $EFT(i, j)$ is the sum of earliest start time of VNF i on the PM j and execution time of VNF i on the PM j. Let $EST\ (i, j)$ be the earliest execution start time of VNF i on the physical machine j. If the VNF i is an entry VNF v_{entry}, then the $EST(i, j)$ is set to zero.

$$\text{EST}\left(v_{\text{entry}}, j\right) = 0 \tag{29}$$

The EST value $EST\ (i, j)$ for VNFs other than entry VNF is presented in Eq. 30.

$$\text{EST}\left(i, j\right) = \max\left\{\text{avail}[j], \max_{m \in \text{pred}(i)}\left(AFT\left(m\right) + c_{m,i}\right)\right\}, \tag{30}$$

Algorithm 2 Finding Upward Rank of VNF.

```
Input: VNF i
        VNF data size: $Data_i$
        VNF processing times matrix: $W$
        Average Communication startup time: $\overline{S}$
        Average data transfer rate: $\overline{R}$
Output:Upward rank of VNF: $rank_u(i)$
 1: function UPWARD- RANK(i)
 2:     $total\_w_{ij} = 0$
 3:     for each PM $j \in P$ do
 4:         $total\_w_{ij}$ += $w_{ij}$
 5:     end for
 6:     $\overline{w_i} = total\_w_{ij}/|P|$
 7:     $\overline{c_{i,k}} = \overline{S} + Data_i/\overline{R}$
 8:     if i== $v_{exit}$ then
 9:         $rank_u(i)$= $\overline{w_i}$
10:     else
11:         max-total-delay = 0
12:         for each $k \in successor(i)$ do
13:             total-delay = $\overline{c_{i,k}}$ + UPWARD-RANK($k$)
14:             if max-total-delay < total-delay then
15:                 max-total-delay = total-delay
16:             end if
17:             $rank_u(i) = \overline{w_i}$ + max-total-delay
18:         end for
19:     end if
20: end function
21: return $rank_u(i)$
```

Algorithm 3 Finding EFT value of VNF on PM.

```
Input: PM $j$, VNF: $i$, SFC: $q$
        VNF Data Size: $Data_i$
        VNF Processing Time Matrix: W
        Communication Startup Time Matrix: $S$
        Data Transfer Rate Matrix: $R$
Output: EFT value of VNF $i$ on PM $j$: $EFT(i, j)$
 1: function EFT- VALUE(i,j)
 2:     for each VNF $i \in V_q$ do
 3:         if i==$v_{entry}$ then
 4:             $EST(i, j) = 0$
 5:         else
 6:             max-ready-time = 0
 7:             for each $m \in predecessors(i)$ do
 8:                 $c_{m,i} = S_{j\prime} + Data_i/R_{j\prime,j}$
 9:                 ready-time = AFT($m$) + $c_{m,i}$
10:                 if max-ready-time < ready-time then
11:                     max-ready-time = ready-time
12:                 end if
13:             end for
14:             $EST(i, j)$ = max(max-ready-time + $w_{i,j}$)
15:         end if
16:         $EFT_{i,j}$ = EST-VALUE(i,j) + $w_{i,j}$
17:     end for
18: end function
19: return $EFT(i, j)$
```

where $pred\,(i)$ is the set of immediate predecessors of VNF i, $avail[j]$ is the available time of PM. The available time is the time when a PM finishes processing the VNFs mapped onto it and is ready to process a new VNF. The inner max block in Eq. 30 is ready-time. It is the time when all the data required by the VNF i reaches PM j. The ready-time of VNF i is calculated as the sum of the predecessor's actual finish time $AFT(m)$ and communication time $c_{m,i}$ between the predecessor m and VNF i respectively. Finally, the outer max block of Eq. 30 returns the time when PM j is ready to process the VNF i, and all the data needed by the VNF i is available at PM j. $c_{m,i}$ is the communication time to transmit data between two VNFs m and i when they are scheduled on PMs $j\prime$ and j respectively. It is defined as below:

$$c_{m,i} = S_{j\prime} + \frac{Data_i}{R_{j\prime,j}} \tag{31}$$

where the first term $S_{j\prime}$, is the communication startup time of PM $j\prime$ and the second term is the ratio between data size $Data_i$ to be transmitted between the VNFs m and i and data transfer rate $R_{j\prime,j}$ between the PMs $j\prime$ and j respectively.

EFT of VNF i on PM j is calculated as the sum of VNF processing time $w_{i,j}$ on PM j and EST $EST(i, j)$. It is defined as follows:

$$\text{EFT}\,(i, j) = w_{i,j} + \text{EST}\,(i, j) \tag{32}$$

The detailed step by step explanation of Algorithm 3 is as follows. Steps 2-14 of the algorithm calculate the EST of VNF. If the VNF is not an entry VNF, then the EST value for the VNF is calculated in steps 5-15.

The communication time $c_{m,i}$ between VNF i and each predecessor m is determined in step 8. The ready-time of VNF i is calculated in step 9. In steps 10-12, the maximum ready time of the predecessor is determined. Step 14 computes the EST value of VNF i on PM j as the maximum of max-ready-time and processing time of VNF $w_{i,j}$. The EFT value of the VNF i on PM j is computed in step 16. Finally, the EFT value of the VNF is returned in step 19.

5.1 Time complexity

Steps 1 - 3 of the Algorithm 1 invoke Algorithm 2 for $\mathcal{O}(|V_q|)$ times where $|V_q|$ is the number of VNFs. Steps 3 - 5 of the Algorithm 2 take $|P|$ time, where $|P|$ is the number of physical machines. Steps 6-9 take constant time, and the loop in steps 12 - 18 takes at most $|V_q|$ times. Therefore, the overall running time of algorithm 2 is $|P| + |V_q| = \mathcal{O}(|P|)$. Step 4 of Algorithm 1 takes $\mathcal{O}(|V_q|\log|V_q|)$ time. Steps 7 - 9 of Algorithm 1 invoke Algorithm 3 for $|P|$ times. Steps 7 - 13 of Algorithm 3 take $\mathcal{O}(|V_q|)$ time. The overall time complexity of Algorithm 3 is $\mathcal{O}(|V_q|)^2$. Hence, steps 5 - 12 of Algorithm 1 take $\mathcal{O}(V_q * |P| * (|V_q|)^2)$ time. Therefore, the total time complexity of proposed SEFT is $\mathcal{O}(|V_q| * |P|)$+ $\mathcal{O}(|V_q|\log|V_q|)$+ $\mathcal{O}(V_q * |P| * (|V_q|)^2) \approx \mathcal{O}(|P||V_q|)^3$.

6 Results and analysis

In this section, we describe the networks used for simulation, existing approaches used for comparison with the proposed work, and performance comparison of the proposed algorithm with existing approaches from the literature.

6.1 Simulation setting

We considered two networks from internet topology [57] to evaluate the performance of the proposed and baseline approaches. The networks include: AT&T Network(28 nodes and 45 links) and Iris Network (51 nodes and 64 links). The latitude and longitude of each node, along with the bandwidth of links, are extracted from the graphml file corresponding to the input networks. The delay of each link is calculated using bandwidth and distance, i.e., delay = distance/bandwidth. The available processing capacity of the Physical Machine (PM) is considered as the target resource in resource management and allocation. The processing capacity generally refers to the CPU resources, which play a crucial role in determining the performance and efficiency of the PM [58]. The bandwidth of the physical links is generated in the range of 1 KBps to 10 KBps. Each SFC randomly selects a pair of nodes as its source and destination nodes. The size of the SFC or the number of VNFs in the SFC is uniformly generated in the range [5, 10] The processing capacity needed by each VNF is generated from 1 KB to 50 KB. The bandwidth demanded by the virtual links between the VNFs is generated within the range of 1 KBps to 50 KBps. Specifically, to implement the proposed SFC scheduling model, we used Python version 3.10 running on an Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz with 64 GB of memory. The simulation parameters used for the performance evaluation of the proposed study are represented in Table 3.

