Prioritized Deployment of Dynamic Service Function Chains

Behrooz Farkiani, Bahador Bakhshi, S. A. Mirhassani, Tim Wauters, Bruno Volckaert, and Filip De Turck

***Abstract*—Service Function Chaining and Network Function Virtualization are enabling technologies that provide dynamic network services with diverse QoS requirements. Regarding the limited infrastructure resources, service providers may prioritize some service requests over others and even reject some of the low-priority requests to satisfy the requirements of high-priority services. In this paper, we study the problem of deployment and reconfiguration of a set of chains with different priorities with the objective of maximizing the service provider's profit; wherein, we also consider management concerns including the ability to control the migration of virtual functions. We show the problem is more practical and comprehensive than the previous studies, and propose an MILP formulation of it along with two solving algorithms. The first algorithm is a fast polynomial-time heuristic that calculates an initial feasible solution to the problem. The second algorithm is an exact method that utilizes the initial feasible solution to achieve the optimal solution quickly. Using extensive simulations, we evaluate our algorithms and show the proposed heuristic can find a feasible solution in at least 83% of the simulation runs in less than 7 seconds, and the exact algorithm can achieve 25% more profit 8 times faster than the state-of-the-art MILP solving method.**

***Index Terms*—Service function chaining, network function virtualization, priority, exact solution.**

I. Introduction

SERVICE function chaining is an emerging technology that enables service providers to dynamically order service functions, which results in flexible management of network services. Alongside, Network Function Virtualization (NFV) utilizes the virtualization technologies to consolidate various types of network appliances onto standard high-volume off-the-shelf servers. These technologies bring about significant advantages to service providers including reducing the capital and operation expenditures , such as equipment costs and energy bill, and improving the flexibility and innovation in the offered services. One of the key challenges of these technologies is handling diverse QoS requirements of different service via proper chain composition and deployment [1], [2].

Network flows have various QoS requirements, including but not limited to bandwidth and delay [3]. Due to the limited physical resources, flow provisioning should be prioritized: at first, the high priority flows with more strict QoS requirements should be provisioned, and then the low priority flows use the remaining resources. Here, we assume emergency use cases [4], such as processing the high-resolution output video of a fire department's drone, and define two types of flows: emergency flows with strict bandwidth requirements and best-effort flows. The best-effort flows have flexible bandwidth requirements and only use the remaining resources of the infrastructure after the requirements of the emergency flows are satisfied. We define two categories of chains, namely emergency and best-effort chains, to process the emergency and best-effort flows, respectively. While emergency chains have higher priorities and strict bandwidth requirements, the allocated bandwidth for a best-effort chain is a function of remaining resources and the profit of the service provider who deploys the chains on the physical infrastructure.

In the described settings, the main problem is how to allocate processing, memory, and bandwidth resources to the chains to maximize the total profit of the service provider. The service provider also must consider changes in the number of chains and their bandwidth requirements: the input traffic to the chains can vary over time, or new chains may arrive. There are also management concerns, including defining a set of the permitted locations to deploy each function and the possibility of migrating deployed functions. Therefore, the question that this paper answers is "how to deploy or reconfigure the deployment of a set of emergency and best-effort chains to maximize the service provider's profit while satisfying the bandwidth and management requirements?" To answer this question, we first propose an MILP formulation of the problem named Prioritized Service Function Chain Deployment (**PSFCD**) problem. Then, we propose two algorithms to solve the PSFCD efficiently. Although a great deal of research has been conducted on resource allocation and SFC deployment problems in NFV environments [5], [6], to the best of our knowledge, we are the first in investigating the PSFCD problem.

A. Contributions

In this paper, we formulate the PSFCD problem with the following features:

- It can deploy a set of best-effort and emergency chains with the objective of maximizing the total profit. The input traffic to the emergency chains must be satisfied, and the best-effort chains can use the remaining resources.
- It allows the service provider to define a set of candidate servers to deploy each function.

B. Farkiani and B. Bakhshi are with Department of Computer Engineering, Amirkabir University of Technology (Tehran Polytechnic), Iran (e-mail: behrooz.farkiani@aut.ac.ir, bbakhshi@aut.ac.ir)

S. A. Mirhassani. is with Department of Mathematics and Computer Science, Amirkabir University of Technology (Tehran Polytechnic), Iran (e-mail: a_mirhassani@aut.ac.ir)

T. Wauters, B. Volckaert, and F. De Turck are with IDLab, Department of Information Technology Ghent University, Belgium (e-mail: tim.wauters@ugent.be, bruno.volckaert@intec.ugent.be, filip.deturck@ugent.be)

- It allows the service provider to define whether it is possible to migrate a function or not.
- It considers the limited processing, memory, and bandwidth resources of the infrastructure.
- It considers the limited traffic processing capability of VNF instances and utilize horizontal scaling to handle the input traffic.

We assume the service provider requires an optimal solution to the PSFCD problem in the shortest possible time. In order to achieve this goal, we propose two algorithms and evaluate them by using extensive simulations. The first algorithm is a fast polynomial-time heuristic that can calculate a feasible solution for large instances of the PSFCD problem in less than 7 seconds, with a success ratio of at least 83%. The obtained solution can be used as an initial solution in other algorithms. To achieve the optimality, we propose an exact algorithm that uses decomposition and achieves the optimality over a number of limited iterations. The results show the combination of the exact algorithm and the initial heuristic can outperform the state-of-the-art MILP solving algorithm with achieving 25% more profit 8 times faster.

In summary, the major contributions of this paper are as follows:

- We introduce and formulate the PSFCD problem to deploy and reconfigure a set of best-effort and emergency chains with the objective of maximizing the total profit. To the best of our knowledge, the PSFCD problem is the most comprehensive service function deployment and management problem.
- We develop a fast polynomial-time heuristic algorithm to calculate a feasible solution to the PSFCD problem.
- We propose an exact algorithm that can calculate the optimal solution to the PSFCD problem.
- By using extensive simulations, we show the proposed algorithms can outperform the state-of-the-art MILP solving methods.

B. Paper Structure

The rest of this paper is organized as follows. In Section II, the most relevant research is discussed. Section III presents the formulation of the PSFCD problem, and Section IV describes the proposed algorithms for solving it. The performance of the algorithms is evaluated in Section V. Finally, Section VI concludes this paper.

II. Related Research

The SFC deployment problem and its variations have gained a lot of attention during the past few years [5], [6], [15]. In this section, we briefly review and compare the most relevant research. A summary of the comparison is shown in Table I, which is done based on the following criteria. We consider the objective function of the defined problems and whether changes in the requested services and admission control are considered or not. We also consider the assumptions used in the problem definition. More precisely, we focus on the following aspects: the first one is whether the limited traffic processing capacity (throughput) of the VNF instance is taken into account or not. Then, whether the input rates to the chains are considered to be predefined values or can be treated as variables. Moreover, the SFC deployment problems may assume that the input traffic to a chain is limited in a way not to exceed the processing capacity of VNF instances, which is denoted by "Limited" and "Not limited" in Table I. If the input traffic to an abstract function exceeds the throughput of one instance of the corresponding VNF, it is necessary to instantiate more instances and distribute the load between them. However, it is possible to consider vertical scaling or other assumptions to tackle the situation; "Other assumptions" in Table I refers to these assumptions. Finally, we investigate whether it is possible to reject requested chains or not.

A VNF deployment problem to minimize the overall network OPEX and physical resource fragmentation was studied in [16]. Kuo et al. in [11] dealt with the joint VNF placement and path selection problem with the objective of maximizing the sum rate of the admitted demands. The authors proved that finding a non-trivial feasible solution to the problem is NP-Hard. The provisioning of service chains with the objective of minimizing the bandwidth requirement was studied in [12]. Jang et al. [13] assumed the input traffic was variable and designed an MILP problem to deploy a set of requests with the objective of maximizing the acceptable flow rate and minimizing the energy cost. Sun et al. [17] focused on an online and energy-efficient deployment of SFC requests. The authors in [18] studied the problem of VNF sharing with the objective of minimizing the cost for the mobile operator. Farkiani et al. investigated the energy-aware SFC deployment problem in [14] with the objective of minimizing the total energy consumption of the infrastructure servers and switches. The authors also proposed a polynomial-time solvable algorithm to calculate near-optimal solutions quickly.

The above-mentioned studies did not consider dynamic changes in the input traffic or the arrival of new requests. In [7], the authors proposed an MILP problem with the objective of minimizing the average link utilization and the number of utilized servers in the provisioning of new requests. Moreover, the authors also considered traffic changes and used the same problem with a modified objective function to minimize the server and link changes. To deploy new chains and readjust in-service requests for moving users, in [8] an MILP problem with the objective of maximizing the service provider's profit was proposed. The authors in [9] investigated two problems: a VNF-placement problem and a resource-consolidation/de-consolidation problem. The former dealt with the deployment of requests by considering the peak traffic with the objective of maximizing the amount of data that can be processed, and the latter utilized the live instance migration and vertical scaling to reconfigure and migrate the deployed instances when traffic changes. The objective function of the second problem was minimizing the sum of energy consumption costs and revenue loss due to migration. A similar problem was studied in [19] that considered cold migration. In [10], the authors considered changes in the input traffic and showed that better resource utilization could be achieved by dynamically scaling in or out VNF instances. The authors proposed a traffic forecasting method along with two dynamic VNF scaling algorithms for

TABLE I: Summary of related research

Ref.	Admission control	Input traffic					Changes		Instance throughput	
		Variable	Fixed parameter	Limited	Not limited	Other assumptions	Number of chains	Input traffic	Limited	Not limited
[7]			X			X	X	X		X
[8]	X		X	X			X		X	
[9]	X		X			X		X		X
[10]			X		X			X	X	
[11]	X		X	X					X	
[12]			X			X				X
[13]		X			X				X	
[14]			X		X				X	
This paper	X	X	X		X		X	X	X	

two problems with different objectives. In the first problem, it was assumed that a whole chain was placed inside a rack; therefore, the objective function was minimizing the number of VNF instances. In the other problem, it was assumed each rack could only host one type of VNF, and the objective function was minimizing the amount of traffic carried between racks. The authors in [20] proposed a proactive approach to provision new VNF instances ahead of time, based on estimated flow rates. In [21], a VNF migration problem was investigated, and the authors proposed a method to predict resource requirements. The objective was minimizing the sum of migration and bandwidth overhead, which depended on the migrated memory. The authors in [22] proposed an online learning algorithm to predict the incoming rate to chains along with a problem to manage the deployment of the VNF instances with the objective of minimizing the total operational expenditure. In [23], the authors utilized both vertical and horizontal scaling and designed a problem with the objective of maximizing the number of accepted requests under a predefined limited budget.

Table I compares the most important studies based on the described criteria. This paper has the following differences with previous studies regarding the problem definition:

- We consider prioritization in the deployment of network service and define two categories of SFCs, namely best-effort and emergency chains.
- In the PSFCD, we consider initial deployment, migration, modification of input traffic to the chains, and admission control in both deployment and migration of network services, which makes it a comprehensive problem.
- We consider real-world assumptions and provide a formulation that is 1) easy for the operator to work with it, and 2) the obtained solution is easy to implement.

