Towards Latency Optimization in Hybrid Service Function Chain Composition and Embedding

Danyang Zheng[1], Chengzong Peng[1], Xueting Liao[1], Ling Tian[2], Guangchun Luo[2], and Xiaojun Cao[1]

[1]Department of Computer Science, Georgia State University, USA
[2]School of Computer Science and Engineering, University of Electronic Science and Technology of China
[1]dzheng5@student.gsu.edu, cpeng7@student.gsu.edu, xliao3@student.gsu.edu and cao@gsu.edu
[2]lingtian@uestc.edu.cn and gcluo@uestc.edu.cn

***Abstract*— In Network Function Virtualization (NFV), to satisfy the Service Functions (SFs) requested by a customer, service providers will composite a Service Function Chain (SFC) and embed it onto the shared Substrate Network (SN). For many latency-sensitive and computing-intensive applications, the customer forwards data to the cloud/server and the cloud/server sends the results/models back, which may require different SFs to handle the forward and backward traffic. The SFC that requires different SFs in the forward and backward directions is referred to as hybrid SFC (h-SFC). In this paper, we, for the first time, comprehensively study how to optimize the latency in Hybrid SFC composition and Embedding (HSFCE). When each substrate node provides only one unique SF, we prove the NP-hardness of HSFCE and propose the first 2-approximation algorithm to jointly optimize the processes of h-SFC construction and embedding, which is called Eulerian Circuit based Hybrid SFP optimization (EC-HSFP). When a substrate node provides various SFs, we extend EC-HSFP and propose the efficient Betweenness Centrality based Hybrid SFP optimization (BC-HSFP) algorithm. Our extensive simulations and analysis show that EC-HSFP can hold the 2-approximation, while BC-HSFP outperforms the algorithms directly extended from the state-of-art techniques by an average of 20%.**

I. Introduction

Network Function Virtualization (NFV) transforms the implementation of network function from the dedicated hardware-based appliance to the software-based virtual module called Virtual Network Function (VNF) or Service Function (SF) [1]. As a virtualized module, the SF can be flexibly installed on or removed from the substrate/physical nodes (e.g., high volume servers, or servers in datacenters). With NFV, a Network Service Request (NSR) from the customer includes a set of SF nodes and the related resource demands (e.g., CPU and bandwidth) [2]. To satisfy an NSR, the service provider chains the requested SF nodes through SF links to composite a Service Function Chain (SFC), which is then mapped onto a shared Substrate/physical Network (SN) by reserving the necessary substrate resource [3]. The composited SFC specifies the ordered set of SFs that will be applied to the traffic (packets) from the customers. The process for satisfying an NSR from the customer is referred to as SFC composition and Embedding (SFCE), which consists of three inter-related sub-processes: SFC composition, SF node mapping and SF link mapping. The SFC composition sub-process will create a chain of SF nodes according to SF ordering constraints in the NSR. The SF node mapping embeds the SF nodes onto substrate nodes, while the SF link mapping accommodates each SF link with physical links/paths. The physical path accommodating all the SF nodes/links of an NSR is called the Service Function Path (SFP) [4]. Note that, when an NSR specifies the complete order of required SFs or the SFC is already given (or fixed), SFCE is reduced to the traditional Virtual Network Embedding (VNE) [5].

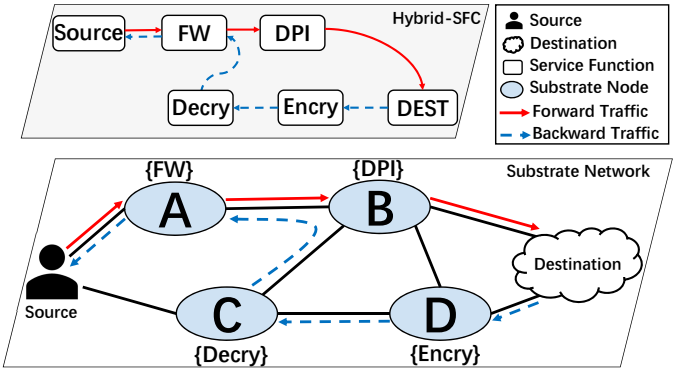


Fig. 1: An example of in-service h-SFC for one on-line machine learning scenario

Recently, 5G and Multi-access Edge Computing (MEC) empowers the development of the latency-sensitive or computation-intensive applications such as realtime Virtual Reality (VR)/Augmented Reality (AR) games and on-line machine learning [6]. In these applications, the forward traffic from the user and the backward traffic from the MEC server/cloud may require different sets of SFs. The SFC that requires different sets of SFs in the forward and backward directions is referred to as hybrid SFC (h-SFC) [7]. Fig. 1 demonstrates an example of an in-service h-SFC for on-line machine learning. The forward traffic (containing data sets) from the source (i.e., user) requires Fire Wall (FW) and Deep Packet Inspection (DPI), while the backward traffic (containing the trained model) has to go through Encryption (Encry), Decryption (Decry) and FW. In the SN, the forward SFP (f-SFP) is Source→A→B→DEST, while the backward SFP (b-SFP) includes DEST⇢D⇢C⇢B⇢A⇢Source. To save the Operating Expense (OPEX) and latency, the SFs required by both directions are generally installed on the same

substrate node (e.g., FW in Fig. 1) [8] [9].

In this paper, for the first time, we comprehensively study how to jointly composite and embed an NSR with hybrid traffic onto a shared substrate network. We define a new problem called *Hybrid SFC composition and Embedding* (HSFCE) and propose novel analysis and algorithms for various substrate network scenarios. Our main contributions in this paper are summarized as follows.

- We mathematically model the problem of Hybrid SFC composition and Embedding (HSFCE) with the objective of minimizing the latency for the constructed hybrid SFP.
- When each substrate node provides only one unique SF, we prove the NP-hardness of HSFCE and propose a 2-approximation algorithm to jointly composite and embed the h-SFC. The proposed algorithm is called Eulerian Circuit based Hybrid SFP optimization (EC-HSFP).
- When a substrate node can provide various SFs, we propose an effective heuristic algorithm called Betweenness Centrality based Hybrid SFP optimization (BC-HSFP).
- Through extensive analysis and simulations, we prove that EC-HSFP guarantees the 2-approximation performance and show that BC-HSFP outperforms the algorithms directly extended from the existing techniques by an average of 20%.