The proposed SEFT algorithm is compared with three existing algorithms: i) Greedy Best Availability algorithm (GBA) ii) Greedy Least Loaded algorithm (GLL) iii) Stable Matching algorithm (MA) and iv) Cost-Efficient VNF Placement and Scheduling (CE-VPS). These algorithms are briefly described below:

1. **Greedy Best Availability algorithm:** This approach solves the SFC mapping and scheduling problem by choosing physical machines with the highest remaining processing capacity. The above process continues until all VNFs of SFC are mapped and scheduled.
2. **Greedy Least Loaded algorithm:** In this approach, the SFC mapping and scheduling problem is solved by choosing the physical machines with the minimum number of VNFs placed on them. The above process continues until all VNFs of SFC requests are mapped and scheduled.
3. **Stable Matching Algorithm:**
   We also compare our proposed approach with a matching theory based VNF scheduling model [59]. This approach computes preference lists for each physical machine as well as for each VNF. Then, each VNF picks the first physical machine from its preference list. If PM has enough resources, the VNF is placed on PM. Otherwise, the VNF rejects the PM and removes that PM from its preference list. Then, the PM also rejects and removes the VNF along with all lower-priority VNFs from its preference list. This matching procedure is repeated until all the VNFs of SFC are mapped onto the ideal physical machines. Finally, the VNF mapping and scheduling solution is returned.
4. **Cost-efficient VNF scheduling approach:** We also compare our proposed model with cost-efficient VNF scheduling (CE-VPS) [60]. In this approach, physical machines in the network are grouped into a finite number of zones. The zone that can host the maximum number of VNF instances is defined as the optimal zone. First, the optimal zone is determined using the matrix-based method to place the VNFs of the given SFCs. For each VNF, the qualified PMs i.e., those PMs can satisfy the resource capacity demand of the given VNF within the specified deadline, are defined as candidate PMs. After determining the candidate PMs, the VNFs are scheduled on the candidate PMs from the optimal zone. If the PMs in the optimal zone do not have enough resource capacity or are unable to process the VNFs by the defined deadline,VNFs are scheduled on random PMs from other zones. If no VNF instance is available with enough resource capacity to host a VNF, a new VNF instance is installed on a PM.

6.2 Simulation results

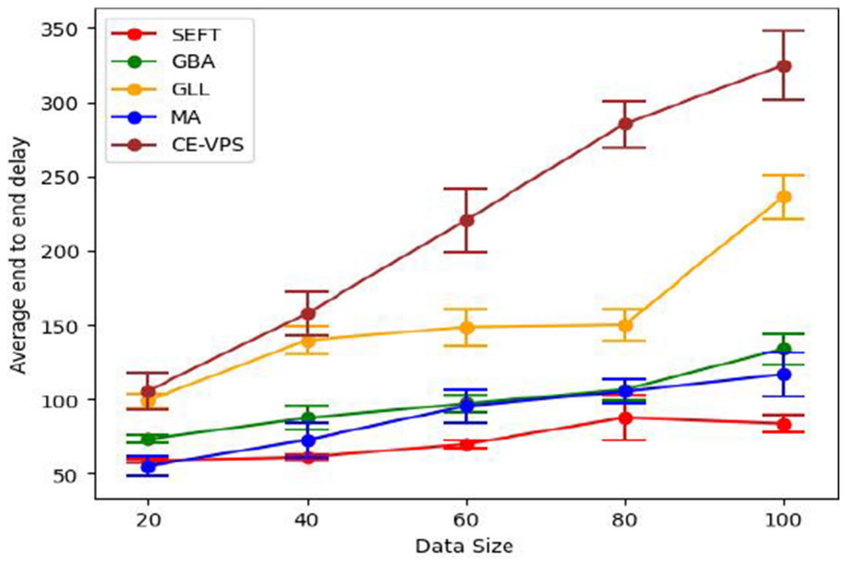
This subsection summarizes the performance comparison of the proposed SEFT algorithm with the existing GBA, GLL, MA, and CE-VPS algorithms in terms of average end-to-end delay, SLA violation ratio, resource utilization, and acceptance ratio. Further, the impact of formulating the end to end delay on SLA violations ratio is presented.

6.2.1 End to end delay with variation of data size

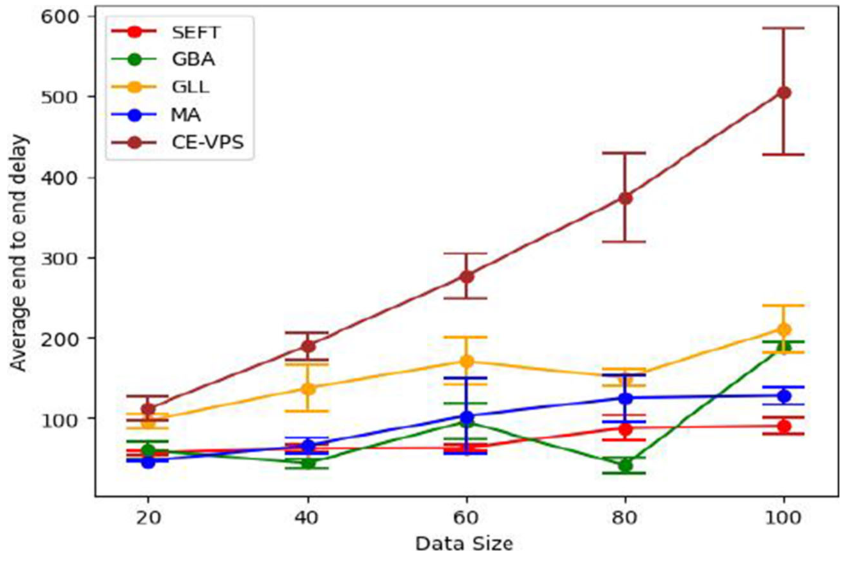
Figure 1 represents the 95% confidence interval for the average end-to-end delay with the variation of data size on AT&T network. The figure compares the performance of the proposed approach on AT&T network to GBA, GLL, MA, and CE-VPS approaches in terms of average end to end delay of the SFC request.The number of SFC requests is set to 50 and size of data to be processed is varied from 20 KB to 100 KB. Fig. 1a illustrates that the proposed SEFT approach

Table 3 Simulation parameters

Parameter	Range
Available processing capacity of PM	[400, 600, 800, 1000] KBps
Available processing capacity of physical link	[1, 10] KBps
Required processing capacity of VNF	[1, 50] KBps
Required bandwidth of virtual link	[1, 50] KBps
SFC length	[4, 8]
SFC Deadline	60 sec



(a)



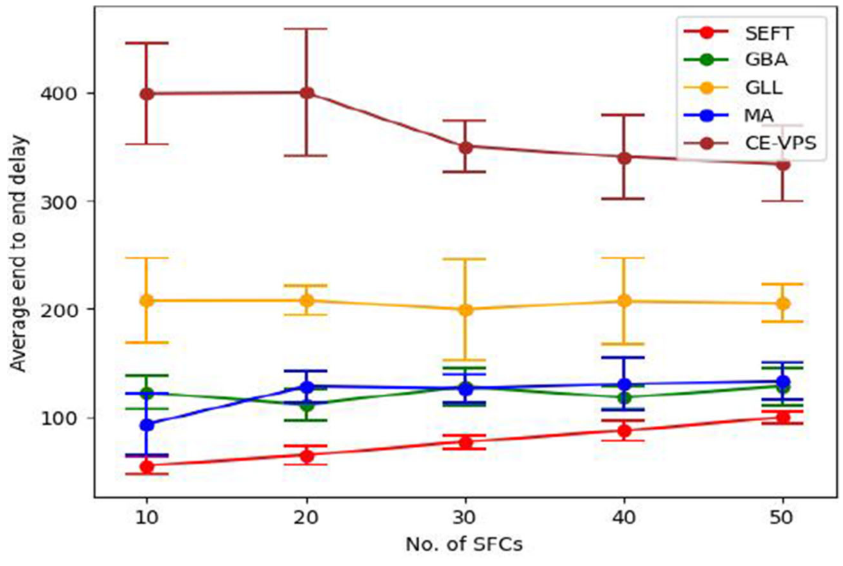
(b)

Fig. 1 Variation of average end to end delay with data size (a) AT&T Network. (b) Iris Network