III. Problem Formulation

In this paper, we investigate the problem of deploying a set of best-effort and emergency SFC requests with the objective of maximizing the total profit. First, the assumptions and models of the infrastructure network, the SFC requests, and the provider's profit are discussed. Then, we present the formulation of the Prioritized SFC Deployment problem (PSFCD). The set of all notations used in this paper is shown in Tables II and III.

A. Infrastructure Model

The infrastructure is comprised of servers and switches and modeled as a directed graph $G = (V, E)$. All infrastructure links are bidirectional with a limited bandwidth capacity in each direction. Each directed link from node i to node j is shown by pair (i, j). Each infrastructure server has limited processing and memory resources and has one link attached to one switch. We denote the set of servers and switches by V^S and V^N, respectively, and $V = V^S \cup V^N$. To use similar terminology, we represent the set of ingress/egress points of the infrastructure network traffic by a set of dummy servers Dum. Dummy servers are always on and have infinite resources. Traffic from these points enters the infrastructure to be processed by the chains, and the processed traffic from the chains flows to these points.

B. SFC Requests Model

Each best-effort or emergency chain q is modeled by a linear SFC graph $G^{V,q} = \left(V^{V,q}, E^{V,q}\right)$, wherein $V^{V,q}$ contains a source, a destination, and a set of unique abstract functions such as NAT and firewall, and $E^{V,q}$ contains the links between functions. The source and destination nodes are dummy functions, which are located on the dummy servers, and respectively represent the ingress and egress points of the input traffic to the chain. The set of sources and destinations of all chains are represented by Src and Dst sets, respectively. We denote the function u of chain q by u_q. The input traffic to a best-effort chain q, represented by Θ^q, is a variable that is determined by the solution of the PSFCD problem, and it must be in a range defined by the requesting user. On the other hand, the input traffic to an emergency chain, represented by $\dot{\Theta}^q$, is a given parameter that must be satisfied by the solution. Similar to [13], we assume there is a traffic modification ratio, denoted by $\omega^{q,u}$, assigned with each function, which determines the change in the input traffic rate after being processed by the function. For example, if the modification ratio of a firewall is 1.2, the ratio of the output traffic rate to the input traffic rate will be 1.2 at maximum. We assume there is a set of candidate or preferred locations to deploy each function of the requested chains.

C. Assumptions

We assume that at any given time, there is a set Q of SFC requests that contains both new and previously deployed best-

TABLE II: Notations: Parameters

Parameters			
Dum	Set of ingress/egress points of the infrastructure network	$\omega^{q,u}$	Traffic modification ratio of function u of chain q
$\mathcal{Q} = \{1, .., \lvert Q\rvert\}$	The set of SFCs	$\dot{\Theta}^q$	Input traffic to emergency chain q (Mb/s)
$\mathcal{Q} = B \cup M$	B: best-effort chains M: emergency chains	$\dot{\phi}^{q,u} = \dot{\Theta}^q \times \prod_{i=1}^{u-1} \omega^{q,i}$	Input traffic to function u of emergency chain q (Mb/s)
$G^{V,q} = \left(V^{V,q}, E^{V,q}\right)$	Graph of chain q, including the source and destination nodes	$\theta_{q,UB}$, $\theta_{q,LB}$	Upper and lower bounds on the input traffic to best-effort chain q
$\mathcal{F} = \{1, 2, \ldots, \lvert F\rvert\}$	Set of VNF types such as firewall and NAT	$\varsigma'^{q,u}$	The current number of instances for function u_q
$VNFs$	The set of all abstract functions u_q of all chains	$ACPU^i/AMEM^i$	Available vCPU/MEM resources on server i
$\eta_i^{q,u} \in \{1, 0\}$	Equals to 1 if it is permitted to deploy function u_q on server i	$ICPU^i$	Installed number of vCPUs on server i
$\Im^{q,u}$	The amount of traffic that an instance of VNF u of chain q can process (Mb/s) based on its template	$xC_i^{q,u} \in \{1, 0\}$	The current placement of function u_q
$CPU^{q,u}/MEM^{q,u}$	The required amount of vCPU/MEM resources to create an instance of VNF u of chain q based on its template	P_{mig}	Additional power consumption for migration
$UMEM^{q,u}$	The utilized memory of function u of chain q based on the total utilized memory of its VNF instances	BW_{mig}	Dedicated bandwidth for migration
P_{max}^i	The maximum power consumption of server i	$ABW_{(i,j)}$	Available bandwidth on link (i, j)
P_{idle}^i	The idle power consumption of server i	$\pi^{q,u}$	If 1, migration is permitted for function u_q
$State^i \in \{1, 0\}$	Current on/off state of server i	Δ	Considered duration (in hours) in calculating the energy price
$\mathcal{E}$	Energy price (kWh)	ψ^q	Price of processing 1 Mb of incoming traffic to chain q

TABLE III: Notations: Variables

Variables	
$x_i^{q,u} \in \{1, 0\}$	Placement of abstract function u_q on server i
$\varsigma_i^{q,u} \in \mathbb{Z}_{\geq 0}$	Number of instances of function u_q on server i
$y_{(i,j)}^{q,(u,v)} \in \mathbb{R}_{\geq 0}$	The allocated bandwidth of the link (u, v) of SFC q over the physical link (i, j)
$\phi^{q,u} = \Theta^q \times \prod_{i=1}^{u-1} \omega^{q,i}$	Input traffic to the function u of best-effort chain q (Mb/s)
$\Theta^q \in \mathbb{R}_{\geq 0}$	Input traffic to best-effort chain q (Mb/s)
$CR_i^{q,u}$/ $REM_i^{q,u} \in \mathbb{Z}_{\geq 0}$	Number of created/removed instances of function u_q on server i
$m^{q,u} \in \{1, 0\}$	If 1, the function u_q will migrate to another server
$\alpha^q \in \{1, 0\}$	Must be equal to 1 to accept chain q, otherwise 0
$\beta_i \in \{1, 0\}$	If 1, server i must be on
$LinQX_i^{q,u} = \Theta^q \times x_i^{q,u}$	Linearization variables

effort and emergency requests. It is possible that some chains of set Q have different input rates from their current deployment. To deploy each chain, the service provider must determine the placement of all abstract functions and the virtual links between them. We assume there is a VNF corresponding to each abstract function; for example, there will be a firewall VNF corresponding to the function firewall. The set of all VNFs is shown by $VNFs$.

To deploy an abstract function, a sufficient number of VNF instances must be created to process the input traffic to the function. To instantiate from a VNF, the service provider uses a VNF template that determines the required processing and memory requirements to process a specified and limited volume of traffic, known as the instance throughput. If the input traffic rate to an abstract function is larger than the instance throughput, the service provider must create more than one instance and distribute the input traffic between them. In this condition, we assume all of the created instances must be deployed on the same server to avoid the complexity of network-wide load balancing among the instances of the function. We also assume there is only one template for each VNF type. For example, all of the firewall instances have the same specifications. For each best-effort chain, the service provider must decide whether to accept the request or not. Moreover, if a best-effort request is accepted, the input traffic to it must be decided according to the requested ranges. We assume all emergency chains must be deployed, and there are enough resources to deploy them. We also utilize the multi-commodity flow formulation in the deployment of virtual links. Therefore, each virtual link can be mapped on multiple physical paths of the infrastructure network. In addition, we assume a virtual link cannot completely be placed inside a server, i.e., each end of a virtual link must be placed on different servers. This is because it requires the service provider to measure the inter-VM throughput of each server, which is not practical.

For each abstract function, a set of preferred or candidate servers to deploy the function is determined by the service provider or requesting user. In order to achieve this, parameter $\eta_i^{q,u}$ must be set to 1 to allow the function u of chain q to be placed on server i. By using this parameter, the service provider can force a function to be placed on a specific server, which facilitates the management. In addition, the number of chains and their input traffic may change over time. Therefore, it is necessary to allow the deployed instances to migrate from a server to another server. The service provider can control the migration of the abstract functions by using parameter $\pi^{q,u}$. If this parameter is set to zero, the deployed function stays at the same place. Using this parameter, the service provider can avoid the burden of reconfiguring instances, and it can reduce

the problem solution time. The PSFCD deploys chains without sharing the instances of similar functions between them; since we assumed all instances of a function reside on the same server, migration of a shared instance requires transferring all instances of all functions that are using the shared instance to the same physical server. To avoid the complexity of this scenario, we chose not to share VNF instances between different chains. A similar assumption was made in previous studies [16]. Finally, We assume that there is a reserved amount of bandwidth in the infrastructure network for transferring the memory images of the migrated VNF instances.

D. Provider's Profit Model

The service provider deploys the set of requested chains with the objective of maximizing the total profit. In recent years, energy efficiency has been considered as one of the main concerns of the data center operators, and even in some cases, the energy costs exceeded the cost of purchasing hardware [24]. Consequently, in this paper, we assume the total profit consists of three terms: 1) the income from accepted emergency and best-effort chains, 2) the energy costs due to the energy consumption of servers, and 3) the energy costs due to migrations. We define the income from a chain as a function of the input traffic to the chain and the considered duration. More specifically, the total income is defined as follows:

$$\left(\sum_{q\in\mathcal{Q}}\left(\Theta^q+\dot{\Theta}^q\right)\times\psi^q\times\Delta\right)$$

where, ψ^q is the price of processing 1 unit of incoming traffic to chain q, and Δ is the considered duration.

We utilize a well-known power consumption model for servers which was broadly used in previous research [24], [25] as follows.

$$P_{Server}=P^i_{idle}+\left(P^i_{max}-P^i_{idle}\right)\times\frac{(\#\ allocated\ vCPUs)}{(\#\ installed\ vCPUs)}$$

The above equation formulates the power consumption of an arbitrary server i based on its allocated vCPUs.