The rest of this paper is organized as follows. In Section II, we introduce the concepts of hybrid SFC, SFP and SFC composition and Embedding (SFCE). Section III mathematically formulates the HSFCE problem. In Section IV and V, we provide analysis and algorithms for HSFCE in various substrate network scenarios. Section VI presents the experimental results and analysis. We conclude our work in Section VII.

II. Service Function Chain composition and Embedding

In this section, we introduce the concepts of hybrid SFC (h-SFC), hybrid SFP and classify the Service Function Chain composition and Embedding (SFCE) related problems.

A. Hybrid Service Function Chain

An h-SFC consists of a forward SFC (f-SFC $=\{s, f_{f_1}, ..., f_{f_i}\}$) and a backward SFC (b-SFC $=\{b_{f_1}, ..., b_{f_j}, s\}$), where s is the source node; f_{f_i}/b_{f_j} represents a specific SF in the forward/backward direction. To keep the continuity of h-SFC data stream, the backward traffic is required to start with the last forward SF f_{f_i} (i.e., $b_{f_1} = f_{f_i}$), even though f_{f_i} may not be required by the backward traffic [7].

B. Service Function Path for Hybrid Service Function Chain

When the requested backward SFs are totally different from the forward SFs (i.e., f-SFC $\cap$ b-SFC $= \emptyset$), one can regard the hybrid SFP as a concatenation of two independent SFPs. However, if the forward and backward SFCs share some SFs (i.e., f-SFC $\cap$ b-SFC $\neq \emptyset$), the hybrid SFP cannot be simply treated as a concatenation of two independent SFPs. This is because, mapping the common SF(s) (i.e., the SFs requested in both f-SFC and b-SFC) onto the same substrate node(s) can significantly save the OPEX and reduce the latency [8] [9]. In [8], the authors reported that scaling an SF by enlarging the capacity of an existing SF instance only needs milliseconds, while scaling an SF by initializing a new SF instance needs tens seconds. Similarly, the work in [9] finds out that doubling the workload of an SF instance implemented by the micro-service, the CPU usage will be much less than double. Accordingly, we assume that the common SFs between f-SFC and b-SFC shall be mapped onto the same substrate node.

C. SFC Composition and Embedding Classification

The SFCE related problems can be classified into three groups according to the traffic constraint as shown in Fig. 2.

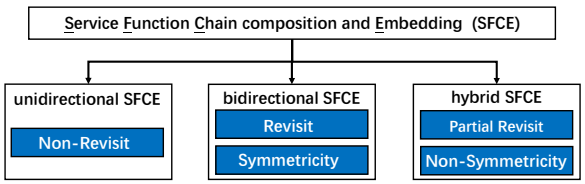
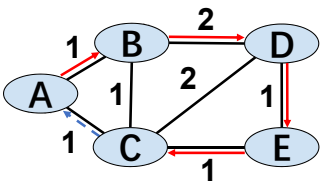


Fig. 2: SFC composition and embedding classification

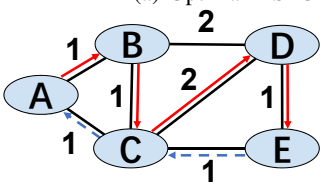
Unidiretional SFCE: A unidirectional SFC requires that the customer's traffic is forwarded through a set of specified SFs in one direction. The *unidirectional SFP* generated from the unidirectional SFCE process is *non-revisit* (i.e., each of the required SFs is executed only once). There is much existing work focusing on optimizing the unidirectional SFCE related problems [10]–[18]. For example, in [10], the authors investigated the NFV location problem with approximation algorithms. The authors in [11] showed that to satisfy the SFC requirement, the SF placement problem can be treated as an instance of the set cover problem. When the SFC is given a prior, the authors in [12] developed the SFC-constrained shortest path schemes with the transformation of network graphs. The authors in [13] investigated joint optimization over placement and routing of SFCs in datacenters.

Bidirectional SFCE: A bidirectional SFC requires the traffic processed by the SFs in the forward and backward directions identically. Therefore, the SFP generated from the bidirectional SFCE process needs to execute the required SFs twice (*Revisit*). As aforementioned, the common SFs should be embedded onto the same substrate nodes. Hence, one can employ the existing unidirectional SFCE schemes to directly accommodate a bidirectional SFCE.

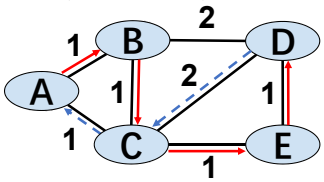
Hybrid SFCE (HSFCE): With hybrid SFC, the SFs required by the forward and backward traffic are not the same (*non-Symmetricity*). Only the common SFs (i.e., requested by both forward and backward traffic) in the hybrid SFP need to be visited twice (*Partial-Revisit*). Since the last SF node of the f-SFC is not determined a priori, HSFCE is different from the unidirectional/bidirectional SFCE and the unidirectional SFCE techniques cannot be efficiently applied. We use Fig. 3 to illustrate the challenges in HSFCE. Fig. 3 shows an SN with enough bandwidth, where the number beside each link represents the latency (or cost). Substrate node A, B, C, D and E can host SF instance: f1, f2, f3, f4 and f5, respectively. We assume that there is an NSR starting



(a) Optimal HSFCE result (total cost = 6)



(b) Impact of endpoints selection (total cost = 7)



(c) Optimal SFCE result (total cost = 7)

Fig. 3: HSFCE and SFCE

at node A and requiring f1, f2, f3, f4, f5, where f1 and f3 are requested by both forward and backward traffic. Fig. 3a is the optimal accommodation for the given NSR. Fig. 3b demonstrates the impact of the selection of the endpoint (i.e. last SF node) of f-SFC. Fig. 3c is the result of applying the traditional optimized SFCE technique. In Fig. 3a and 3b, the composited forward SFCs are f1→f2→f4→f5→f3 and f1→f2→f3→f4→f5, whose forward SFPs have the same cost as 5 (i.e., SFP A→B→D→E→C and A→B→C→D→E). Fig. 3b has to composite the backward SFC as f5⇢f3⇢f1 for the backward traffic via SFP E⇢C⇢A, whose cost is 2. Fig. 3a employs the backward SFC as f3⇢f1 and SFP C⇢A with a cost of 1. Fig. 3c embeds the NSR with forward SFC f1→f2→f3→f5→f4 along the shortest SFP A→B→C→E→D at the cost of 4. For the backward traffic, it costs 3 to revisit the SF instances for f1 and f3 (i.e., A and C).