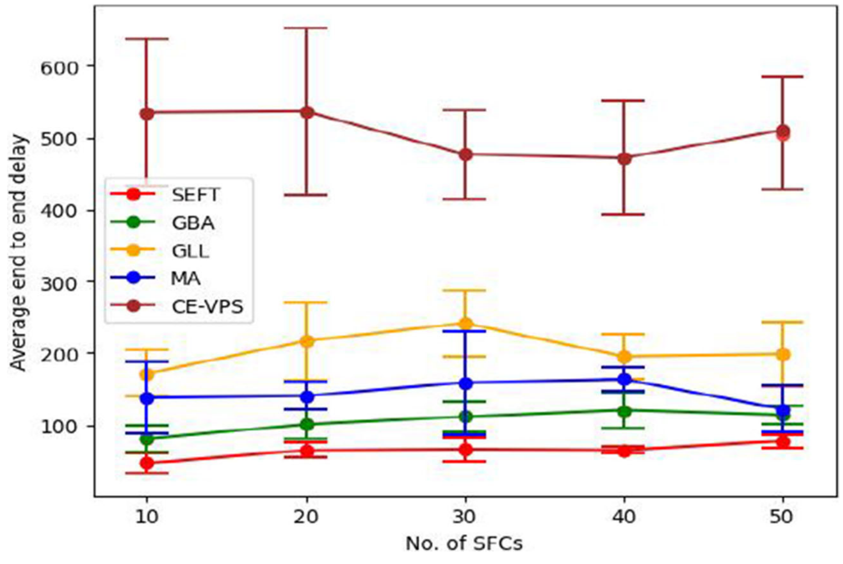
has the minimum average end-to-end delay. The average delay of the proposed approach ranges from 58 to 83 seconds for 20 KB to 100 KB data size, respectively. The proposed approach achieves at least 26.31%, 51.04%, 14.97% and 63.05% lower average end to end delay over the GBA, GLL, MA and CE-VPS approaches, respectively. Because the proposed approach prefers physical machines with minimum finish times while considering the delay required to transfer the data among those physical machines. Further, it can also be observed that the average end-to-end delay increases with the data size for every approach. It can be observed from Fig. 1b the average end-to-end delay of the proposed approach on the IRIS network when compared with existing approaches. The results are plotted with a confidence level of 95%. The average end-to-end delay of the GBA and MA approaches is close to the SEFT approach when the data size is 20 KB. However, the proposed approach results in at least 50.59%, 12.75% and 69.85% lower average end-to-end delay over the GLL, MA and CE-VPS approaches, respectively.

6.2.2 End to end delay with variation of number of SFC requests

Figure 2 describes the performance of the proposed SEFT approach and existing approaches in terms of average end-to-end delay while varying the number of SFC requests on AT&T and IRIS networks. The size of the data to be processed is set to 100 KB, and the number of SFC requests



(a)



(b)

Fig. 2 Variation of average end to end delay with number of SFCs. (a) AT&T Network. (b) Iris Network

is varied from 10 to 50. It can be observed from Fig. 2a that the proposed approach results in lowest average end-to-end delay when compared to GBA, GLL, MA and CE-VPS approaches. Specifically, the average end to end delay of our proposed approach is at least 36.27%, 61.52%, 34.60%, and 78.28% lower than the GBA, GLL, MA, and CE-VPS approaches, respectively. Figure 2b shows average end-to-end delay obtained by SEFT approach compared with GBA, GLL, MA, and CE-VPS approaches while varying the number of SFC requests on the IRIS network. Since this work is primarily aimed on optimizing the end-to-end delay, the average end-to-end delay of our proposed approach is at least 38.64%, 67.18%, 51.92%, and 86.99% lower than GBA, GLL, MA, and CE-VPS approaches, respectively. As the existing works not formulated the end to end delay while including the delays such as transmission delay, processing delay, propagation delay, queuing delay, and virtualization delay, resulted in increased average end to end delays.

6.2.3 Impact of formulating end-to-end delay on SLA violations ratio

We consider an end-to-end delay requirement of 60 seconds for every SFC request. Hence, a SLA violation occurs when an SFC request takes more than 60 seconds to process. SLA violation ratio is defined as the fraction of SFC requests that violates the end to end delay requirement. The impact of formulating end to end delay while considering virtualization and edge queuing delays on SLA violations ratio while varying the data size on the AT & T network is presented in the Fig. 3. The proposed SEFT approach while considering virtualization and edge queuing in end to end delay formulation achieves 38.83% lower SLA violations ratio when compared to the SEFT approach which does not consider the virtualization delay and edge queuing delay in end to end delay formulation. This is due to the inclusion of virtualization and edge queuing delays, which allows for more accurate modeling of real-world network conditions, leading to better optimization and management of resources to meet SLA requirements.

Figure 4 illustrates the impact of formulating end to end delay while considering virtualization delay and processing delay on SLA violations ratio while varying the data size on the AT & T network. The proposed SEFT approach with virtualization delay and processing delay in end to end delay formulation achieves 44.62% lower SLA violations ratio when compared to the SEFT approach which does not consider the virtualization delay and processing delay in end to end delay formulation.

Figure 5 illustrates the impact of formulating end to end delay while considering transmission delay, edge queueing delay,propagation delay on SLA violations ratio while varying the data size on the AT & T network. We can observe that the proposed SEFT approach with transmission delay, edge queuing delay and propagation delay in end to end delay formulation results in 56.72% lower SLA violations ratio compared to the SEFT approach that does not consider the aforementioned delays.

6.2.4 Variation of SLA violation ratio with data size

Figure 6 evaluates the efficiency of proposed algorithm compared with the baseline approaches in terms of SLA violation ratio. The total number of SFC requests is fixed to 50 and the data size to be processed is varied from 20 KB to 100 KB. Figure 2a illustrates the SLA violation ratio of proposed approach and existing approaches on AT&T network. It can be observed from Fig. 2a that the SEFT approach has the lowest SLA violation ratio for all data sizes. The SLA violation ratio of the existing approaches are increasing with the data size. As the proposed approach formulated the end to end delay while considering the transmission delay, processing delay, propagation delay, queuing delay, and virtualization delay, it resulted in the lowest SLA violations compared to the existing approaches. Specifically, our proposed approach achieves at least 32.21%, 39.63%, 12.72% and 38.30% of reduction in the SLA violation ratio when compared to the GBA, GLL, MA and CE-VPS approaches, respectively. The GLL approach prioritizes the least-loaded physical machines without focusing on any communication delays between the machines while scheduling VNFs. Therefore, the GLL approach results in a higher end-to-end delay as well as SLA violation ratio. Figure 2b shows the SLA violation ratio of the proposed SEFT approach and existing approaches on iris network. We can observe that the SLA violation ratio of our proposed approach is at least 39.08%, 5.67%, 42.04% lower than GLL, MA, CE-VPS approaches, respectively.

6.2.5 SLA violation ratio with variation of number of SFCs

Figure 7 describes the performance of proposed and existing approaches in terms of SLA violation ratio while varying the number of SFC requests on AT&T and IRIS networks. The size of data to be processed is set to 100 KB and the number of SFCs is varied from 10 to 50. Figure 7a shows the performance of the proposed SEFT approach and existing GBA, GLL, MA and CE-VPS approaches in terms of SLA violation ratio for the AT&T network. GBA approach selects the physical machines with the most remaining resources for the deployment of VNFs. It causes more delay as the delay required to transfer the data among the physical machines is not focused. Subsequently, the GBA approach results in a higher SLA violation ratio. In GLL, MA and CE-VPS approaches, the delay required to transfer data among the physical machines is also not given much importance. Thus,

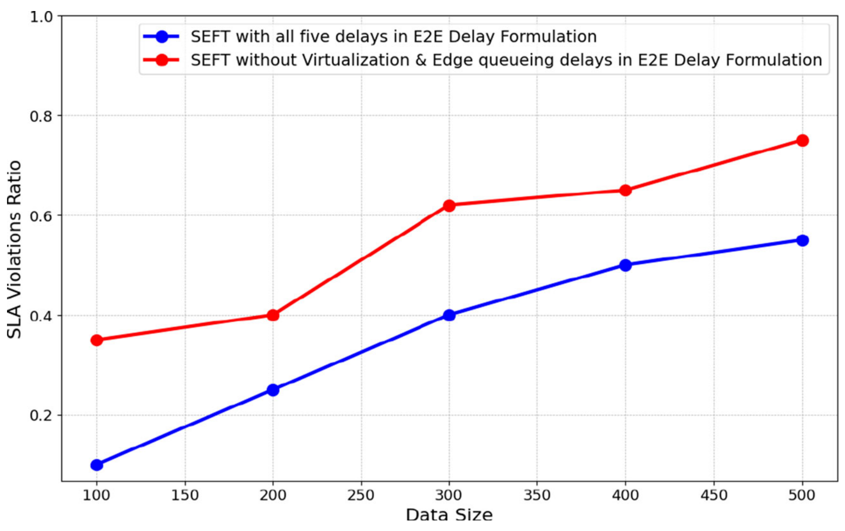


Fig. 3 Impact of Virtualization and Edge queuing delay on SLA Violations ratio while varying data size