Finally, to formulate the energy consumption due to migration, we consider the live migration technique and use the results in [26]. Strunk and Dargie in [26] discovered that the power consumption during migration is about 63% more than the idle state. Accordingly, we use the following equation to consider migration cost:

$$\left(P^{src}_{mig}+P^{dst}_{mig}\right)\times t_{mig}$$

where P^{src}_{mig} and P^{dst}_{mig} are the additional power consumption at the source and destination nodes during migration, and t_{mig} is the migration duration. As stated in [26], the migration duration increases as utilized memory increases and decreases as the available bandwidth for migration increases. Since the exact value of additional power consumption varies with server architectures and vendors, we use parameter P_{mig} to represent both source and destination additional power consumption. Therefore, the following equation can be used to calculate the consumed energy due to migration of function u of chain q:

$$\left(\frac{UMEM^{q,u}}{BW_{mig}}\times 2P_{mig}\right)\times\mathcal{E}$$

If it is not necessary to migrate the memory of a function, for example a stateless function, the corresponding utilized memory $UMEM^{q,u}$ can be set to zero. Here we assume the utilized memory of each function equals the total reserved memory of the corresponding instances. More precisely, when we say a function migrates from a server, it means all instances of the function on that server migrate to the destination server. Therefore, we use the following equation in the formulation of the PSFCD:

$$\left(\varsigma'^{q,u}\times\frac{MEM^{q,u}}{BW_{mig}}\times 2P_{mig}\right)\times\mathcal{E}$$

E. PSFCD Formulation

The formulation of the PSFCD problem is presented in Appendix A. As previously discussed in Section III-D, the objective function maximizes the total profit of the service provider. Constraint (1) enforces that all emergency chains must be deployed while Constraint (2) controls the admission of the best-effort chains. Constraints (3) and (4) control the number of instances to be adequate to process the input traffic to the abstract functions by taking the limited traffic processing capacity of the VNF instances into account. Since PSFCD considers changes in the input traffic and migration, the currently deployed instances and possible migrations must be considered in calculating the required number of instances, which is the role of Constraint (5). Constraints (6)-(9) enable the service provider to manage the changes in the number of SFCs or changes in the input traffic to the deployed chains; for emergency chains, the deployed functions can migrate, and for best-effort chains, it is possible to migrate the deployed functions or even reject some chains. By setting parameter $\pi^{q,u}$ to 1 in Constraint (6), the service provider can allow an already deployed function of an emergency chain to migrate to another server. However, for best-effort chains, the admission control must also be considered along with the migration permission, which is formulated in Constraints (7)-(9). Constraints (10) and (11) respect the limited processing and memory resources of the servers, respectively. Constraint (12) guarantees if the PSFCD decides to deploy a function on a powered-off server, the server will be turned on. Constraint (13) enforces that a virtual link cannot completely be placed inside a server. Flow conservation for infrastructure switches is enforced by Constraint (14) and for infrastructure servers by Constraints (15) and (16). Constraint (17) respects the limited bandwidth of the infrastructure links. Finally, Constraint (18) limits the input traffic rate to each accepted best-effort chain to be in the range defined by the requesting user. It is worth mentioning that the source and the destination node of each chain are already mapped to one of the ingress/egress points of the network, and the presence of them in Constraints (1) and (2) is because of simplicity in the modeling.

Since Constraints (4) and (16) contain a multiplication of two variables $x^{q,u}_i$ and Θ^q, linearization techniques must

be used to convert the PSFCD to an MILP problem. We use the following equations to linearize the above-mentioned constraints. To linearize the input traffic to function u of chain q, we introduce variable $LinQX_i^{q,u} = x_i^{q,u} \times \Theta^q$ and rewrite $x_i^{q,u} \times \phi^{q,u}$ as follows:

$$x_i^{q,u} \times \phi^{q,u} = x_i^{q,u} \times \Theta^q \times \prod_{i=1}^{u-1} \omega^{q,i} = LinQX_i^{q,u} \times \prod_{i=1}^{u-1} \omega^{q,i}$$

We add the following constraint to the PSFCD to define the relation between $LinQX_i^{q,u}$ and other variables:

$$LinQX_i^{q,u} \leq \Theta^q \quad \forall i \in V^S \cup Dum,\ q \in B,\ u_q \in VNFs \cup \{Src, Dst\} \quad \textbf{(19)}$$

$$LinQX_i^{q,u} \leq \theta_{q,UB} \times x_i^{q,u} \quad \forall i \in V^S \cup Dum,\ q \in B,\ u_q \in VNFs \cup \{Src, Dst\} \quad \textbf{(20)}$$

$$LinQX_i^{q,u} \geq \Theta^q - \left(1 - x_i^{q,u}\right) \theta_{q,UB} \quad \forall i \in V^S \cup Dum,\ q \in B,\ u_q \in VNFs \cup \{Src, Dst\} \quad \textbf{(21)}$$

$$LinQX_i^{q,u} \in \mathbb{R}_{\geq 0} \quad \forall i \in V^S \cup Dum,\ q \in B,\ u_q \in VNFs \cup \{Src, Dst\}$$

The PSFCD problem and its modeling are independent of time; the decision about when the model should be executed is completely dependent on the operator's condition. Indeed, the operator can execute the problem whenever they want to initialize the network, deploy new chains, or manage and re-optimize the current deployment.

F. An Illustrative Example

This section presents an example to clarify the PSFCD problem. The infrastructure is illustrated in Figure 1, and the set of SFCs are shown in Table IV. Chains 0 to 3 are best-effort and chains 4 and 5 are emergency chains. A green circle above a server denotes the server is on, and a red circle means the server is off. The specifications of the servers are according to Table VIII of the simulation section. Servers 1, 2, 7, and 8 are of Class 3-Low, and servers 3 to 6 are of Class 2-Low. The available processing units for servers 1, 3, 5, 7 are 40, 20, 20, and 50 vCPUs, and the available memories are 60, 40, 60 and, 100 GB, respectively. The bandwidth of all links equals 40 Gbps in each direction. The problem is solved two times at Time 1 and 2 for the given set of SFC requests. The service provider can deploy requested functions on any server. There is just one exception for the video optimizer function that can only be deployed on server 1 because of the specific hardware. Also, the service provider is not allowed to migrate firewall instances due to high reconfiguration overhead. The source and the destination nodes of each chain point to the ingress point of the infrastructure. The specifications of the VNFs and the rest of the simulation parameters are equal to the parameters described in Tables VII, IX, and X in the simulation section.

At Time 1, the first set of chains arrives, and the service provider uses the PSFCD problem to deploy them. All the chains are accepted at their maximum requested input rate, and server 6 is turned on. The total profit equals 81668.71$. The placement is shown in Table V using the tuples of the form (x, y, z) in which x represents the chain number, y represents the function number, and z denotes the number of deployed instances. For example, $(0, 1, 8)$ means 8 instances are created to process the input traffic to the firewall function of chain 0.

TABLE IV: The set of requested SFCs

Time	Chain		Input (Mbps)
1	0	Src->FW->IDS->Dst	[1000-8000]
	1	Src->FW->Router->NAT->Dst	[5000-10000]
	2	Src->FW->NAT->Dst	[1000-10000]
	3	Src->FW->NAT->Dst	[2000-10000]
2	-	Chains 0-3 without any modifications	
	4	Src->FW->Video Optimizer->NAT->Dst	7000
	5	Src->FW->IDS->NAT->Dst	6000

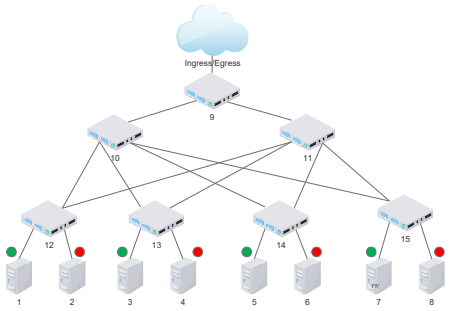


Fig. 1: The topology used in the example

Later, at Time 2, a set of emergency chains arrive. Again, the service provider uses the PSFCD to deploy the new chains and reconfigure the deployment of the first set. Therefore, the inputs to the PSFCD are the set of all (both old and new) demands and also the placement of the current instances of the old chains. The solution to the problem is as follows; the input rate to chain 0 is decreased from 8000 to 7000 Mbps, and chain 3 is rejected. The other chains are accepted with their maximum input traffic. Moreover, function Router of chain 1 migrates to server 6. The total profit equals 97838.3$, and the placement of the functions is presented in Table 4 under the name of the second run. Therefore, the arrival of new emergency chains resulted in a decrease in the input rate to a best-effort chain, migration of a function, and the rejection of a best-effort chain to satisfy the bandwidth requirements of the emergency chains.

Moreover, if the service provider needs to deploy the arriving emergency chains as fast as possible, they can set the migration parameter $\pi^{q,u}$ to zero for the first set of chains. In this case, the solution time decreases and the input traffic rates to all chains remain the same as the second run; However, the solution is suboptimal with a total profit of 97801.37$ because server 4 should be turned on. With this approach, the service provider can handle emergency situations quickly, and after the emergency is resolved, they can re-run the PSFCD problem with enabled migrations to re-optimize the whole infrastructure.

IV. Proposed Solution Methodology

In this section, we propose two approaches to solve the PSFCD problem. First, we consider there are some cases in which the operator needs to deploy emergency chains quickly. To satisfy this requirement, we introduce a fast heuristic algorithm

TABLE V: Placement of functions

Server	Placed Functions	
	First run (before emergency)	Second run (after emergency)
1	(3,1,10)	(4,2,13)
3	(2,2,9)	(2,2,9)
5	(1,2,9)	(5,1,6) (5,3,5)
6	(0,2,15) (3,2,9)	(0,2,13) (1,2,9) (4,1,7) (4,3,4) (5,2,11)
7	(0,1,8) (1,1,10) (1,3,9) (2,1,10)	(0,1,7) (1,1,10) (1,3,9) (2,1,10)

that can find a feasible solution by considering only emergency chains. The obtained solution can be used as an initial solution and be fed to another algorithm to improve it. Second, to maximize the service provider's profit, we introduce the second algorithm that solves the PSFCD problem optimally and faster than the state-of-the-art MILP solving methods. The second algorithm uses the solution obtained from the first algorithm and decomposes the PSFCD into two sub-problems.

A. Finding a Feasible Solution

In this section, we utilize some ideas of the feasibility pump methods [27]–[29] and propose a heuristic, named Fast Initial Heuristic (**FIH**), to calculate an initial feasible solution to the PSFCD problem.

According to Constraints (1) and (2), a feasible solution must deploy all emergency chains and can reject all the best-effort chains. Therefore, first, we consider only the emergency chains and solve the following LP problem:

$$\max \left(\sum_{q\in M} \left(\dot{\Theta}^q\right) \times \psi^q \times \Delta \right)$$
$$- \left(\sum_{q\in M \& u_q \in VNFs} m^{q,u} \times \left(\varsigma'^{q,u} \times \frac{MEM^{q,u}}{BW_{mig}} \times 2P_{mig} \right) \times \mathcal{E} \right)$$
$$- \Big(\sum_{i\in V^S} \left(\beta_i \times {P_{idle}}^i \times \left(1 - State^i\right)\right.$$
$$\left. + \sum_{q\in M \& u_q \in VNFs} \varsigma_i^{q,u} \times CPU^{q,u} \times \frac{P^i_{max} - P^i_{idle}}{ICPU^i} \right) \times \Delta \times \mathcal{E} \Big)$$
$$Constraints\ (1),\ (3),\ (5),\ (6),\ (10-15),\ (17)\ of\ the\ PSFCD$$
$$x_i^{q,u}, m^{q,u}, \beta_i \in [0-1]$$
$$\varsigma_i^{q,u}, y_{(i,j)}^{q,(u,v)}, CR_i^{q,u},\ REM_i^{q,u} \in \mathbb{R}_{\geq 0}$$

This problem is very similar to the PSFCD problem; however, only the emergency chains are present in the formulation, and the best-effort chains are removed.

Then, we use the Round function that executes Algorithm 1 to convert the obtained LP solution to an integer solution. Algorithm 1 processes VNFs according to their order in the chains, adds a random amount to the obtained not-integer values, and calculates $\tilde{x}_i^{q,u}$ as the rounded integer solution. According to the calculated placement, the integer solution for variables $m^{q,u}$, $\varsigma_i^{q,u}$, $CR_i^{q,u}$, $REM_i^{q,u}$ and β_i are also calculated. The random function used in Lines 9 and 11 of Algorithm 1 was already proposed in [27].