As one can see that, the existing unidirectional SFCE technique cannot be directly applied to efficiently optimize HSFCE. Hence, in the following sections, we propose efficient mathematical model, algorithms and analysis for HSFCE.

III. Hybrid Service Function Chain composition and Embedding

A. Substrate/physical Network Model

The Substrate Network (SN) is denoted by an undirected graph $G = (N, L, F)$, whereas N represents a set of substrate nodes, L is a set of substrate links, and F denotes a set of available SFs. Each substrate node $n \in N$ provides a specific set of SF instances $\mathbb{F}_n$ ($\mathbb{F}_n \subseteq F$) and a certain amount of available computing capacity c_n. Each physical link $l \in L$ can provide a specific amount of bandwidth bw_l and has a certain latency. For simplicity, a physical link $l \in L$ can also be represented as $l_{m,n}$, where $m, n \in N$ are its endpoints. For a physical path $path_{m,n}$, it has an accumulative latency cost $W_{path_{m,n}}$ depending on the links in this path. In this work, we investigate two different network scenarios, Unique Function SN (UFSN) and Multi-Functions SN (MFSN). In UFSN, each substrate node only provides one unique SF instance. That is to say, no two different substrate nodes provide the same SF instance (i.e., $\mathbb{F}_n \cap \mathbb{F}_m = \emptyset$, $\forall m \neq n$). This scenario is practical in the sub-domains of a large network or the resource-constrained systems [19], [20]. In MFSN, each substrate node can provide various SF instances subjecting to the computing capacity and different substrate nodes may offer the same SF instance(s) (i.e., $\mathbb{F}_n \cap \mathbb{F}_m$ may not be empty).

B. Network Service Request with Hybrid Traffic

A Network Service Request with hybrid traffic can be represented as a 4-tuple $NSR =< s, V_f, V_b, BW >$, where s is the source node, V_f represents the set of required forward SF nodes, V_b specifies the set of backward SF nodes and BW denotes the amount of bandwidth demand. Each SF node $v \in V$ ($V = V_f \cup V_b$) requires a specific SF f_v and a certain amount of computing demand c_v.

C. Hybrid Service Function Chain composition and Embedding (HSFCE)

The optimization problem of the HSFCE is defined as: given an NSR with hybrid traffic demands, how to composite and embed the hybrid SFC onto a shared SN such that i) the constraints below are satisfied, and ii) the latency of the constructed SFPs is minimized. The objective function is shown in Eq. (1), where $\mu_{path_{m,n}^{u,v}}$ represents the latency of the $path_{m,n}$ to support the traffic from SF node u to v.

$$min \sum_{v \in V} \sum_{u \in V} \sum_{n \in N} \sum_{m \in N} \mu_{path_{m,n}^{u,v}} \tag{1}$$

SF node mapping constraint: To map an NSR, SF node mapping process needs to follow the constraints in Eqs. (2-6). We use M_n^v in Eq. (2) to represent whether an SF node v is mapped onto the substrate node n and Δ_n^v in Eq. (3) to denote whether the substrate n provides the SF instance for the SF node v. Eq. (4) ensures that every SF node has to be mapped onto one specific substrate node. In Eq. (5), an SF node can only be mapped onto the substrate node that provides the corresponding SF instance. Each substrate node can host a limited number of SF nodes due to the computing capacity as shown in Eq. (6).

$$M_n^v = \begin{cases} 1, & \text{SF node } v \in V \text{ is mapped onto} \\ & \text{substrate node } n \in N \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$\Delta_n^v = \begin{cases} 1, & \text{substrate node } n \text{ provides} \\ & \text{SF instance for } v \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$\sum_{n \in N} M_n^v = 1, \quad \forall v \in V \tag{4}$$

$$M_n^v \leq \Delta_n^v, \quad \forall n \in N, \forall v \in V \tag{5}$$

$$\sum_{v \in V} M_n^v * c_v \leq c_n, \quad \forall n \in N \tag{6}$$

forward/backward SFP (f-SFP/b-SFP) construction constraint: We use $\text{path}_{m,n}^{u,v}$ to denote whether the path from m to n is used to support the traffic from SF node u to v as shown in Eq. (7). In Eq. (8), only the path whose endpoints are mapped by some SF nodes is able to construct the f-SFP/b-SFP. Note that, we use f_s to represent the function for the source node. Eq. (9) guarantees that there must be one path going from the

source to the substrate node n that is mapped by one forward SF node. If an SF node u is demanded in both directions, Eq. (10) ensures that there are two paths starting from the substrate node m where u is mapped (one for f-SFP and the other one for b-SFP). Note that, if an SF node u required by both directions is the last SF node in the f-SFP, the first backward SF node is also u, which is represented as $path^{u,u}_{m,m} = 1$. If an SF node u is demanded only in forward/backward direction, Eq. (11)/(12) ensures that there is only one path starting from the substrate node m where u is mapped. Eq. (13) shows that, for any substrate node m, the number of outgoing path(s) from m equals the number of incoming path(s) to m. The number of connections among any proper-subset of the mapped SF nodes V_{sub} should not be more than $|V_{sub}| - 1$ to avoid the circle, which is shown in Eq. (14). Eq. (15) guarantees that each link of the selected paths should provide enough bandwidth. Eq. (16) shows the latency of $\text{path}^{u,v}_{m,n}$.