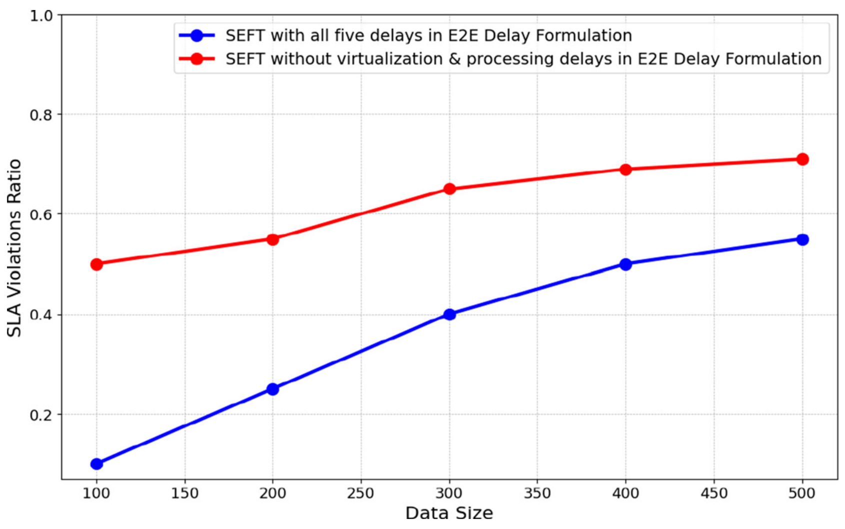


Fig. 4 Impact of Virtualization and Processing delay on SLA Violations ratio while varying data size

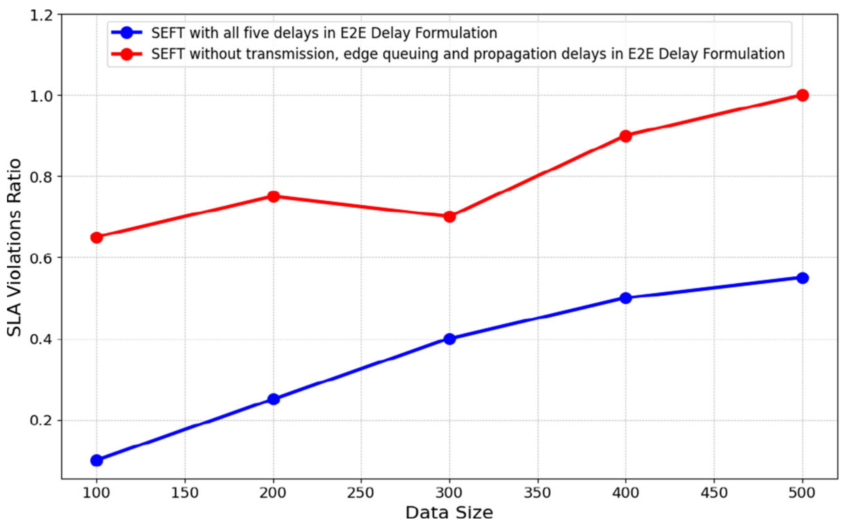
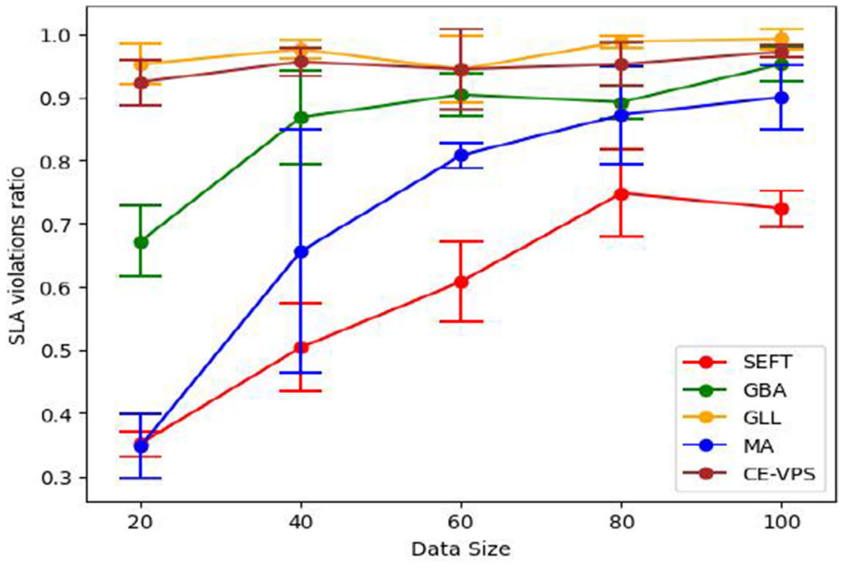
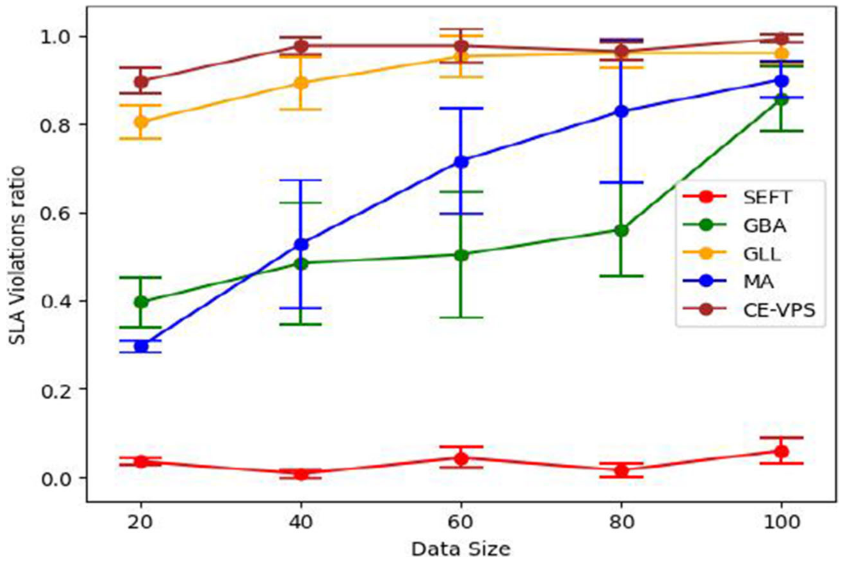


Fig. 5 Impact of transmission, edge queuing, and propagation delay on SLA Violations ratio while varying data size

Fig. 6 Variation of SLA violation ratio with data size. (a) AT&T Network. (b) Iris Network

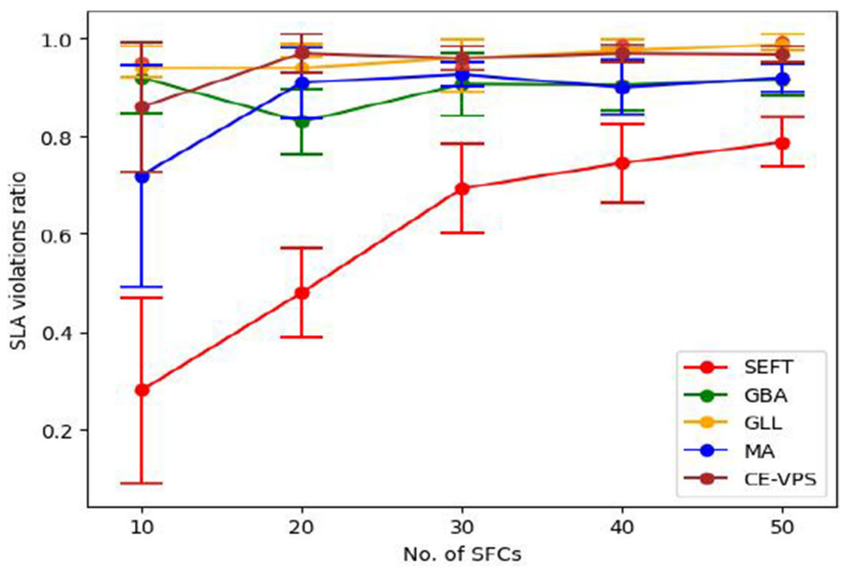


(a)