If the rounded solution is feasible, we save the solution and exit ; otherwise, we pass the solution to the FallBack mechanism to repair it over a predefined number of iterations. In each iteration of the FallBack mechanism, as illustrated

Algorithm 1 The Round algorithm

1: Save the LP solutions as X^*
2: **for** each chain q **do**
3: **for** each function $u \in V^{V,q}$ **do**
4: List $Placement \leftarrow \emptyset$
5: **for** each $i \in V^S \cup Dum$ if $\eta_i^{q,u} = 1$ **do**
6: **if** function $u-1$ is not already placed on i **then**
7: Generate a random number $r \in [0\ 1]$
8: **if** $(r < 0.5)$ **then**
9: add server i to list $Placement$ with weight ${x^*}_i^{q,u} + 2r\,(1-r)$
10: **else**
11: add server i to list $Placement$ with weight ${x^*}_i^{q,u} + 1 - 2r\,(1-r)$
12: Select server s with the biggest weight from list $Placement$ and place function u on it: $\tilde{x}_s^{q,u} \leftarrow 1$ and $\tilde{x}_i^{q,u} \leftarrow 0 \ \ \forall i \neq s$
13: Calculate $\tilde{m}^{q,u}, \tilde{\varsigma}_i^{q,u}, \widetilde{CR}_i^{q,u}, \widetilde{REM}_i^{q,u}$ *and* $\tilde{\beta}_i$ for all chains.
14: **if** the calculated solution was visited before **then**
15: Return 1.
16: **else**
17: Return 0.

in Figure 2, a different formulation of the PSFCD problem, named PSFCD-FP, is solved. We utilize the idea of binarizing the integer variables, which was shown to be very effective in calculating integer solutions [14]. Consequently, In PSFCD-FP, we substitute the variables $\varsigma_i^{q,u} \in \mathbb{Z}_{\geq 0}$ with their binary representation, i.e., for emergency chains, instead of $\varsigma_i^{q,u}$, we use $\sum_{k=0}^{K^{q,u}} \varsigma_{i,k}^{q,u} \times 2^k$ where $2^{K^{q,u}} \geq \frac{\dot{\phi}^{q,u}}{\Im^{q,u}}$ and $K^{q,u}$ is the smallest integer value that satisfies the equation. Then, we solve the PSFCD-FP problem as follows:

$$\min \sum_{i\in V^S} \Big(\sum_{q\in M \& u_q \in VNFs | \tilde{x}_i^{q,u}=0 \& \eta_i^{q,u}=1} x_i^{q,u} + \sum_{q\in M \& u_q \in VNFs | \tilde{x}_i^{q,u}=1 \& \eta_i^{q,u}=1} \left(1 - x_i^{q,u}\right) \Big)$$
$$x_i^{q,u} \times \dot{\phi}^{q,u} \leq \Im^{q,u} \times \textstyle\sum_{k=1}^{K^{q,u}} \varsigma_{i,k}^{q,u} \times 2^k \quad \forall q \in M, u_q \in VNFs, i \in V^S | \eta_i^{q,u} = 1$$
$$\textstyle\sum_{k=1}^{K^{q,u}} \varsigma_{i,k}^{q,u} \times 2^k = \left(\varsigma'^{q,u} \times \left(xC_i^{q,u} + (1 - xC_i^{q,u}) \times x_i^{q,u}\right)\right) + \left(CR_i^{q,u} - REM_i^{q,u}\right) \ \forall q \in M, u_q \in VNFs,\ i \in V^S$$
$$\sum_{q\in M \& u_q \in VNFs} \left(\textstyle\sum_{k=1}^{K^{q,u}} \varsigma_{i,k}^{q,u} \times 2^k \right) \times CPU^{q,u} \leq ACPU^i \quad \forall i \in V^S$$
$$\sum_{q\in M \& u_q \in VNFs} \left(\textstyle\sum_{k=1}^{K^{q,u}} \varsigma_{i,k}^{q,u} \times 2^k \right) \times MEM^{q,u} \leq AMEM^i \quad \forall i \in V^S$$
$$Constraints\ (1),\ (6),\ (12-15),\ (17)\ of\ the\ PSFCD$$
$$\varsigma_{i,k}^{q,u},\ x_i^{q,u},\ m^{q,u},\ \beta_i \in [0-1]$$
$$y_{(i,j)}^{q,(u,v)},\ CR_i^{q,u},\ REM_i^{q,u} \in \mathbb{R}_{\geq 0}$$

The objective function of the PSFCD-FP problem minimizes

the distance between the current infeasible integer solutions $\tilde{x}_i^{q,u}$ and $x_i^{q,u}$, i.e., it tries to find an LP solution close to the current infeasible solution. A similar objective function was already proposed in [28]. After the PSFCD-FP has been solved, the solution will be given to the Round function shown in Algorithm 1. The Round function checks the calculated placement to see whether it is a new solution or the solution is visited before. If the solution is new, it returns 0; otherwise, the Reboot function is invoked to calculate a random solution.

The Reboot function simply assigns random values between 0 to 1 to $x_i^{q,u}$, sorts these values, and selects the biggest value as placement of the function. Finally, the Reboot function checks whether the calculated solution is a new solution and, if it is, it returns 0. Otherwise, the control is returned to the loop.

At the end of each loop, the feasibility of the last calculated solution is checked. If the solution violates Constraint (10) or (11) of the PSFCD problem, then the solution is infeasible, and we remove it from the solution space. Please note that the Constraints (1) and (13) are already satisfied by Algorithm 1. If both the constraints hold, then we check Constraints (14), (15), and (17) of the PSFCD and, if they hold, the solution is feasible, and we exit the loop.

We can use a cut to remove an infeasible solution from the solution space. Consider a solution that violates any of the Constraints (10) or (11) for a server s and set $u_q \in VNFs\ |\tilde{x}_s^{q,u} = 1$ is shown by u_q^s. Then, to prevent the future placement of functions u_q^s on server s the following constraint can be added to the PSFCD-FP:

$$\sum_{u_q \in u_q^s} (1 - x_s^{q,u}) \geq 1$$

The termination condition of the FallBack mechanism is defined as a combination of reaching a maximum number of iterations or finding an integer feasible solution. If the solution obtained by this algorithm is a feasible solution, it can be fed into another algorithm to improve the solution. The choice of the next algorithm depends on the service provider's priorities. The service provider may prefer to use an exact algorithm to calculate the optimal solution to maximize its profit, or they may want a fast near-optimal solution. This paper considers the former and proposes an exact algorithm that can quickly calculate the optimal solution to the PSFCD problem.

1) Complexity Analysis: In the FIH heuristic, first, we have to solve an LP problem that contains only emergency chains. An LP problem is polynomial-time solvable [30] , and its complexity is $\mathcal{O}(n^{3.5})$ in which n is the total size of variables [13] and [31]. Subsequently, Algorithm 1 iterates for $|VNFs| \times |V^S \cup Dum|$ iterations to calculate an integer solution. In each iteration, it sorts the candidate servers based on their weights. Before exiting the Round function, the values of variables expressed in Line 13 is determined based on Constraints (3), (5), (6), and (12) of the PSFCD problem by substituting the calculated placement.

If the rounded solution is not feasible, we resort to the FallBack mechanism in Figure 2. In each iteration of the FallBack mechanism, the PSFCD-FP is solved, which is an LP problem and can be solved in polynomial time. In addition,

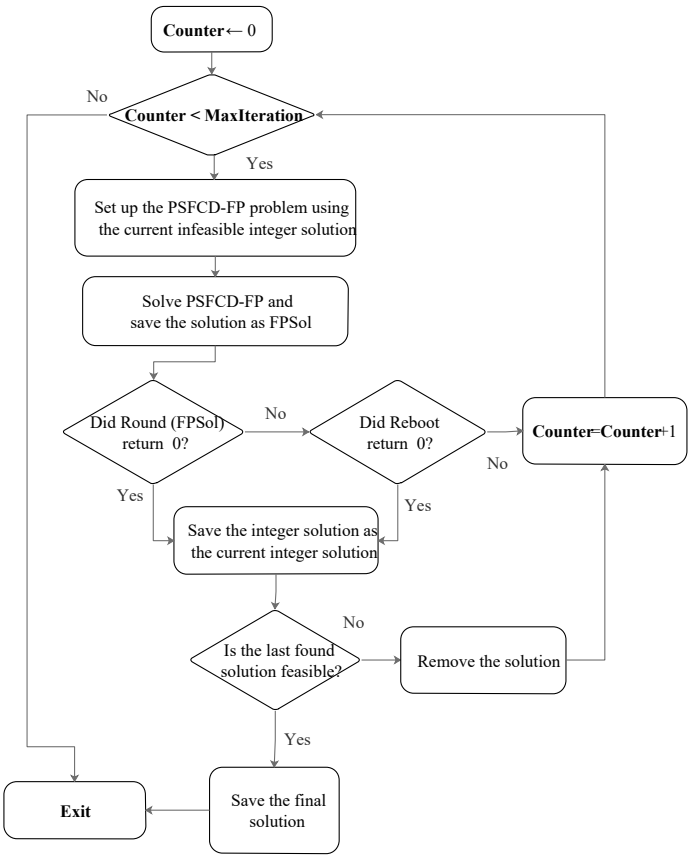


Fig. 2: The FallBack mechanism

after each execution, the Round or Reboot functions are called. The complexity of the Reboot function is less than the Round function since it randomly assigns values to variables without any sorting. To check whether the calculated solution is feasible or not, it is sufficient to first check the limitations of the memory and processing resources of the servers, Constraints (10) and (11) of the PSFCD, are respected or not. Then, the PSFCD-SP problem is solved to check, for a given placement, the placement of virtual links is possible or impossible. The PSFCD-SP is an LP problem and can be solved in polynomial time. Therefore, Since the FallBack mechanism is executed at most **MaxIteration** times, the FIH heuristic is polynomial-time solvable according to the aforementioned explanation.

B. Exact Solving Algorithm

We utilize the idea of the Benders decomposition method [32] and the variation used in [33] to decompose the PSFCD into two subproblems: PSFCD-RMP and PSFCD-SP. Then we use our proposed algorithm to solve the problems iteratively to reach the optimal solution. The flow chart of the proposed exact algorithm, named Accelerated Exact Approach (**AEA**), is displayed in Figure 3. The the PSFCD-RMP decides on the placement of the functions, and the PSFCD-SP uses the obtained placement to calculate the routing of the virtual links regarding the limited capacity of the physical links. We use the objective function of the PSFCD problem for the PSFCD-RMP. Therefore, the PSFCD-SP acts as a feasibility problem without any objective function.