$$\text{path}^{u,v}_{m,n} = \begin{cases} 1, & \text{path from substrate node } m \text{ to } n \text{ is used} \\ & \text{to support the traffic from SF node } u \text{ to } v \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

$$\text{path}^{u,v}_{m,n} \leq \frac{M^u_m + M^v_n}{2}, \forall m, n \in \{N \cup s\}, \forall u, v \in \{V \cup f_s\} \quad (8)$$

$$\sum_{v \in V_f} \sum_{n \in N} \text{path}^{f_s,v}_{s,n} = 1 \quad (9)$$

$$\sum_{n \in N} \sum_{v \in V} (\text{path}^{u,v}_{m,n} + \text{path}^{u,f_s}_{m,s} + \text{path}^{u,u}_{m,m}) = 2 * M^u_m, \quad (10)$$
$$\forall u \in \{V_f \cap V_b\}, u \neq v, \forall m \in N$$

$$\sum_{n \in N} \sum_{v \in V} \text{path}^{u,v}_{m,n} = M^u_m, \quad (11)$$
$$\forall u \in \{V_f - (V_f \cap V_b)\}, u \neq v, \forall m \in N$$

$$\sum_{n \in N} \sum_{v \in V} (\text{path}^{u,v}_{m,n} + \text{path}^{u,f_s}_{m,s}) = M^u_m, \quad (12)$$
$$\forall u \in \{V_b - (V_b \cap V_f)\}, u \neq v, \forall m \in N$$

$$\sum_{n \in \{N \cup s\}} \sum_{v \in \{V \cup f_s\}} \text{path}^{u,v}_{m,n} = \sum_{o \in \{N \cup s\}} \sum_{w \in \{V \cup f_s\}} \text{path}^{w,u}_{o,m}, \quad (13)$$
$$\forall u \in V, \forall m \in N$$

$$\sum_{u \in V_{sub}} \sum_{v \in V_{sub}} \sum_{m \in N} \sum_{n \in N} \text{path}^{u,v}_{m,n} \leq |V_{sub}| - 1, \quad (14)$$
$$\forall V_{sub} \subsetneqq \{V \cup f_s\}, V_{sub} \neq \emptyset, u \neq v$$

$$\sum_{u \in V} \sum_{v \in V} \text{path}^{u,v}_{m,n} * BW \leq bw_{l_{a,b}}, \forall l_{a,b} \in \text{path}_{m,n} \quad (15)$$

$$\mu_{\text{path}^{u,v}_{m,n}} = \text{path}^{u,v}_{m,n} * W_{\text{path}_{m,n}}, \forall u, v \in V, \forall m, n \in N \quad (16)$$

forward/backward SFPs connection constraint: In the f-SFP, only the last substrate node in f-SFP should connect with the substrate node hosting the SF that is only required in the backward direction. Hence, there should be at most one path supporting the traffic from the forward SF (only required by the forward direction) to the backward SF (only required by the backward direction) as shown in Eq. (17). Similarly, if a substrate node m hosts an SF node that is only requested by the backward direction, there should be no connection from m to a substrate node n hosting the SF node that is only required by the forward traffic, which is shown in Eq. (18).

$$\sum_{u \in \{V_f - (V_f \cap V_b)\}} \sum_{v \in \{V_b - (V_f \cap V_b)\}} \text{path}^{u,v}_{m,n} \leq 1, \forall m, n \in N \quad (17)$$

$$\sum_{u \in \{V_b - (V_f \cap V_b)\}} \sum_{v \in \{V_f - (V_f \cap V_b)\}} \text{path}^{u,v}_{m,n} = 0, \forall m, n \in N \quad (18)$$

IV. Hybrid SFCE in UFSNs

In this section, we investigate how to jointly composite and embed the hybrid SFCs onto the UFSN. We prove the NP-hardness of HSFCE in UFSN, and propose a 2-approximation algorithm, namely, Eulerian Circuit based Hybrid SFP optimization (EC-HSFP), which includes two proposed techniques: i) Hybrid Trace Construction (HTC), and ii) Hybrid Eulerian Circuit Construction (HECC).

A. Complexity Analysis of HSFCE in UFSN

When the UFSN is a complete graph and every node provides a requested SF, creating an SFP is equivalent to finding a path that visits each node exactly once. When the UFSN is a random graph, we can convert it to a complete graph and efficiently apply the proposed HTC and HECC techniques. Fig. 4 shows an example of a UFSN, where the substrate nodes A, B, C, D, E provide SF instances: f1, f2, f3, f4, f5, respectively. We assume that an NSR requires f1, f2, f3, f4, where f1 and f2 are required by the forward traffic while f3 and f4 are demanded in both directions. The dark nodes in Fig. 4 will host the required SF instances. We can then generate a complete graph with only dark nodes by connecting the dark nodes via the shortest paths in the UFSN. This complete graph is called SFP Complete Graph (SFP-CG). Fig. 5 shows the SFP-CG generated from Fig. 4, where the number beside the link represents the smallest latency between that pair of substrate nodes.

Lemma 1. *Given a UFSN, an optimal hybrid SFP for an NSR with hybrid traffic exists in the SFP-CG.*

Theorem 1. *Hybrid SFC composition and embedding in unique function substrate network is NP-hard.*

Proof. We prove the NP-hardness of HSFCE in UFSN via the reduction of the Travelling Salesman Path Problem (TSPP) [21]. The TSPP tries to find the shortest path that visits each city (node) in the graph exactly once. The HSFCE in UFSN tries to find the shortest SFP connecting the substrate nodes that provide the required SF instances in the UFSN for the forward and backward traffic, respectively; which is equivalent to finding two travelling salesman paths sharing the same

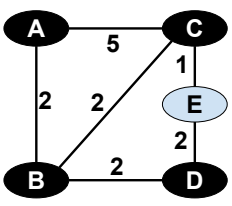


Fig. 4: An example of UFSN

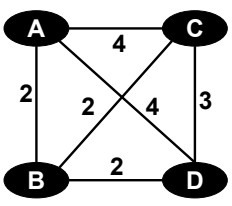


Fig. 5: SFP complete graph

endpoints and connecting the substrate nodes in the SFP-CG. Thus, HSFCE in UFSN is NP-hard. □

However, the HSFCE in UFSN cannot be simply optimized by applying the TSPP algorithms twice (one for f-SFP and the other one for b-SFP). This is because, when the optimal f-SFP and b-SFP are generated by the TSPP algorithms, the connection between f-SFP and b-SFP may be required, which may introduce large latency for the constructed hybrid SFP. Thus, HSFCE in UFSN cannot be directly optimized by applying the existing TSPP techniques.