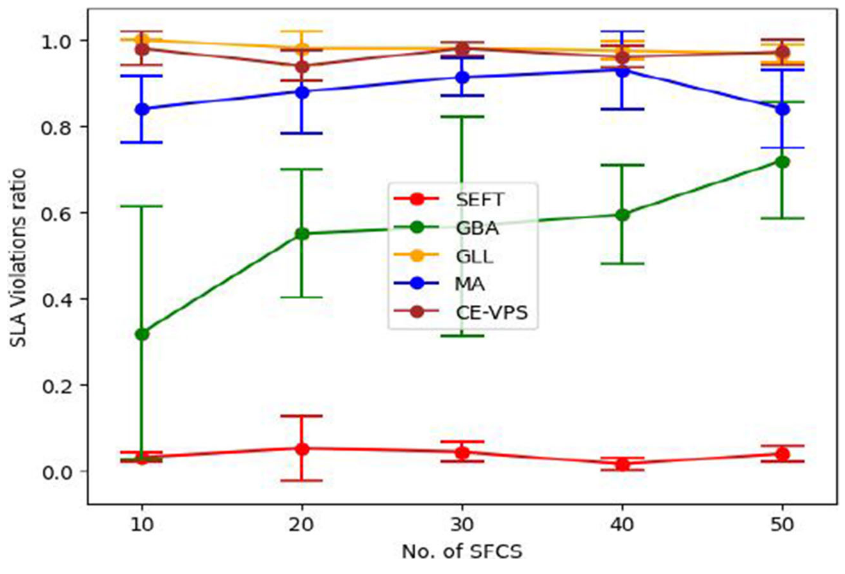


(b)

they also result in a higher end-to-end delay and SLA violation ratio. On the contrary, the proposed approach prefers physical machines with the minimum earliest finish time by considering both the processing delay and the delay required to transfer the data among the physical machines. Hence, it results in the minimum end-to-end delay and SLA violation ratio. The proposed approach achieves at least 32.93%, 38.26%, 30%, and 37.39% reductions in SLA violation ratio when compared to the GBA, GLL, MA, and CE-VPS approaches, respectively. Figure 7b illustrates the performance of proposed and existing approaches in terms of SLA violation ratio for the iris network. The proposed approach effectively incorporated various delays such as transmission delay, processing delay, propagation delay, queuing delay, and virtualization delay in order to calculate the end-to-end latency. Hence, The proposed approach results in a minimum SLA violation ratio while the number of SFCs varies, whereas existing approaches result in a higher SLA violation ratio with the number of SFC requests. The SLA violation ratio of our proposed approach is at least 43.83%, 37.70%, and 42.67% lower compared to the GLL, MA, and CE-VPS approaches, respectively.



(a)



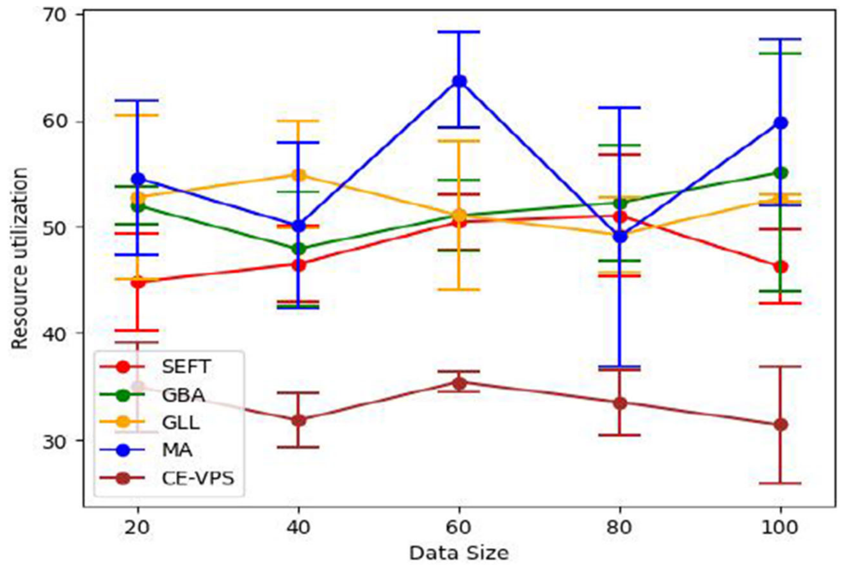
(b)

Fig. 7 Variation of SLA violation ratio with number of SFCs. (a) AT&T Network. (b) Iris Network

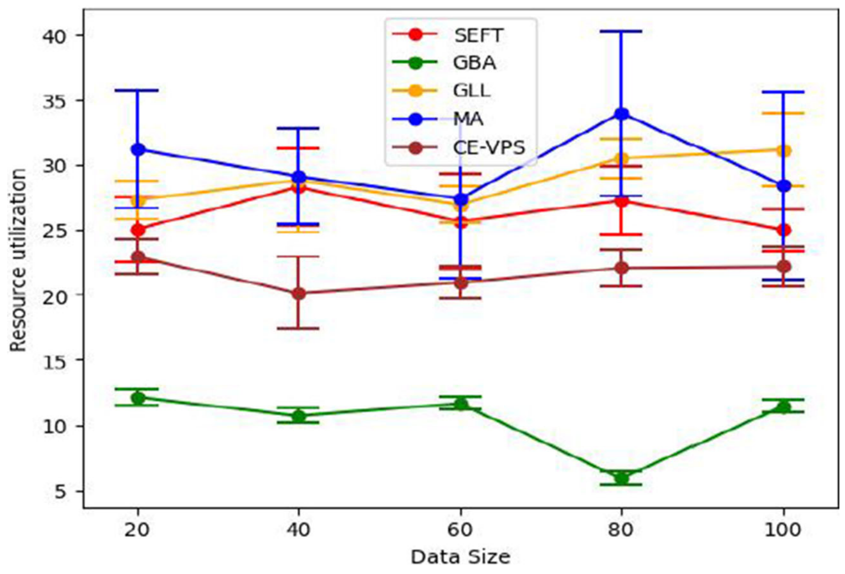
6.2.6 Resource utilization with variation of data size

Resource utilization is the percentage of resources utilized in a network. To evaluate the performance of the proposed approach in terms of resource utilization, the available processing capacity of the PM is considered as the target resource [61]. The available processing capacity of the PM is chosen from the range [400, 600, 800, 1000] KBps. Figure 8a represents the performance of our proposed approach and the existing approaches in terms of resource utilization with the variation of data size on AT&T network. The results are plotted with a confidence level of 95%.

This work is not focused on enhancing resource utilization; consequently, it does not achieve the optimal resource utilization when compared to existing algorithms. But, it achieves a significantly satisfactory percentage of resource utilization. The resource utilization of our proposed approach is 29.04% more than CE-VPS approach. GBA, GLL and MA approaches are showing better resource utilization than our proposed approach. Figure 8b represents the performance of our proposed approach and the existing approaches in terms of resource utilization with the variation of data size on iris network. The proposed approach shows 36.67% and 8.31% more resource utilization than CE-VPS and GBA approaches, respectively.



(a)



(b)

Fig. 8 Variation of Resource utilization with data size. (a) AT&T Network. (b) Iris Network

6.2.7 Resource utilization with variation of number of SFC requests

Figure 9a illustrates the resource utilization of our proposed approach and the existing approaches for the AT&T network while varying the number of SFCs. The number of SFCs is chosen from the range [10, 20, 30, 40, 50]. From the figure, we can observe that our proposed approach results in 37.79% and 72.68% more resource utilization than the GBA and CE-VPS approaches, respectively. Next, the performance of all the presented approaches in terms of resource utilization while varying the number of SFCs is represented in Fig. 9b. It shows that the resource utilization of all the approaches increases with the increase in the number of SFCs. The proposed approach also performs better in terms of resource utilization, though this work is not aimed at improving resource utilization.