The the PSFCD-SP problem is defined as follows which is a set of constraints without any objective function:

$$\sum_{(i,j)\in E} y_{i,j}^{q,(u,v)} - \sum_{(j,i)\in E} y_{j,i}^{q,(u,v)} = 0 \; \forall i \in V^N, q \in \mathcal{Q}, (u,v) \in E^{V,q}$$

$$\lambda_{(i,j)}^{q,(u,v)} : y_{(i,j)}^{q,(u,v)} - y_{(j,i)}^{q,(u,v)} = \dot{\phi}^{q,v} \times \left(\hat{x}_i^{q,u} \times \eta_i^{q,u} - \hat{x}_i^{q,v} \times \eta_i^{q,v}\right)$$
$$\forall i \in V^S \cup Dum, (i,j) \in E, \; q \in M, \; (u,v) \in E^{V,q}$$

$$\mu_{(i,j)}^{q,(u,v)} : y_{(i,j)}^{q,(u,v)} - y_{(j,i)}^{q,(u,v)} =$$
$$\prod_{i=1}^{v-1} \omega^{q,i} \times \left(\widehat{LinQX}_i^{q,u} \times \eta_i^{q,u} - \widehat{LinQX}_i^{q,v} \times \eta_i^{q,v}\right)$$
$$\forall i \in V^S \cup Dum, (i,j) \in E, \; q \in B, \; (u,v) \in E^{V,q}$$

$$\varpi_{(i,j)} : \sum_{q\in Q} \sum_{(u,v)\in E^{V,q}} y_{(i,j)}^{q,(u,v)} \leq ABW_{(i,j)} \; \forall (i,j) \in E$$

$$y_{(i,j)}^{q,(u,v)} \in \mathbb{R}_{\geq 0}$$

In the PSFCD-SP, $\lambda_{(i,j)}^{q,(u,v)}$, $\mu_{(i,j)}^{q,(u,v)}$ and $\varpi_{(i,j)}$ are the dual variables. $\hat{x}_i^{q,u}$ and $\widehat{LinQX}_i^{q,u}$ are parameters, and their values are directly substituted from the solution to the PSFCD-RMP.

If we denote the objective function of the PSFCD by Z, the PSFCD-RMP is defined as follows:

$$\min -Z$$

$$Restate\ Constraints\ (1)-(3),\ (5)-(13)\ and\ (18)-(21)$$
$$of\ the\ PSFCD\ as\ Constraints\ \textbf{(1)}-\textbf{(16)}$$

$$LinQX_i^{q,u} \times \prod_{i=1}^{u-1} \omega^{q,i} \leq \Im^{q,u} \times \varsigma_i^{q,u}$$
$$\forall\, q \in B, u_q \in VNFs, i \in V^S | \eta_i^{q,u} = 1 \qquad \textbf{(17)}$$

$$\sum_{q\in Q} \sum_{(i,j)\in E} \sum_{(u,v)\in E^{V,q}} \Bigg(\dot{\phi}^{q,v} \times \left(\hat{\lambda}_{(i,j)}^{q,(u,v)}\right)^k \times \left(x_i^{q,u} \times \eta_i^{q,u} - x_i^{q,v} \times \eta_i^{q,v}\right) + \prod_{i=1}^{v-1} \omega^{q,i} \times \left(\hat{\mu}_{(i,j)}^{q,(u,v)}\right)^k \times \left(LinQX_i^{q,u} \times \eta_i^{q,u} - LinQX_i^{q,v} \times \eta_i^{q,v}\right) \Bigg)$$
$$+ \sum_{(i,j)\in E} \left(\hat{\varpi}_{(i,j)}\right)^k \times ABW_{(i,j)} \leq 0 \; \forall k \in K \qquad \textbf{(18)}$$

$$\sum_{q\in M \& u_q \in VNFs} x_i^{q,u} \times \dot{\phi}^{q,u+1} \times \eta_i^{q,u} +$$
$$\sum_{q\in B \& u_q \in VNFs} LinQX_i^{q,u} \times \prod_{i=1}^{u} \omega^{q,i} \times \eta_i^{q,u} \leq ABW_{(i,j)}$$
$$\forall i \in V^S, \; (i,j) \in E \qquad \textbf{(19)}$$

$$\sum_{q\in M \& u_q \in VNFs} x_i^{q,u} \times \dot{\phi}^{q,u} \times \eta_i^{q,u}$$
$$+ \sum_{q\in B \& u_q \in VNFs} LinQX_i^{q,u} \times \prod_{i=1}^{u-1} \omega^{q,i} \times \eta_i^{q,u} \leq ABW_{(j,i)}$$
$$\forall i \in V^S, \; (j,i) \in E \qquad \textbf{(20)}$$

Constraint (17) is the linearization of Constraint (4) of the PSFCD problem. The additional Constraints (18)-(20) of the PSFCD-RMP, in comparison to the PSFCD, are added to improve the decomposition.

If the solution to the PSFCD-RMP is not a feasible solution to the PSFCD, i.e., it violates any of the PSFCD-SP constraints, a feasibility cut in the form of Constraint (18) will be added to the PSFCD-RMP. Constraint (18) is derived from the objective function of the dual problem of the PSFCD-SP as follows. If the PSFCD-SP problem becomes an infeasible problem, after substituting the values of $\hat{x}_i^{q,u}$ and $\widehat{LinQX}_i^{q,v}$, its dual becomes an unbounded problem. Then, we use the extreme directions of the dual problem to create Constraint (18), known as the Benders feasibility cut and add it to the PSFCD-RMP. In this way, we prevent the PSFCD-RMP to find these infeasible solutions in the future. The values of $\left(\hat{\lambda}_{(i,j)}^{q,(u,v)}\right)^k$, $\left(\hat{\mu}_{(i,j)}^{q,(u,v)}\right)^k$, and $\left(\hat{\varpi}_{(i,j)}\right)^k$ represent the extreme direction at iteration k, and set K contains all extreme directions up to the current iteration.

TABLE VI: Different possibilities in solving the subproblems

		PSFCD-RMP	
		Optimal	Sub-optimal
PSFCD-SP	Feasible	Optimal Solution	Depends on the optimality gap
	Infeasible	Remove the solution	Remove the solution

Since there is no connection between the variables of the PSFCD-RMP and PSFCD-SP, it is likely to generate infeasible solutions at first iterations, leading to an increase in the solution time. To remedy this situation, we add Constraints (19) and (20) to the PSFCD-RMP. Constraint (19) states that for each server, the total amount of output traffic from all functions placed on that server must be lower than the available output bandwidth of the server. Constraint (20) states that the total amount of input traffic to all functions must be lower than the input bandwidth of the server. Adding these two constraints improves the solution time significantly.

We set a time limit in the proposed exact algorithm. After solving the subproblems, four possibilities may happen: The PSFCD-RMP reaches the optimal or sub-optimal solution, and after substituting the solution in the PSFCD-SP, it either becomes a feasible or infeasible problem. In each case, we treat the solution in different ways that are depicted in Table VI.

At the first iteration, we use the solution obtained by the proposed heuristic as a warm-start [34] solution to solve the PSFCD-RMP. Using an initial solution significantly enhances the performance of the Benders method [14] [33]. After solving the PSFCD-RMP, we substitute the solution in the PSFCD-SP and solve that problem. If the PSFCD-SP becomes a feasible problem, we check whether the optimality gap is smaller than the predefined value or not. The optimality gap is defined similarly to [35]. If it is smaller, then the algorithm stops; otherwise, the obtained solution is used as a warm-start solution to solve the PSFCD-RMP again. On the other hand, If the PSFCD-SP becomes an infeasible problem, the corresponding extreme direction will be added to set K in Constraint (18).

The termination condition of the proposed exact algorithm is a combination of reaching a global time limit or satisfying the predefined gap, or reaching a predefined number of iterations. As already stated, we also set a time limit for solving the PSFCD-RMP, which is smaller than the global time limit. In the initialization step, we set all these parameters.

V. Numerical Results

In this section, we compare the performance of our proposed exact algorithm, the AEA, with CPLEX Branch & Cut algorithm (B&C) [36], our implementation of the standard Benders decomposition, and the CPLEX implementation of Benders

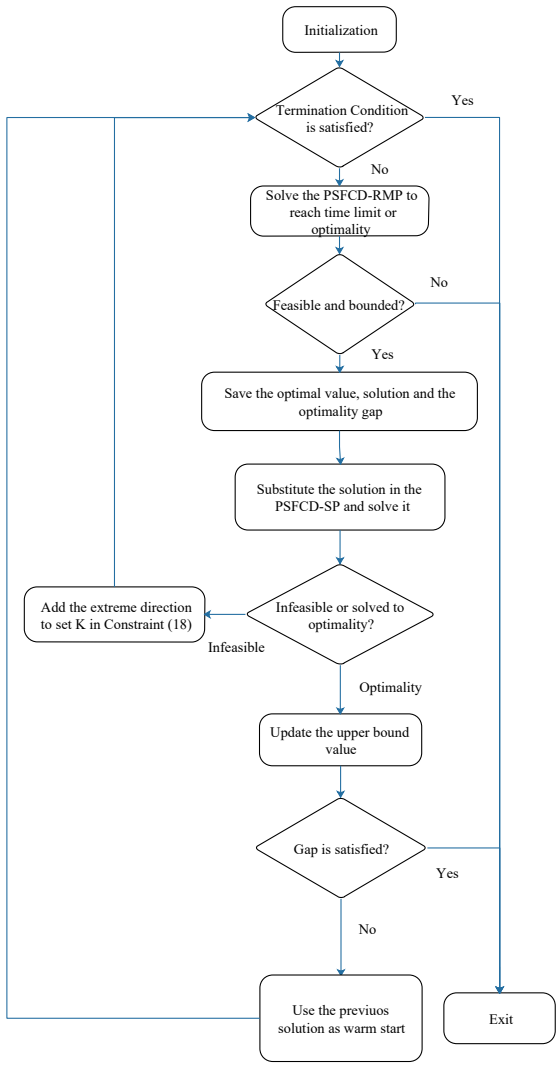


Fig. 3: The exact solving algorithm

[37] in terms of average solution time and total profit. We also investigate the performance of the FIH in terms of the average number of iterations, the percentage of successfully calculating a feasible solution, and the solution time. In calculating the solution time, we only consider the total time spent in solving the models. Finally, the performance of the AEA and B&C is compared in emergency scenarios.

A. Evaluation setup

We performed our simulations on 6-port Fat-tree [38] and USNet topologies [39]. Each simulation run was composed of two steps. The first step was to create a complete topology by randomly attaching servers to the switches and specifying the amount of power profiles, CPU, memory, and bandwidth resources. The next step was to generate a set of best-effort and emergency chains.

For the first step, we configured the USNet topology as follows. We selected 6 switches 0, 4, 9, 10, 18, 23 as core switches and the rest of the switches as edge switches. In each iteration, we randomly attached 54 servers to the edge switches and 6 ingress/egress points to the core switches. We limited the maximum number of servers attached to a switch to 4. The specification of the Fat-tree topology was the same as [38], and we attached 3 servers to each edge switch.

The specifications of all servers and other infrastructure parameters were randomly chosen according to Table VII and VIII. If a server was off, its parameters $AMEM^i$ and $ACPU^i$ were set to the installed CPU and memory resources. We set the bandwidth of the links between ingress/egress points and switches to 40 Gbps. The bandwidth of the links and the rest of the infrastructure parameters were randomly selected according to Table VII. The power profiles of the servers were based on the data of Cisco UCS servers [40], and each server powered off with a probability of 0.25

In each iteration, we randomly generated a set of chains by using the data shown in Table IX and X. The specifications of firewall and router VNFs were based on [41], [42]. The value of the parameter $\eta_i^{q,u}$ was randomly set to 1 with the probability of 0.5 for each server. Also, we allowed all VNFs to migrate, i.e., $\pi^{q,u} = 1\ \forall q \in \mathcal{Q}, u_q \in VNFs$. The source and destination of each chain were randomly mapped to one of the ingress/egress points.