B. Hybrid Trace Construction (HTC)

In graph theory, a Hamiltonian path on a connected graph is a path of minimal length which visits every node of a graph exactly once [22]. Accordingly, the optimal unidirectional SFP is a Hamiltonian path of the SFP-CG. Based on the number of connections in the Hamiltonian path, the substrate nodes involved in the construction of a unidirectional SFP can be two types: i) *endpoints*, and ii) *intermediate nodes*. The endpoints include source and destination, whereas the number of connections is odd. For intermediate nodes, they own even number of connections.

Lemma 2. *The trace for the optimal unidirectional SFP is a path on the SFP-CG.*

Proof. In Lemma 2, a trace is defined as the set of substrate nodes and links that will be visited from the source to destination in the unidirectional SFP. Since each pair of nodes are connected via the shortest path, no node and link in the SFP-CG will be revisited by the optimal unidirectional SFP. □

Logically, a hybrid SFP can be treated as two inter-related unidirectional SFPs. These two unidirectional SFPs share the same source, destination and common SF instances. Therefore, the hybrid trace (i.e., the trace for a hybrid SFP) in the UFSN can be created by merging the traces of the forward SFP (f-SFP) and backward SFP (b-SFP). With Lemma 2, the hybrid SFP starts and ends at the source node, and visits each link in the hybrid trace exactly once, which is in fact a Eulerian circuit of the hybrid trace [22]. That is to say, when the hybrid trace is given a priori, creating a hybrid SFP is equivalent to construct a Eulerian circuit of the hybrid trace. Thus, we propose the Hybrid Trace Construction (HTC) technique to firstly create the hybrid trace as shown in Algorithm 1. HTC first creates the SFP-CG according to the given SN and NSR. Since the Minimum Spanning Tree (MST) has the least length and connects all substrate nodes, to optimize the latency of the constructed hybrid SFP, HTC takes the MST that includes the required forward/backward SF instances as the trace for the f-SFP/b-SFP. We denote these two MSTs by forward and backward MST, respectively. A hybrid trace is then generated by merging the constructed MSTs. As a connected graph has a Eulerian circuit if and only if every node has even degree (connections) [22], HTC doubles the number of links in the hybrid trace to guarantee the existence of the Eulerian circuit.

Algorithm 1 Hybrid Trace Construction (HTC)

1: **Input:** G, NSR;
2: **Output:** hybrid trace;
3: Discarding the links with less than BW bandwidth resource in G;
4: Generate the SFP Complete Graphs (SFP-CGs) for forward SFs and backward SFs;
5: Construct the Minimum Spanning Tree (MSTs) according to the generated SFP-CGs;
6: Create the hybrid trace by merging the forward and the backward MSTs;
7: **for** Each link in the hybrid trace **do**
8: Create one more link with the same endpoints;
9: **end for**
10: **Return** hybrid trace;

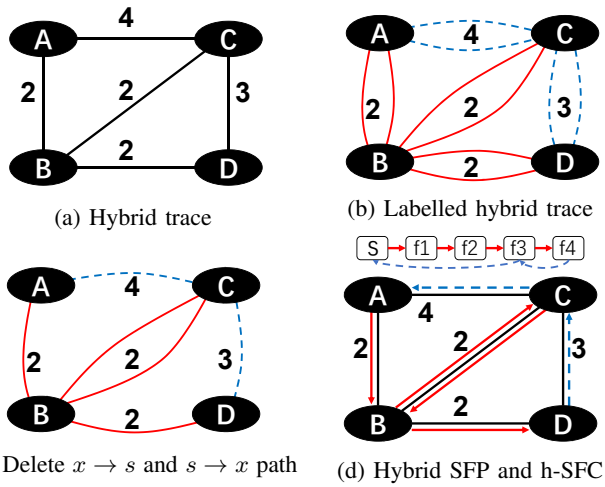
C. Hybrid Eulerian Circuit Construction (HECC)

With the hybrid trace generated from the HTC technique, to construct a Eulerian circuit, we need to visit each link in the hybrid trace exactly once while optimizing latency [22]. To this end, we need to consider: i) the traffic direction for each link (i.e., the link is used for the forward direction or backward direction), ii) the f-SFP ends at which substrate node, and iii) the order of the links to be visited. Accordingly, we introduce the Hybrid Eulerian Circuit Construction (HECC) technique in Algorithm 2, which includes: i) Forward & Backward Link Label (FBLL), ii) Endpoint Determination and iii) Priority Constraint.

Forward & Backward Link Label: Since HTC doubles the number of links in the hybrid trace, the links between each pair of nodes can be either *two-links set* or *four-links set*. For two links set, the FBLL process labels the links with the same direction (i.e., forward or backward) according to the MST that includes this link. That is to say, if a link belongs to the forward MST and is in the two-links set, the FBLL process labels it as forward, vice verse. For four-links set, both endpoints support the SF instance required in both directions. Thus, two links in the four-link set are labelled as forward, while the other two are labelled as backward.

Endpoint Determination: When there is no common SF in both directions, the only substrate node that connects the forward and backward traffic in the hybrid trace is the source node. Thus, the hybrid SFP created from the hybrid trace must firstly visit all required SFs in the f-SFP, go back to the source node, and start the b-SFP. This is also possible for the situation where the forward and backward traffic share some common SF(s). However, additional latency may be added to the constructed hybrid SFP. This is because, the b-SFP can start with the substrate node that hosts a common SF instance instead of going back to the source node. To reduce the latency, the endpoint determination process (i.e., Line 4-9 in Algorithm 2) finds the common SF accommodated by the substrate node x that has the largest sum-distances of the path $s \rightarrow x$ with forward label and path $x \rightarrow s$ with backward label and deletes these two paths.