By optimizing resource utilization, the proposed approach addresses one of the significant factors that can contribute to power savings. More efficient resource utilization can lead to less idle time and more balanced load distribution, which can potentially reduce unnecessary power usage [62]. However, minimizing power consumption is influenced by multiple factors, including the type of hardware resources, the size and nature of workloads, and external environmental conditions such as temperature. While our approach focuses on

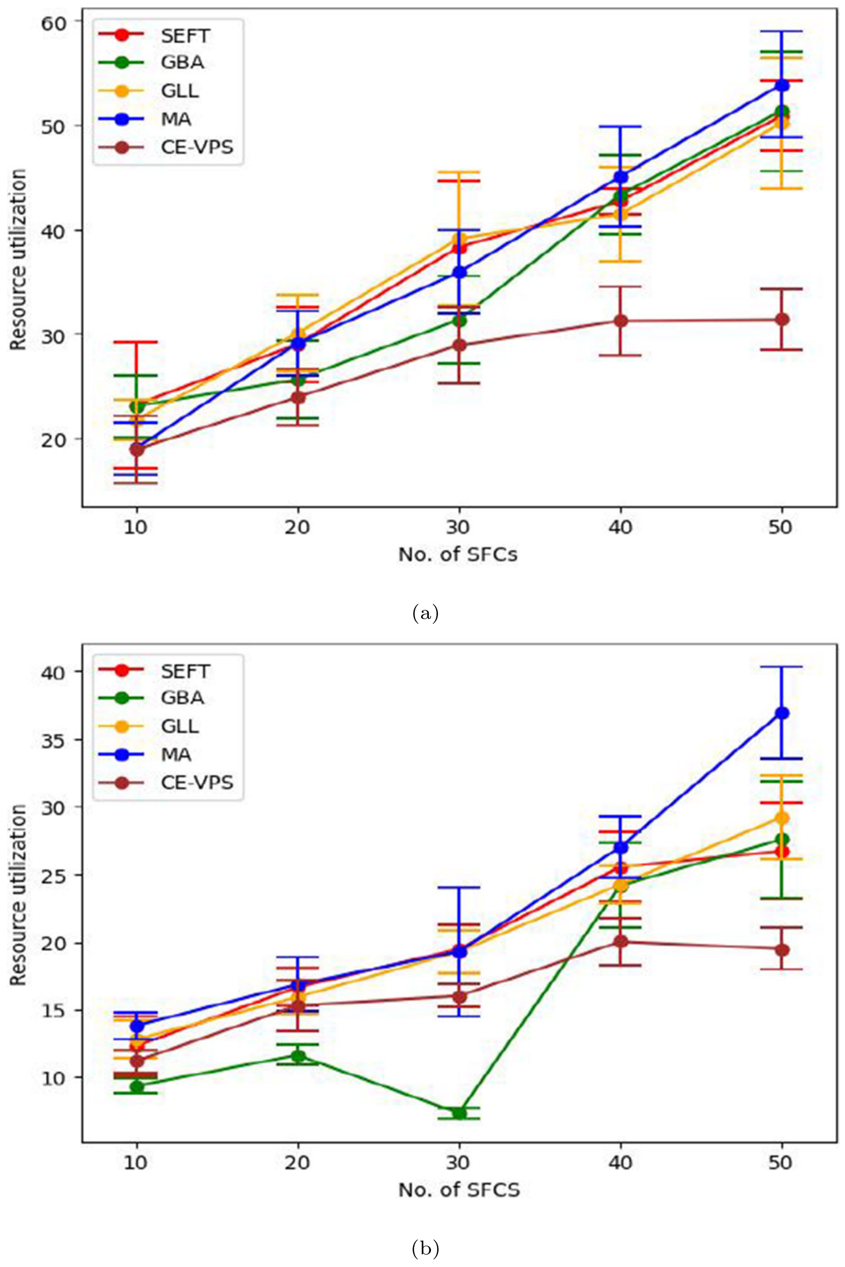


Fig. 9 Variation of Resource utilization with Number of SFCs. (a) AT&T Network. (b) Iris Network

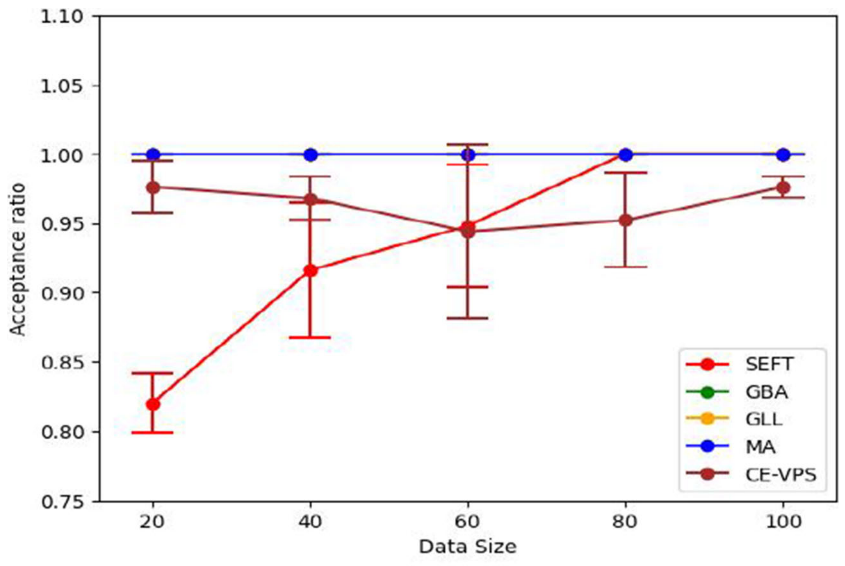
optimizing resource utilization, this alone may not be sufficient to guarantee reduced power consumption under all circumstances.

6.2.8 SFC acceptance ratio

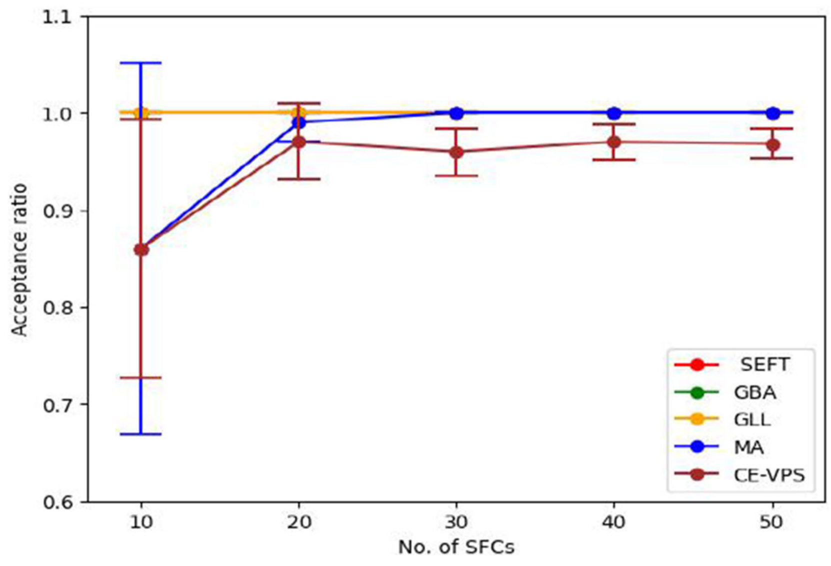
SFC acceptance ratio is the ratio between the number of SFC requests accepted and the total number of SFC requests arrived in a network. Figure 10a represents the performance of proposed SEFT approach and existing approaches in terms of SFC acceptance ratio while varying the data size on the AT & T network. Figure 10b represent the comparison of the proposed approach to the existing approaches in terms of SFC acceptance ratio while varying the number of SFCs. This study is not primarily focused on improving the acceptance ratio, but it is able to show better acceptance ratio when compared to the existing approaches.

6.2.9 Running time

The running time of the proposed approach is compared with the existing approaches as shown in Fig. 11 on the AT&T network. The figure represents the performance of the proposed approach in terms of running time while varying the number of SFC requests. The running time of all the scheduling approaches increase along with increase in the number of



(a)



(b)

Fig. 10 For AT&T network. (a) Variation of acceptance ratio with data size. (b) Variation of acceptance ratio with Number of SFCs

SFC requests, as shown in Fig. 11. Our analysis indicates that the proposed approach takes 480 seconds to schedule 50 SFC requests. In comparison, the GBA, GLL, CE-VPS, and MA approaches take 380, 390, 540, and 600 seconds, respectively to schedule the same number of SFC requests. This comparison helps to understand the running time of our proposed approach relative to established methods.