We used four scenarios to evaluate the performance of the algorithms:

- IND scenario: There were no previously deployed chains. A set of best-effort and emergency chains was created. 40% of chains were emergency chains.
- CHGINPUT scenario: This scenario simulates the changes in the input traffic. First, we generated and deployed a set of best-effort and emergency chains with an optimality gap of 0.1 by using the relative mip-gap function [43]. 40% of the generated chains were emergency chains. Then, we saved the locations of the deployed functions and their instances. After that, we selected a subset of chains with a probability of 50%, changed their input traffic, and applied the algorithms.
- CHGADD scenario: This scenario simulates the arrival of new chains. First, we generated a set of best-effort and emergency chains of which 40% of were emergency chains. Then we selected $1/3$ and $1/2$ of the best-effort and emergency chains, respectively, and deployed them with an optimality gap of 0.1. After that, we saved the current instances and locations of the deployed functions and applied the algorithms with the whole set of chains.
- MIX scenario: This scenario is a combination of CHGINPUT and CHGADD scenarios. First, similar to CHGADD, we generated a set of chains and deployed $1/3$ of the best-effort and $1/2$ of emergency chains with an optimality gap of 0.1. Then, we selected a subset of deployed chains with a probability of 50% and changed their input traffic. Finally, all algorithms were executed for the whole set of chains.

All algorithms were implemented by CPLEX v12.8 in Java, and all simulations were carried out on a server with 16 GB of memory and 2 E5-2687 CPU cores operating at 3 GHz. In the B&C, we set the optimality gap to 0.01 and the maximum runtime to 410 seconds. The maximum runtime for solving the Benders master problem in our implementation of Benders was set to 400 seconds. We set the MaxIteration parameter of

TABLE VII: Infrastructure parameters

Links between switches	Uniform [25 Gbps - 40 Gbps]
Links between powered-on servers and switches	Uniform [5 Gbps - (10 or 40) Gbps]
Links between powered-off servers and switches	10 Gbps or 40 Gbps
$AMEM^i$	Uniform [20 - installed Memory]
$ACPU^i$	Uniform [10 - installed vCPU]
Number of Servers-Switches-Ingress/Egress	54-45-9 (Fat-tree) 54-24-6 (USNet)
Energy Price (kWh)	10 cents [44]
Simulation Duration	72 hours
Bandwidth Price (1 Mb/Month)	9.5$ [45]
ψ^q	Number of VNFs $\times$ Bandwidth price
P_{mig}	400 W
BW_{mig}	100 Mbps

TABLE VIII: Servers specifications

	Class 3		Class 2		Class 1	
	High	Low	High	Low	High	Low
Installed vCPU	160	144	104	72	48	32
Installed Memory (GB)	2048	1408	1536	288	768	192
Link (Gbps)	40	40	40	40	10	10
Idle Power (W)	280	270	260	190	180	170
Max Power (W)	1220	1160	960	950	700	600

the FallBack to 30. For the AEA, we set the time limit of the PSFCD-RMP to 50 seconds and the termination condition to reaching 500 iterations or exceeding 400 seconds or achieving an optimality gap below 0.0001. In our implementation of Benders, we calculate the difference between lower and upper bound values [14], [33], and if this value becomes less than 0.01 or the algorithm reaches 500 iterations or exceeds 400 seconds of execution, the algorithm will be terminated. Consequently, the execution of the algorithms may be terminated before reaching the global optimal solution. In the CPLEX implementation of Benders, we set the strategy to FULL and let CPLEX decide about the decomposition. We also set the termination time to 410 seconds. Finally, each point in the figures is the average of at least 30 simulation runs, and the standard deviations are shown as error bars.

TABLE IX: VNFs specifications

VNF Type	CPU	MEM	Throughput (Mbps)	Modification Ratio
Firewall	1	2	1000	0.9
NAT	2	4	1000	1.2
Intrusion Detection System (IDS)	2	3	500	0.8
Router	1	4	1000	1
Deep Packet Inspection (DPI)	2	4	500	0.8
Video Optimizer	3	4	500	0.5

TABLE X: Chains parameters

Chain length (including source and destination)	Uniform [3 - 7]
$\theta_{q,LB}$	Uniform [500 - 4750]
$\theta_{q,UB}$	Uniform [($\theta_{q,LB}$+250) - 5000]
Θ^q	Uniform [500 - 5000]

B. Performance Evaluation of the FIH

In this section, we investigate the performance of the proposed heuristic in terms of the following metrics:

- Solution time: The average of the total time spent solving the models until satisfying the termination condition.
- Success ratio: The number of times a feasible solution is calculated divided by the total number of simulation runs multiplied by 100.
- Iterations: the average total number of iterations until satisfying the termination condition.

As already stated, the FIH only considers the emergency chains. The results are shown in Figure 4. As it is shown, by increasing the total number of requests, the average solution time and the number of iterations increase. On the other hand, the success ratio decreases. Please note that 40% of all requests are emergency SFC requests. For example, if the total number of requests is 50, then there are 20 emergency and 30 best-effort requests.

Increasing the number of requests from 5 to 50 leads to an increase in the solution time from below 1 second to 6.36 and 2.42 seconds in Fat-tree and USNet topologies, respectively. Moreover, the success ratio drops from 100% to 84% and 83% in Fat-tree and USNet topologies, respectively. As it is shown in Figures 4(e) and 4(f), by increasing the number of requests, the average number of iterations increases and reaches 11.8 and 10.36 for Fat-tree and USNet topologies, respectively.

The results show that the proposed heuristic can calculate a feasible solution in less than 7 seconds with a worst-case success ratio of 83% in both Fat-tree and USNet topologies. Therefore, the FIH can be used as a fast and effective heuristic to speed up the main solving algorithm.

C. Performance Evaluation of the AEA

In this section, we compare the performance of the AEA with our implementation of the Benders algorithm, denoted by *Benders*, along with the CPLEX implementation Benders, denoted by *ABenders*, and B&C in terms of solution time and the total profit of the service provider. The results are shown in Figures 5, 6, 7, and 8. As it is shown, by increasing the number of requests, the solution time and total profit increase. In all scenarios, the Benders algorithm fails to find a feasible solution after reaching a specific number of requests. Therefore, we removed the results of the Benders algorithm after that number.

In the IND scenario, there is no previous placement, and the PSFCD must deploy the whole set of requests at once. As it is shown in Figures 5(a) and 5(c), the AEA is 4.63 and 1.18 times faster than the B&C method in deploying 50 chains in the Fat-tree and USNet topologies, respectively, and the Benders algorithm is the slowest algorithm. As it is shown in Figure 5(b), in the Fat-tree topology, the AEA achieves 6.59% more profit than the B&C method in deploying 50 chains. Figure 5(d) compares the performance of the algorithms in the USNet topology and shows the performance of the AEA and B&C is very close in deploying chains up to 45 requests ; however, the B&C outperforms the AEA in deploying 50 chains with 4.02% more profit. Therefore in the USNet topology, the AEA

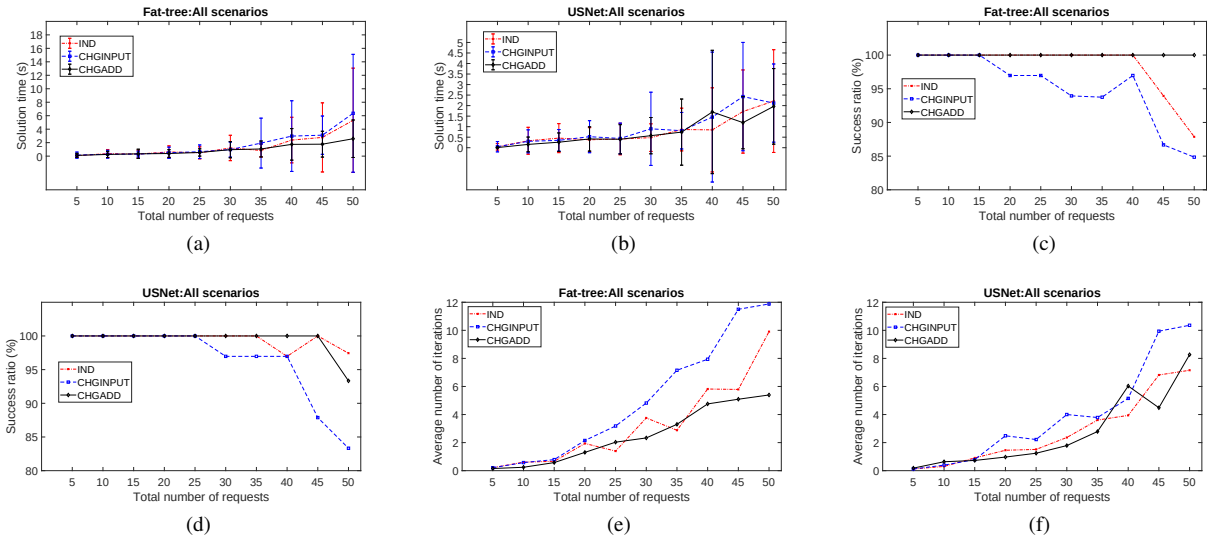


Fig. 4: The performance of the FIH in all scenarios.

performs faster than B&C with a similar performance up to 45 requests. Moreover, in the Fat-tree scenario, the AEA performs better than the B&C in both solution time and total profit.

The CHGINPUT scenario investigates the changes in the input of previously deployed chains. Therefore, there is no new request. The results of the CHGINPUT scenario are shown in Figure 6. As it is shown in Figures 6(a) and 6(c), the AEA performs faster than B&C while the standard Benders method is the slowest algorithm. In the requests with a size of 50, the AEA is 7.94 and 2.26 times faster than the B&C in the Fat-tree and USNet topologies, respectively. Moreover, the AEA achieves 8.4% and 1.25% more profit than the B&C in the Fat-tree and USNet topologies, respectively. Therefore, under the CHGINPUT scenario, the AEA outperforms the B&C and Benders algorithms.

CHGADD scenario deals with the arrival of new SFC requests. The results of the CHGADD are shown in Figure 7. Similar to the previous scenarios, the AEA performs faster than the B&C and Benders algorithms, and the Benders algorithm is the slowest algorithm. In deploying 50 chains, the AEA performs 8.1 and 2.7 times faster than the B&C in Fat-tree and USNet topologies, respectively. As it is shown in Figure 7(b), the AEA achieves 25% more profit than the B&C in the Fat-tree topology. In the USNet topology, the two algorithms AEA and B&C perform very similar in terms of total profit. For example, while the AEA achieves 1.59% more profit in requests of size 45, the B&C achieves 0.37% more profit in the requests of size 50, which is ignorable. Therefore, in the CHGADD scenario, the AEA is the fastest algorithm and can achieve a higher profit than the B&C and the Benders algorithms.