Priority Constraint: To form a hybrid SFP in the hybrid trace, one needs to start and end at s, while each link is visited exactly once. We can label the link that has involved



(a) Hybrid trace (b) Labelled hybrid trace

(c) Delete $x \rightarrow s$ and $s \rightarrow x$ path (d) Hybrid SFP and h-SFC

Fig. 6: An example of EC-HSFP

in the construction of the Eulerian circuit as "visited". The priority constraint ensures that, in each direction, the node with even number of unvisited forward/backward links has a higher priority to be visited. If both directions share some common SFs, there is only one path connecting s to x with the forward label and x to s with the backward label. Thus, the forward traffic ends at x, while the backward traffic ends at s.

Algorithm 2 Hybrid Eulerian Circuit Construction (HECC)

1: **Input:** NSR, hybrid trace;
2: **Output:** h-SFC, hybrid SFP;
3: Applying **Forward & Backward Link Label** to the hybrid trace;
4: **if** $V_f \cap V_b \neq \emptyset$ **then**
5: Find the the other endpoint x for f-SFP that hosts the SF $f_v \in V_f \cap V_b$ with the longest shortest path from s to x;
6: **else** x is s;
7: **end if**
8: Delete the $s \rightarrow x$ path with forward label;
9: Delete the $x \rightarrow s$ path with backward label;
10: Start with s, visit forward links while considering the **Priority Constraint**;
11: Start with x, visit the backward links while considering the **Priority Constraint**;
12: Record the visiting trace as the hybrid SFP and generate the corresponding h-SFC;
13: **Return** hybrid SFP, h-SFC;

D. Eulerian Circuit based Hybrid SFP optimization

Based on the proposed HTC and HECC techniques, we present the EC-HSFP algorithm in Algorithm 3. As one can see that, the step with the highest time complexity is to generate the SFP Complete Graph (SFP-CG), which needs to run the shortest path algorithm for multi-times. In the worst case ($|V| = |N|$), when applying the shortest path algorithm whose time complexity is $|NL+N^2logN|$, the time complexity of EC-HSFP is $|N^2L + N^3logN|$.

Algorithm 3 Eulerian Circuit based Hybrid SFP optimization (EC-HSFP) Algorithm

1: **Input:** G, NSR;
2: **Output:** h-SFC, hybrid SFP;
3: Create empty sets for hybrid trace, h-SFC and hybrid SFP;
4: hybrid trace = HTC(G, NSR);
5: {hybrid SFP, h-SFC} = HECC(NSR, hybrid trace);
6: **Return** hybrid SFP, h-SFC;

To elaborate how EC-HSFP works, we use Fig. 5 as the input. First, EC-HSFP applies the HTC technique to generate the hybrid trace as shown in Fig. 6a. Next, the HTC technique doubles the number of links in the hybrid trace and applies the HECC technique. During the process of HECC, FBLL labels the link as forward and backward in Fig. 6b, where the red links and blue dotted links represent the forward and backward labels, respectively. Since D supports the common SF instance (i.e., required in both directions) and has the largest sum-distances to the source node A, D is selected as the endpoint x of the f-SFP. After deleting the $A \rightarrow B \rightarrow D$ path with the forward label and $D \dashrightarrow C \dashrightarrow A$ path with the backward label, Fig. 6b is converted to Fig. 6c. Next, HECC technique generates the hybrid SFP and the corresponding h-SFC by visiting the substrate nodes with the priority constraint as shown in Fig. 6d. In the end, the generated hybrid SFP is $A \rightarrow B \rightarrow C \rightarrow B \rightarrow D \dashrightarrow C \dashrightarrow A$, while the h-SFC is $s \rightarrow f1 \rightarrow f2 \rightarrow f3 \rightarrow f4 \dashrightarrow f3 \dashrightarrow s$.

E. EC-HSFP is 2-Approximation

Theorem 2. *EC-HSFP generates the hybrid SFP within a 2-approximation boundary of the optimal hybrid SFP.*

Proof. We denote the length of the forward and backward MSTs by |f-MST| and |b-MST|. The length of the hybrid SFP generated by EC-HSFP is represented as $|\text{SFP}_{\text{EC-HSFP}}|$, while the length of the optimal SFP is denoted by $|\text{h-SFP}_{\text{OPT}}|$. The lengths of the optimal f-SFP and b-SFP are represented as $|\text{f-SFP}_{\text{OPT}}|$ and $|\text{b-SFP}_{\text{OPT}}|$, respectively. The length of the hybrid trace is |HT|. Since MST is the least length connected structure for a connected graph, Eq. (19) and Eq. (20) hold.

$$|\text{f-MST}| \leq |\text{f-SFP}_{\text{OPT}}| \tag{19}$$

$$|\text{b-MST}| \leq |\text{b-SFP}_{\text{OPT}}| \tag{20}$$

Eq. (21) shows the relationship between the optimal hybrid SFP, f-SFP and b-SFP.

$$|\text{f-SFP}_{\text{OPT}}| + |\text{b-SFP}_{\text{OPT}}| \leq |\text{h-SFP}_{\text{OPT}}| \tag{21}$$

According to EC-HSFP, the length of the hybrid trace equals the length sum of f-MST and b-MST as shown in Eq. (22).

$$|\text{f-MST}| + |\text{b-MST}| = |\text{HT}| \tag{22}$$

After doubling the number of links in hybrid trace, the generated hybrid SFP is the Eulerian circuit of the hybrid trace. Therefore, Eq. (23) holds.

$$|\text{SFP}_{\text{EC-HSFP}}| = 2 * |\text{HT}| \tag{23}$$

When x is not s, one needs to delete two paths in the hybrid trace. Thus, from Eq. (19)-(23), we have Eq. (24).

$$|\text{SFP}_{\text{EC-HSFP}}| \leq 2 * |\text{h-SFP}_{\text{OPT}}| \tag{24}$$

Hence, EC-HSFP achieves a 2-approximation boundary. □

Lemma 3. *If the UFSN is a tree structure (e.g., fat tree), EC-HSFP generates the optimal hybrid SFP.*

V. Hybrid SFCE in MFSN

In this section, we extend EC-HSFP to the Multi-Functions Substrate Network (MFSN). The MFSN allows multiple SF nodes mapped onto the same substrate node (subject to the substrate node's computing capacity and the SF instance availability). Fig. 7 and Table I show an example of the MFSN, where the number beside the link represents the link latency. The NSR starts at node B and requires $f2$, $f3$ and $f4$, each of which demands 20 computing resource. Additionally, SFs $f2$ and $f4$ handle the traffic in both directions.