7 Conclusion

In this paper, we formulated the end-to-end delay-aware SFC mapping and scheduling problem as an integer non-linear programming problem (INLP), while considering placement, capacity, flow, and scheduling constraints. We proposed an algorithm that selects VNFs from a given SFC and schedules them on physical machines with minimum finish times. The performance of the proposed approach is compared with GBA, GLL, MA and CE-VPS approaches in terms of end-to-end delay, SLA violation ratio, resource utilization, and acceptance ratio. Simulation results proved that the proposed approach (SEFT) achieved at least a 26.31%, 51.04%, 14.97%, and 63.05% lower average end-to-end delay with variation of data size when compared to the GBA, GLL, MA and CE-VPS approaches, respectively. The proposed SEFT approach also achieved at least a 38.64%,

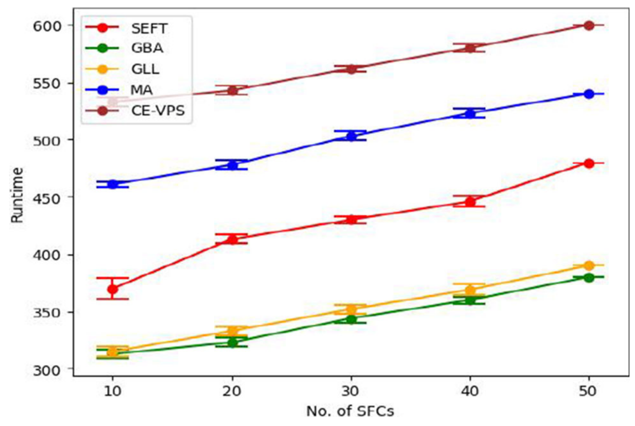


Fig. 11 Comparison of running time with variation of number of SFCs on AT&T Network

67.18%, 51.92%, and 86.99% reduction in average end-to-end delay with variation of number of SFCs over the GBA, GLL, and MA approaches, respectively. Further, our proposed approach resulted in at least a 32.21%, 39.63%, 12.72% and 38.30% lower SLA violation ratios with variation in data size when compared to the GBA, GLL, MA and CE-VPS approaches, respectively. Significant performance improvement is observed in terms of SLA violation ratio with variation in the number of SFCs, resource utilization, and acceptance ratio.

Author Contributions All authors contributed equally to this work.

Funding Not applicable

Data Availability No datasets were generated or analysed during the current study.

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

Competing interests The authors declare no competing interests.

Ethics approval and consent to participate All authors have participated in this study and all ethics have been taken into consideration.

References

1. Jin P, Fei X, Zhang Q, Liu F, Li B (2020) Latency-aware vnf chain deployment with efficient resource reuse at network edge. In: IEEE INFOCOM 2020-IEEE conference on computer communications, IEEE, pp 267–276
2. Thiruvasagam PK, Chakraborty A, Murthy CSR (2021) Latency-aware and survivable mapping of vnfs in 5g network edge cloud. In: 2021 17th International Conference on the Design of Reliable Communication Networks (DRCN), IEEE, pp 1–8
3. Dubba S, Gupta S, Killi B (2024) Predictive resource allocation and vnf deployment using ensemble learning. Multimed Tool Appl:1–26. https://doi.org/10.1007/s11042-024-18673-3
4. Dubba S, Killi B (2023) Energy Efficient Virtual Network Function Placement in NFV Enabled Networks, pp 537–548. https://doi.org/10.1007/978-3-031-28451-9_47
5. Yuan Z, Luo L, Guo D, Wong DC-K, Cheng G, Ren B, Zhang Q (2024) To deploy new or to deploy more?: an online sfc deployment scheme at network edge. IEEE Internet Things J 11(2):2336–2350. https://doi.org/10.1109/JIOT.2023.3293817
6. Cai J, Zhou Z, Huang Z, Dai W, Yu FR (2024) Privacy-preserving deployment mechanism for service function chains across multiple domains. IEEE Trans Netw Serv Manage 21(1):1241–1256. https://doi.org/10.1109/TNSM.2023.3311587
7. Chen M, Sun Y, hu H, Tang L, Fan B (2020) Energy-saving and resource-efficient algorithm for virtual network function placement with network scaling. IEEE Trans Green Commun Netw:1–1. https://doi.org/10.1109/TGCN.2020.3042675
8. Yala L, Frangoudis PA, Ksentini A (2018) Latency and availability driven vnf placement in a mec-nfv environment. In: 2018 IEEE Global Communications Conference (GLOBECOM), IEEE, pp 1–7
9. Cao J, Zhang Y, An W, Chen X, Sun J, Han Y (2017) Vnf-fg design and vnf placement for 5g mobile networks. SCIENCE CHINA Inf Sci 60(4):1–15
10. Lin T, Zhou Z, Tornatore M, Mukherjee B (2016) Demand-aware network function placement. J Lightwave Technol 34(11):2590–2600
11. Subramanya T, Riggio R (2019) Machine learning-driven scaling and placement of virtual network functions at the network edges. In: 2019 IEEE Conference on Network Softwarization (NetSoft), IEEE, pp 414–422
12. Oljira DB, Grinnemo K-J, Taheri J, Brunstrom A (2017) A model for qos-aware vnf placement and provisioning. In: 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), pp 1–7. https://doi.org/10.1109/NFV-SDN.2017.8169829
13. Qi D, Shen S, Wang G (2019) Towards an efficient vnf placement in network function virtualization. Comput Commun 138:81–89. https://doi.org/10.1016/j.comcom.2019.03.005
14. Mohamad A, Hassanein HS (2019) On demonstrating the gain of sfc placement with vnf sharing at the edge. In: 2019 IEEE Global Communications Conference (GLOBECOM), pp 1–6. https://doi.org/10.1109/GLOBECOM38437.2019.9014106
15. Ye Q, Zhuang W, Li X, Rao J (2019) End-to-end delay modeling for embedded vnf chains in 5g core networks. IEEE Internet Things J 6(1):692–704. https://doi.org/10.1109/JIOT.2018.2853708
16. Chintapalli VR, Killi BR, Partani R, Tamma BR, Murthy CSR (2024) Energy- and reliability-aware provisioning of parallelized service function chains with delay guarantees. IEEE Trans Green Commun Netw 8(1):205–223. https://doi.org/10.1109/TGCN.2023.3317927
17. Liu L, Guo S, Liu G, Yang Y (2021) Joint dynamical vnf placement and sfc routing in nfv-enabled sdns. IEEE Trans Netw Serv Manage 18(4):4263–4276. https://doi.org/10.1109/TNSM.2021.3091424
18. He N, Yang S, Li F, Trajanovski S, Zhu L, Wang Y, Fu X (2023) Leveraging deep reinforcement learning with attention mechanism for virtual network function placement and routing. IEEE Trans Parallel Distrib Syst 34(4):1186–1201. https://doi.org/10.1109/TPDS.2023.3240404
19. Xu Y, Kafle VP (2019) An availability-enhanced service function chain placement scheme in network function virtualization. J Sens Actuator Netw 8(2):34
20. Varasteh A, Madiwalar B, Van Bemten A, Kellerer W, Mas-Machuca C (2021) Holu: power-aware and delay-constrained