The MIX scenario compares the algorithms in more realistic situations in which there are changes in both the number of requests and the input to them. The results are shown in Figure 8. In both topologies, our implementation of the standard Benders method fails to find a feasible solution to deploy 25 chains or larger. In the USNet, the AEA performs 2.01 and 2.56 times faster than the B&C and ABenders in deploying 50 chins, respectively. In the same size, the AEA reaches 3% and 241%, on average, more profit than the B&C and ABenders, respectively. A similar situation happens in the Fat-tree scenario: the AEA outperforms other algorithms in terms of profit and solution time. The AEA solves the PSFCD problem 3.53 and 3.62 times faster than the B&C and ABenders, respectively. Moreover, the AEA calculates solutions that can gain 37.31% and 228.7% more profit than B&C and ABenders, respectively. In comparison to the Benders, ABenders succeeds in finding feasible solutions in all input sizes. However, the ABenders suffers from a considerably high solution time, even in comparison with Benders. As it is shown in Figures 8(a) and 8(c), the ABenders reaches the time limit in the input of size 15 or higher.

It is also noteworthy to investigate the improvement in the quality of the solution obtained by the AEA from the initial solution calculated by the FIH. A comparison between the profit of these algorithms in the deployment of 50 chains in the MIX scenario is shown in Tabe XI. The last row of Table XI presents a case in which the FIH fails to find a solution while the AEA can successfully calculate a solution. For the other 5 rows, the AEA improves the initial solution by 246% and 215% in Fat-tree and USNet topologies, respectively.

According to the described results, the AEA outperforms other algorithms in terms of solution time. Moreover, the AEA can achieve a comparable or better profit than the B&C. The considerable difference between the performance of the AEA and original Benders is mostly because of the initial solution. The Benders method suffers from a very large number of iterations in which it tries to find a feasible solution. Providing the Benders method with an initial solution will likely improve the solution time. When the FIH fails to find an initial solution, similar to the last row of Table XI , other improvements will become important: the problem-dependent cuts and our flowchart of execution (Figure 3). We included problem-dependent cuts to create a connection between the variables

TABLE XI: A comparison between the performance of the FIH and AEA in the MIX scenario

#	Fat-tree		USNet	
	FIH	AEA	FIH	AEA
1	179002.07	543761.7743	156809.13	518269.24
2	165808.7	565606.12	137438.33	479156.16
3	145481.17	557661.53	174725.25	541131.3
4	160433.53	497175.19	145044.01	451921.6867
5	115908.85	454963.44	156149.38	435260.12
6	–	533699.27	–	389520.02

of sub-problems. This is done by adding Constraints (19) and (20) to the PSFCD-RMP. We also introduced our flowchart of execution to achieve the best results. These features result in the superior performance of the AEA, while the original Benders and ABenders are the slowest algorithms. In addition, the Benders cannot find a feasible solution even in the medium-sized instances of the PSFCD problem, and the ABenders overcomes this issue at the cost of a higher solution time.

D. Performance Evaluation in Emergency Scenarios

In some situations, the service provider must quickly decide to deploy a set of new emergency chains. In these situations, the service provider can set the migration parameter $\pi^{q,u}$ to zero for all previously deployed chains and solve the PSFCD problem. This approach results in a suboptimal solution, but the solution time is much shorter than considering the migrations. In this section, we investigate the performance of this approach in terms of solution time. We considered 50 chains of which 20 were emergency chains. First, we deployed 30 best-effort chains with an optimality gap of 0.1. Then we added 20 emergency chains, forbade the functions of the best-effort chains to migrate, and solved the PSFCD problem with the whole 50 chains again. In this situation, some of the best-effort chains may be rejected or the input rate to them may be decreased. The solution time and total profit of the Fat-tree and USNet topologies are shown in Figure 9.

As shown in Figure 9, the AEA can calculate a better solution faster than the B&C algorithm. The AEA is 2.9 and 1.5 times faster than the B&C while achieving 0.8% and 0.4% more profit in the Fat-tree and USNet topologies, respectively. In both topologies, the AEA can optimally deploy 20 emergency chains and reconfigure 30 best-effort chains in less than 40 seconds, which makes it suitable for emergency situations of this scale.

VI. Conclusion and Future Work

In this paper, we investigated prioritized service function chain deployment and formulated the PSFCD problem that satisfies the QoS requirements of emergency chains in the presence of best-effort chains. The PSFCD aims to maximize service providers' profit subject to satisfying the requested input traffic to emergency chains and resource limits. We included admission control for best-effort chains and management capabilities such as migration control to provide a comprehensive network service deployment and management formulation. In addition to providing the MILP formulation, we introduced two algorithms to satisfy the needs of service operators: if a service provider desires to deploy emergency requests quickly, they can use our polynomial-time heuristic algorithm, FIH. On the other hand. If the service provider considers the maximization of their profit, they can use our exact algorithm, AEA, to obtain an optimal solution faster than other state-of-art algorithms. We showed that the AEA could benefit from the solution obtained by the FIH and outperform the B&C and Benders methods in both solution time and total profit.

Ultimately, from a theoretical viewpoint, it would be interesting to design a heuristic polynomial-time algorithm that considers both best-effort and emergency chains and can find a suboptimal solution. Such algorithms would likely outperform the AEA in terms of solution time and be inferior to it in total profit, which is our main concern. The solution obtained by such algorithms can be fed to other exact algorithms, such as the AEA, to improve it.

References

[1] "Network function virtualization-introductory white paper," ETSI, Tech. Rep., 2012.

[2] J. Halpern and C. Pignataro, "Service function chaining (SFC) architecture," IETF, Tech. Rep., 2015.

[3] Y. Chen, T. Farley, and N. Ye, "QoS requirements of network applications on the internet," *Inf. Knowl. Syst. Manag.*, vol. 4, no. 1, pp. 55–76, 2004-01.

[4] J. Moeyersons, B. Farkiani, B. Bakhshi, S. A. Mirhassani, T. Wauters, B. Volckaert, and F. D. Turck, "Enabling emergency flow prioritization in SDN networks," in *Proc. IEEE CNSM*, 2019.

[5] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 2016.

[6] H. Hantouti, N. Benamar, T. Taleb, and A. Laghrissi, "Traffic steering for service function chaining," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 487–507, 2019.

[7] W. Rankothge, F. Le, A. Russo, and J. Lobo, "Optimizing Resource Allocation for Virtualized Network Functions in a Cloud Center Using Genetic Algorithms," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 2, pp. 343–356, 2017.

[8] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu, "On Dynamic Service Function Chain Deployment and Readjustment," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 543–553, 2017.

[9] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca, "An Approach for Service Function Chain Routing and Virtual Function Network Instance Migration in Network Function Virtualization Architectures," *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2008–2025, 2017.

[10] H. Tang, D. Zhou, and D. Chen, "Dynamic Network Function Instance Scaling Based on Traffic Forecasting and VNF Placement in Operator Data Centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 3, pp. 530–543, 2019.

[11] T. Kuo, B. Liou, K. C. Lin, and M. Tsai, "Deploying Chains of Virtual Network Functions: On the Relation Between Link and Server Usage," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1562–1576, 2018.

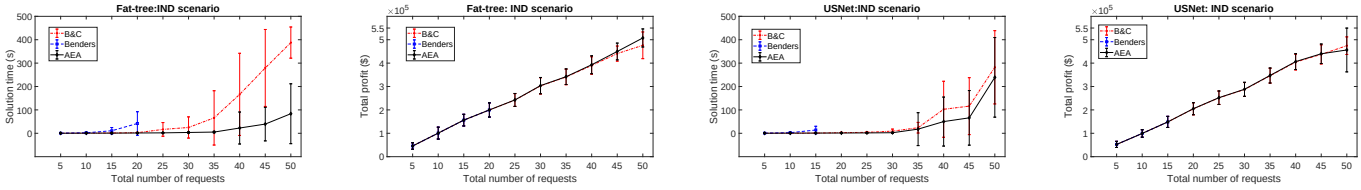
[12] N. Huin, B. Jaumard, and F. Giroire, "Optimal Network Service Chain Provisioning," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1320–1333, 2018.

[13] I. Jang, D. Suh, S. Pack, and G. Dán, "Joint Optimization of Service Function Placement and Flow Distribution for Service Function Chaining," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2532–2541, 2017.

[14] B. Farkiani, B. Bakhshi, and S. Mirhassani, "A Fast Near-Optimal Approach for Energy-Aware SFC Deployment," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1360–1373, 2019.

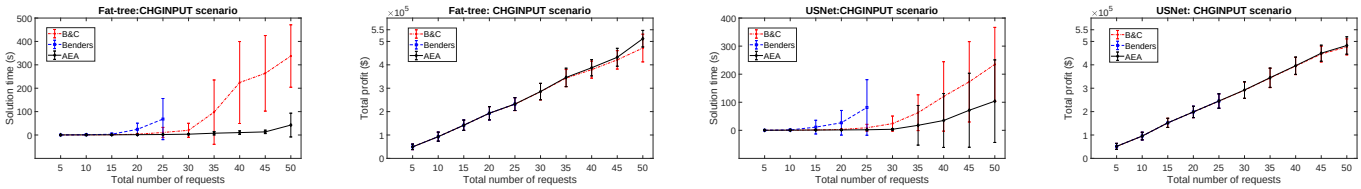
[15] A. Laghrissi and T. Taleb, "A Survey on the Placement of Virtual Resources and Virtual Network Functions," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1409–1434, 2019.

[16] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating Virtualized Network Functions," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 4, pp. 725–739, 2016.



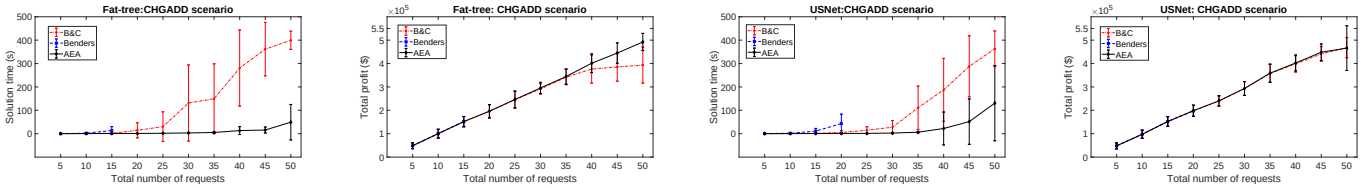
(a) (b) (c) (d)

Fig. 5: The performance of the algorithms in the IND scenario.



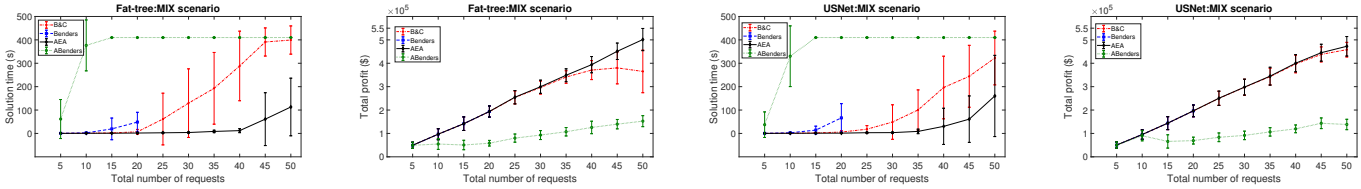
(a) (b) (c) (d)

Fig. 6: The performance of the algorithms in the CHGINPUT scenario.