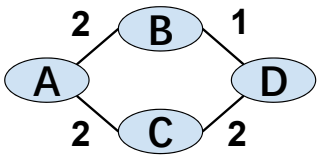


Fig. 7: An example of MFSN

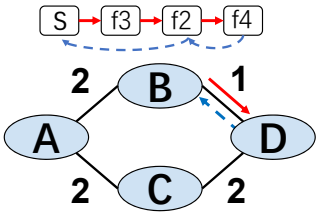


Fig. 8: BC-HSFP result

TABLE I: Multi-Functions Substrate Network

Substrate Node	A	B	C	D
Network Function	$f1, f2, f3$	$f2, f3$	$f3, f5$	$f1, f4$
Capacity	40	40	20	20

To map the SF nodes onto the appropriate substrate nodes in MFSN while reducing the latency cost; we propose an efficient heuristic algorithm called Betweenness Centrality based Hybrid SFP optimization (BC-HSFP), which includes the proposed Betweenness Centrality (BC) based node deployment approach and EC-HSFP.

Betweenness Centrality based Node Deployment: Traditionally, the Betweenness Centrality (BC_n) of a substrate node n can be calculated as $BC_n = \frac{path_n}{path_{total}}$, where $path_n$ and $path_{total}$ represent the amount of the shortest path(s) passing node n and the total number of the shortest path(s) in the graph, respectively. The more shortest paths passing a node, the higher probabilities that this node will connect with other substrate nodes via the shortest path rather than a longer detour. That is to say, a substrate node with the higher BC value will more likely reduce the latency by connecting itself with the other substrate nodes that provide the requested SF instances via the shortest path.

However, this traditional BC technique only considers the connection links between different substrate nodes, which ignores the internal connections within a substrate node in MFSN. In specific, if more than one SF nodes are mapped onto the same substrate node, the connections between these SF instances can also be counted as the potential shortest path(s) passing this substrate node. For example, if $f2$ and $f3$ are embedded onto substrate node B in Fig. 7, then the connection between SF instances $f2$ and $f3$ within node B can also be part of the shortest paths. Here, we create a virtual inner connections between SF instances inside a substrate node, while the outer connection indicates the traditional shortest path passing the substrate node. Accordingly, we propose the Inner-connection-included Betweenness Centrality (IBC) to measure the importance of the substrate node in MFSN, which takes both the inner and outer shortest path connections into account. We specify the number of the inner shortest connections of a substrate node n as $path_{n_{in}}$, while the number of the outer paths is $path_{n_{out}}$. We denote the $\sum_{v \in V} |\{f_v\} \cap \mathbb{F}_n|$ by the number of SF nodes that matches the SF instances installed in the substrate node n and δ_n is the number of SF nodes that can be mapped onto n (limited by its computing capacity). Eq. (25) calculates $path_{n_{in}}$ while Eq. (26) calculates the IBC value of a substrate node n.

$$path_{n_{in}} = min(\sum_{v \in V} |f_v \cap \mathbb{F}_n|, \delta_n) - 1 \tag{25}$$

$$IBC_n = \frac{path_{n_{in}} + path_{n_{out}}}{path_{total}} \tag{26}$$

With the IBC technique, we propose Algorithm 4 to accommodate an NSR with hybrid traffic onto a shared MFSN. Note that, the substrate candidate of an SF node is the substrate node that can provide the corresponding SF instance.

Algorithm 4 Betweenness Centrality based Hybrid SFP optimization (BC-HSFP) Algorithm

1: **Input:** G, NSR;
2: **Output:** hybrid SFP and h-SFC;
3: Initialize the SFP list as an empty list;
4: Calculate the IBC value for substrate nodes that can provide at least one requested SF instance as in Eq. (26);
5: Sort SF nodes in ascending order according to the amount of substrate candidate(s);
6: Map the sorted SF node onto the substrate candidate with the highest IBC value;
7: Generate an induced subgraph G_{induce} including the substrate nodes that host at least one SF nodes;
8: Call EC-HSFP(G_{induce}, NSR) to find the hybrid SFP and h-SFC;
9: Return hybrid SFP and h-SFC;

We use Fig. 7 and 8 to illustrate how the BC-HSFP algorithm works. In Fig. 7, the NSR starts at node B and requires f2, f3 and f4, where f2 and f4 are demanded in both directions. First, the algorithm calculates the IBC value of each substrate node that may participate in the construction of the hybrid SFP as shown in Table II. Then, the algorithm sorts the requested SFs as $\{f4, f2, f3\}$ according to the number of substrate candidate(s). Based on the IBC values shown in Table II and the sorted SF nodes, the algorithm deploys $f4$ onto node D, and then $f2, f3$ onto node B. At last, the algorithm generates the induced graph of B and D in Fig. 7 and calls the EC-HSFP algorithm to form a hybrid SFP as $B \rightarrow D \dashrightarrow B$ with the h-SFC as $s \rightarrow f3 \rightarrow f2 \rightarrow f4 \dashrightarrow f2 \dashrightarrow s$. As shown in Fig.