vnf placement and chaining. IEEE Trans Netw Serv Manage 18(2):1524–1539
21. Wang Y, Huang C-K, Shen S-H, Chiu G-M (2021) Adaptive placement and routing for service function chains with service deadlines. IEEE Trans Netw Serv Manage 18(3):3021–3036
22. Yang S, Li F, Trajanovski S, Chen X, Wang Y, Fu X (2021) Delay-aware virtual network function placement and routing in edge clouds. IEEE Trans Mob Comput 20(2):445–459. https://doi.org/10.1109/TMC.2019.2942306
23. Talpur A, Gurusamy M (2021) Reinforcement learning-based dynamic service placement in vehicular networks. In: 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), IEEE, pp 1–7
24. Sun C, Bi J, Zheng Z, Yu H, Hu H (2017) Nfp: enabling network function parallelism in nfv. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3098822.3098826
25. Pham C, Tran NH, Ren S, Saad W, Hong CS (2017) Traffic-aware and energy-efficient vnf placement for service chaining: Joint sampling and matching approach. IEEE Trans Serv Comput 13(1):172–185
26. Araújo SMA, de Souza FSH, Mateus GR (2021) A hybrid optimization-machine learning approach for the vnf placement and chaining problem. Comput Netw 199:108474. https://doi.org/10.1016/j.comnet.2021.108474
27. Chintapalli VR, Korrapati SB, Adeppady M, Tamma BR, A AF, Killi BR (2023) Nfvpermit: toward ensuring performance isolation in nfv-based systems. IEEE Trans Netw Service Manag 20(2):1717–1732
28. Zhang Y, Zhang F, Tong S, Rezaeipanah A (2022) A dynamic planning model for deploying service functions chain in fog-cloud computing. Journal of King Saud University - Computer and Information Sciences 34(10, Part A):7948–7960. https://doi.org/10.1016/j.jksuci.2022.07.012
29. Cai J, Huang Z, Luo J, Liu Y, Zhao H, Liao L (2020) Composing and deploying parallelized service function chains. J Netw Comput Appl 163:102637. https://doi.org/10.1016/j.jnca.2020.102637
30. Yaghoubpour F, Bakhshi B, Seifi F (2022) End-to-end delay guaranteed service function chain deployment: a multi-level mapping approach. Comput Commun 194:433–445. https://doi.org/10.1016/j.comcom.2022.08.005
31. Khoshkholghi MA, Mahmoodi T (2022) Edge intelligence for service function chain deployment in nfv-enabled networks. Comput Netw 219:109451. https://doi.org/10.1016/j.comnet.2022.109451
32. Miyamura T, Misawa A (2023) Joint optimization of optical path provisioning and vnf placement in vcdn. Opt Switch Netw 49:100740. https://doi.org/10.1016/j.osn.2023.100740
33. Khatiri A, Mirjalily G (2022) A cost-efficient, load-balanced and fragmentation-aware approach for deployment of vnf service chains in elastic optical networks. Comput Commun 188:156–166. https://doi.org/10.1016/j.comcom.2022.03.005
34. Cappanera P, Paganelli F, Paradiso F (2019) Vnf placement for service chaining in a distributed cloud environment with multiple stakeholders. Comput Commun 133:24–40
35. Bu C, Wang J, Wang X (2022) Towards delay-optimized and resource-efficient network function dynamic deployment for vnf service chaining. Appl Soft Comput 120:108711. https://doi.org/10.1016/j.asoc.2022.108711
36. Qu L, Assi C, Shaban K (2016) Network function virtualization scheduling with transmission delay optimization. In: NOMS 2016 - 2016 IEEE/IFIP network operations and management symposium, pp 638–644. https://doi.org/10.1109/NOMS.2016.7502870
37. Riera J, Escalona E, Batalle J, Gras E, García-Espín J (2014) Virtual network function scheduling: Concept and challenges, pp 1–5. https://doi.org/10.1109/SaCoNeT.2014.6867768
38. Mijumbi R, Serrat J, Gorricho J, Bouten N, De Turck F, Davy S (2015) Design and evaluation of algorithms for mapping and scheduling of virtual network functions. https://doi.org/10.13140/RG.2.1.4088.6885
39. Li B, Cheng B, Liu X, Wang M, Yue Y, Chen J (2021) Joint resource optimization and delay-aware virtual network function migration in data center networks. IEEE Trans Netw Serv Manage 18(3):2960–2974. https://doi.org/10.1109/TNSM.2021.3067883
40. Qu L, Assi C, Shaban K (2016) Delay-aware scheduling and resource optimization with network function virtualization. IEEE Trans Commun 64(9):3746–3758. https://doi.org/10.1109/TCOMM.2016.2580150
41. Yang S, Li F, Yahyapour R, Fu X (2022) Delay-sensitive and availability-aware virtual network function scheduling for nfv. IEEE Trans Serv Comput 15(1):188–201. https://doi.org/10.1109/TSC.2019.2927339
42. Qu L, Yu L, Khabbaz M (2023) Latency-sensitive parallel multi-path service flow routing with segmented vnf processing in nfv-enabled networks. IEEE Trans Netw Service Manag:1–13. https://doi.org/10.1109/TNSM.2023.3328644
43. Iwamoto M, Suzuki A, Kobayashi M (2023) Optimal vnf scheduling for minimizing duration of qos degradation. 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC), pp 855–858
44. Gu L, Hu J, Zeng D, Guo S, Jin H (2020) Service function chain deployment and network flow scheduling in geo-distributed data centers. IEEE Transactions on Network Science and Engineering 7(4):2587–2597. https://doi.org/10.1109/TNSE.2020.2997376
45. Li J, Shi W, Wu H, Zhang S, Shen X (2021) Cost-aware dynamic sfc mapping and scheduling in sdn/nfv-enabled space-air-ground integrated networks for internet of vehicles. IEEE Internet Things J
46. Gu L, Zeng D, Li W, Guo S, Zomaya AY, Jin H (2019) Intelligent vnf orchestration and flow scheduling via model-assisted deep reinforcement learning. IEEE J Sel Areas Commun 38(2):279–291
47. Li J, Shi W, Zhang N, Shen X (2021) Delay-aware vnf scheduling: a reinforcement learning approach with variable action set. IEEE Trans Cogn Commun Netw 7(1):304–318. https://doi.org/10.1109/TCCN.2020.2988908
48. Gu L, Zeng D, Tao S, Guo S, Jin H, Zomaya AY, Zhuang W (2019) Fairness-aware dynamic rate control and flow scheduling for network utility maximization in network service chain. IEEE J Sel Areas Commun 37(5):1059–1071
49. Cao H, Du J, Zhao H, Luo DX, Kumar N, Yang L, Yu FR (2021) Resource-ability assisted service function chain embedding and scheduling for 6g networks with virtualization. IEEE Trans Veh Technol 70(4):3846–3859. https://doi.org/10.1109/TVT.2021.3065967
50. Zhang Y, He F, Sato T, Oki E (2020) Network service scheduling with resource sharing and preemption. IEEE Trans Netw Serv Manage 17(2):764–778. https://doi.org/10.1109/TNSM.2019.2956949
51. Chen Y, Wu J (2020) Flow scheduling of service chain processing in a nfv-based network. IEEE Trans Netw Sci Eng 8(1):389–399
52. Gao T, Li X, Wu Y, Zou W, Huang S, Tornatore M, Mukherjee B (2020) Cost-efficient vnf placement and scheduling in public cloud networks. IEEE Trans Commun 68(8):4946–4959
53. Huang Q, Su S, Li J, Xu P, Shuang K, Huang X (2012) Enhanced energy-efficient scheduling for parallel applications in cloud. In: 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), IEEE, pp 781–786
54. Shen G, Li Q, Jiang Y, Wu Y, Lv J (2020) A four-stage adaptive scheduling scheme for service function chain in nfv. Comput Netw 175:107259. https://doi.org/10.1016/j.comnet.2020.107259
55. Oljira DB, Grinnemo K-J, Taheri J, Brunstrom A (2017) A model for qos-aware vnf placement and provisioning. In: 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE, pp 1–7

56. Ghaznavi M, Shahriar N, Kamali S, Ahmed R, Boutaba R (2017) Distributed service function chaining. IEEE J Sel Areas Commun 35(11):2479–2489. https://doi.org/10.1109/JSAC.2017.2760178
57. Knight S, Nguyen HX, Falkner N, Bowden R, Roughan M (2011) The internet topology zoo. IEEE J Sel Areas Commun 29(9):1765–1775. https://doi.org/10.1109/JSAC.2011.111002
58. Beloglazov A, Buyya R (2010) Energy efficient resource management in virtualized cloud data centers. In: 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing, pp 826–831. https://doi.org/10.1109/CCGRID.2010.46
59. Pham C, Tran NH, Hong CS (2018) Virtual network function scheduling: A matching game approach. IEEE Commun Lett 22(1):69–72. https://doi.org/10.1109/LCOMM.2017.2747509
60. Gao T, Li X, Wu Y, Zou W, Huang S, Tornatore M, Mukherjee B (2020) Cost-efficient vnf placement and scheduling in public cloud networks. IEEE Trans Commun 68(8):4946–4959. https://doi.org/10.1109/TCOMM.2020.2992504
61. Xiao Z, Song W, Chen Q (2013) Dynamic resource allocation using virtual machines for cloud computing environment. IEEE Trans Parallel Distrib Syst 24(6):1107–1117. https://doi.org/10.1109/TPDS.2012.283
62. Beloglazov A, Buyya R (2010) Energy efficient resource management in virtualized cloud data centers. In: 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing, IEEE, pp 826–831