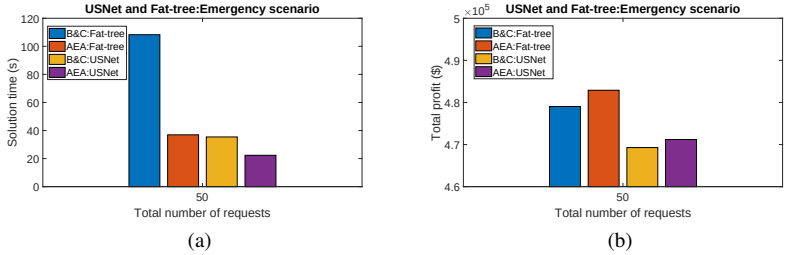
(a) (b) (c) (d)

Fig. 7: The performance of the algorithms in the CHGADD scenario.



(a) (b) (c) (d)

Fig. 8: The performance of the algorithms in the MIX scenario.



(a) (b)

Fig. 9: The performance of the AEA and B&C in the emergency scenario.

[17] G. Sun, Y. Li, H. Yu, A. V. Vasilakos, X. Du, and M. Guizani, "Energy-efficient and traffic-aware service function chaining orchestration in multi-domain networks," *Future Generation Computer Systems*, vol. 91, pp. 347–360, 2019.

[18] F. Malandrino, C. F. Chiasserini, G. Einziger, and G. Scalosub, "Reducing Service Deployment Cost Through VNF Sharing," *IEEE/ACM Trans. Netw.*, pp. 1–14, 2019.

[19] V. Eramo, M. Ammar, and F. G. Lavacca, "Migration Energy Aware Reconfigurations of Virtual Network Function Instances in NFV Architectures," *IEEE Access*, vol. 5, pp. 4927–4938, 2017.

[20] X. Fei, F. Liu, H. Xu, and H. Jin, "Adaptive VNF Scaling and Flow Routing with Proactive Demand Prediction," in *Proc. IEEE INFOCOM*, 2018, pp. 486–494.

[21] L. Tang, X. He, P. Zhao, G. Zhao, Y. Zhou, and Q. Chen, "Virtual Network Function Migration Based on Dynamic Resource Requirements Prediction," *IEEE Access*, vol. 7, pp. 112 348–112 362, 2019.

[22] Y. Gu, Y. Hu, Y. Ding, J. Lu, and J. Xie, "Elastic Virtual Network Function Orchestration Policy Based on Workload Prediction," *IEEE Access*, vol. 7, pp. 96 868–96 878, 2019.

[23] M. Huang, W. Liang, Y. Ma, and S. Guo, "Maximizing Throughput of Delay-Sensitive NFV-Enabled Request Admissions via Virtualized Network Function Placement," *IEEE Trans. on Cloud Comput.*, pp. 1–1, 2019.

[24] M. Dayarathna, Y. Wen, and R. Fan, "Data Center Energy Consumption Modeling: A Survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 732–794, 2016.

[25] A. Marotta, F. D'Andreagiovanni, A. Kassler, and E. Zola, "On the energy cost of robustness for green virtual network function placement in 5G virtualized infrastructures," *Computer Networks*, vol. 125, pp. 64–75, 2017.

[26] A. Strunk and W. Dargie, "Does Live Migration of Virtual Machines Cost Energy?" in *Proc. IEEE AINA*, 2013, pp. 514–521.

[27] L. Bertacco, M. Fischetti, and A. Lodi, "A feasibility pump heuristic for general mixed-integer problems," *Discrete Optimization*, vol. 4, no. 1, pp. 63–76, 2007.

[28] M. Fischetti, F. Glover, and A. Lodi, "The feasibility pump," *Mathematical Programming*, vol. 104, no. 1, pp. 91–104, 2005.

[29] M. Fischetti and D. Salvagnin, "Feasibility pump 2.0," *Mathematical Programming Computation*, vol. 1, no. 2, pp. 201–222, 2009.

[30] L. G. Khachiyan, "Polynomial algorithms in linear programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 20, no. 1, pp. 53–72, 1980.

[31] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, no. 4, pp. 373–395, 1984.

[32] J. F. Benders, "Partitioning Procedures for Solving Mixed-variables Programming Problems," *Numerische Mathematik*, vol. 4, no. 1, pp. 238–252, 1962.

[33] B. Farkiani, B. Bakhshi, and S. MirHassani, "Stochastic virtual network embedding via accelerated Benders decomposition," *Future Generation Computer Systems*, vol. 94, pp. 199–213, 2019.

[34] addmipstart. [Online]. Available: https://ibm.co/32H102c

[35] getmiprelativegap. [Online]. Available: https://ibm.co/3ajE3EQ

[36] CPLEX 12.8 Java API Reference Manual. [Online]. Available: https://ibm.co/2VG4h09

[37] Benders algorithm. [Online]. Available: https://ibm.co/2IGNo0R

[38] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," in *Proc. SIGCOMM*. ACM, 2008, pp. 63–74.

[39] C. Ren, S. Wang, J. Ren, W. Qian, X. Zhang, and J. Duan, "Energy-efficient virtual topology design in IP over WDM mesh networks," *Computer Networks*, vol. 112, pp. 223–236, 2017.

[40] Cisco NFV Infrastructure. [Online]. Available: http://bit.ly/2HqT1gU

[41] Cisco Adaptive Security Virtual Appliance (ASAv) Data Sheet. [Online]. Available: http://bit.ly/2TA5TZw

[42] Cisco Cloud Services Router 1000v Data Sheet. [Online]. Available: http://bit.ly/2Ci5n7B

[43] Mipgap. [Online]. Available: https://ibm.co/2IvDBHL

[44] Electricity Rates by State (December 2019). [Online]. Available: http://bit.ly/2Tga37b

[45] Comparing Bandwidth Costs of Amazon, Google and Microsoft Cloud Computing. [Online]. Available: http://bit.ly/2TwPi65

APPENDIX A
THE PSFCD FORMULATION

$$\max\Bigg(\Bigg(\sum_{q\in\mathcal{Q}}\Big(\Theta^q+\dot{\Theta}^q\Big)\times\psi^q\times\Delta\Bigg) - \Bigg(\sum_{u_q\in VNFs} m^{q,u}\times\Big(\varsigma'^{q,u}\times\tfrac{MEM^{q,u}}{BW_{mig}}\times 2P_{mig}\Big)\times\mathcal{E}\Bigg) - \Bigg(\sum_{i\in V^S}\big(\beta_i\times P^i_{idle}\times\big(1-State^i\big) + \sum_{u_q\in VNFs}\varsigma_i^{q,u}\times CPU^{q,u}\times\tfrac{P^i_{max}-P^i_{idle}}{ICPU^i}\big)\times\Delta\times\mathcal{E}\Bigg)\Bigg) \quad \textbf{PSFCD}$$

$$\sum_{i\in V^S\cup Dum} x_i^{q,u}\times\eta_i^{q,u}=1\ \forall q\in M, u_q\in VNFs\cup\{Src,Dst\} \quad (1)$$

$$\sum_{i\in V^S\cup Dum} x_i^{q,u}\times\eta_i^{q,u}=\alpha^q\ \forall q\in B, u_q\in VNFs\cup\{Src,Dst\} \quad (2)$$

$$x_i^{q,u}\times\dot{\phi}^{q,u}\le\Im^{q,u}\times\varsigma_i^{q,u}\ \forall q\in M, u_q\in VNFs, i\in V^S|\eta_i^{q,u}=1 \quad (3)$$

$$x_i^{q,u}\times\phi^{q,u}\le\Im^{q,u}\times\varsigma_i^{q,u}\ \forall q\in B, u_q\in VNFs, i\in V^S|\eta_i^{q,u}=1 \quad (4)$$

$$\varsigma_i^{q,u}=\big(\varsigma'^{q,u}\times\big(xC_i^{q,u}+(1-xC_i^{q,u})\times x_i^{q,u}\big)\big) + \big(CR_i^{q,u}-REM_i^{q,u}\big)\ \forall q\in\mathcal{Q}, u_q\in VNFs, i\in V^S|\eta_i^{q,u}=1 \quad (5)$$

$$\pi^{q,u}\times m^{q,u}=1-x_i^{q,u}\quad \forall q\in M,\ u_q\in VNFs, i\in V^S|xC_i^{q,u}=1 \quad (6)$$

$$\pi^{q,u}\times m^{q,u}\le 1-x_i^{q,u}\quad \forall q\in B,\ u_q\in VNFs, i\in V^S|xC_i^{q,u}=1 \quad (7)$$

$$\pi^{q,u}\times m^{q,u}\le\alpha^q\qquad \forall q\in B, u_q\in VNFs \quad (8)$$

$$\pi^{q,u}\times m^{q,u}\ge\alpha^q-x_i^{q,u}\quad \forall q\in B, u_q\in VNFs, i\in V^S|xC_i^{q,u}=1 \quad (9)$$

$$\sum_{u_q\in VNFs}\varsigma_i^{q,u}\times CPU^{q,u}\le ACPU^i\ \forall i\in V^S \quad (10)$$

$$\sum_{u_q\in VNFs}\varsigma_i^{q,u}\times MEM^{q,u}\le AMEM^i\ \forall i\in V^S \quad (11)$$

$$x_i^{q,u}\le\beta_i\ \forall q\in\mathcal{Q}, u_q\in VNFs, i\in V^S \quad (12)$$

$$x_i^{q,u}+x_i^{q,u+1}\le 1\quad \forall q\in Q, u_q\in VNFs, i\in V^S|\left(\eta_i^{q,u}=1\&\eta_i^{q,u+1}=1\right) \quad (13)$$

$$\sum_{(i,j)\in E} y_{i,j}^{q,(u,v)}-\sum_{(j,i)\in E} y_{j,i}^{q,(u,v)}=0\quad \forall i\in V^N, q\in\mathcal{Q}, (u,v)\in E^{V,q} \quad (14)$$

$$y_{(i,j)}^{q,(u,v)}-y_{(j,i)}^{q,(u,v)}=\dot{\phi}^{q,v}\times\big(x_i^{q,u}\times\eta_i^{q,u}-x_i^{q,v}\times\eta_i^{q,v}\big)\quad \forall i\in V^S\cup Dum, (i,j)\in E,\ q\in M,\ (u,v)\in E^{V,q} \quad (15)$$

$$y_{(i,j)}^{q,(u,v)}-y_{(j,i)}^{q,(u,v)}=\phi^{q,v}\times\big(x_i^{q,u}\times\eta_i^{q,u}-x_i^{q,v}\times\eta_i^{q,v}\big)\quad \forall i\in V^S\cup Dum, (i,j)\in E, q\in B, (u,v)\in E^{V,q} \quad (16)$$

$$\sum_{q\in Q}\sum_{(u,v)\in E^{V,q}} y_{(i,j)}^{q,(u,v)}\le ABW_{(i,j)}\ \forall(i,j)\in E \quad (17)$$

$$\theta_{q,LB}\times\alpha^q\le\Theta^q\le\theta_{q,UB}\times\alpha^q\ \forall q\in B \quad (18)$$