TABLE II: IBC Value Calculation

Substrate Node	A	B	C	D
# of Inner Path	1	1	0	0
# of Forward Out Path	3	4	3	3
# of Backward Out Path	1	2	0	2
IBC Value	$\frac{5}{10}$	$\frac{7}{10}$	$\frac{3}{10}$	$\frac{5}{10}$

8, the red lines and dashed blue lines represent the forward and backward SFP, respectively. As one can see in Algorithm 4, the process with the highest running time complexity is to calculate the IBC value for substrate nodes that can provide at least one required SF instance. In the worst case, when every substrate node can provide at least one required SF instance, the time complexity of this algorithm is $|N^2L + N^3logN|$.

VI. Numerical Results and Analysis

In this section, we analyze and compare the performance of the proposed algorithms with the schemes that are directly extended from the state-of-art techniques [12], [23].

A. Simulation Environment

We use the 24-nodes US-NET as the Substrate Network (SN), which can be configured as the UFSN or MFSN. For the UFSN, each substrate node supports one unique SF instance, and each physical link has a latency in the range of $[1-5]$. For the MFSN, the number of SF instances supported by a substrate node is in the range of $[2-6]$, while the computing capacity is in the range of $[20-100]$. The link latency in MFSN is in the range of $[1-5]$ and has the bandwidth in the range of $[5-20]$. We set the number of required SF nodes in each NSR within the range of $[4-17]$, while the number of required bidirectional SF nodes is in the range of $[4-13]$. For each SF node, the required computing capacity is in the range of $[10-20]$. The source node is randomly generated and the bandwidth demand is in the range of $[1-10]$.

In [12], the authors proposed the algorithm to map a given unidirectional SFC onto a shared SN with the shortest length when the bandwidth is abundant. We extend the algorithm in [12] as Shortest Path based HSFP optimization ("SP-HSFP") by generating five random hybrid SFCs, creating five corresponding hybrid SFPs and calculating the average latency. In [23], the authors utilized the Closeness-Centrality (CC) technique in the node mapping process, which aims to determine the substrate candidate with the least average latency cost to connect with others. We combine this CC node mapping with EC-HSFP as "CC-HSFP".

B. Performance Metrics

We use the following metrics to assess the performance of the proposed algorithms.

Approximation Ratio (AR): To evaluate the performance of EC-HSFP and SP-HSFP algorithms, we compare their results with the length of the Minimum Spanning Tree (MST), which is proved as the lower boundary in Eq. (19) and (20). AR can be calculated as $AR = \frac{SFP_{created}}{|MST|}$, where the $SFP_{created}$ represents the latency of the hybrid SFP created by the proposed algorithms and $|MST|$ represents the length sum of the forward MST and backward MST.

Average Latency of the Created SFP (ALCS): ALCS is calculated as $ALCS = \frac{\sum_{NSR_i \in NSR_C} SFP_{created}}{|NSR_C|}$, where NSR_C represents a set of NSRs, and NSR_i is the i_{th} NSR in NSR_C.

C. Approximation Analysis in UFSN

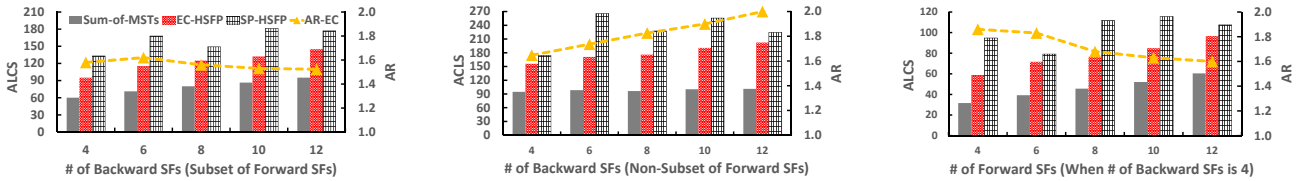
We evaluate the approximate performance of EC-HSFP under three types of NSR (i.e., Fig. 9a, 9b, 9c). In Fig. 9a, 12 forward SFs are required and the backward SFs are a subset of the forward SFs (i.e., $V_b \subseteq V_f$). In Fig. 9b, 12 forward and backward SFs are required, while the number of backward SFs that does not belong to the forward SFs varies. In Fig. 9c, 4 backward SFs are required and the number of forward SFs varies. In Fig. 9, the grey, red and dark gridded bars represent the ALCS of MSTs, EC-HSFP and SP-HSFP, respectively. The yellow dashed curve denotes the AR for EC-HSFP.

When increasing the number of backward SFs that belong to the set of required forward SFs in Fig. 9a, EC-HSFP and MSTs need more latency to finish the transmission but the latency required by SP-HSFP fluctuates. This is because the performance of the SP-HSFP totally depends on the given SFCs. Due to the given SFCs, SP-HSFP may take multiple detours leading to fluctuated latency for the constructed hybrid SFP. For EC-HSFP, the AR is under 2 in any situation, which matches Theorem 2. It is worth noting that, when the number of backward SFs is small (e.g., 4), the lengths of the deleted paths (i.e., $s \rightarrow x$ forward path and $x \rightarrow s$ backward path) may not contribute much to the total latency. Thus, the AR is relatively high when the number of backward SFs is in the range of $[4, 6]$. However, when further increasing the number of backward SFs, there is a higher probability that the $x \rightarrow s$ forward path and $s \rightarrow x$ backward path become longer. Particularly, when the number of backward SFs is bigger than 10, the deleted paths most likely are the longest one in the generated MSTs. Therefore, the AR is relatively low and the proposed EC-HSFP performs even better when the number of backward SFs is larger than 6.

When the number of backward SFs increases in Fig. 9b, EC-HSFP needs more latency while the sum lengths of MSTs do not vary much. Again, since the performance of SP-HSFP totally depends on the constructed SFCs, it is unstable in UFSN. It is worth noting that, when the number of backward SFs that does not belong to the set of forward SFs equals 12, the AR is 2. This is because the only common substrate node that the forward MST and backward MST share is the source node; thus no path will be deleted, whereas $|\text{SFP}_{\text{EC-HSFP}}| = 2 * |HT| = 2 * (|\text{f-MST}| + |\text{b-MST}|)$.

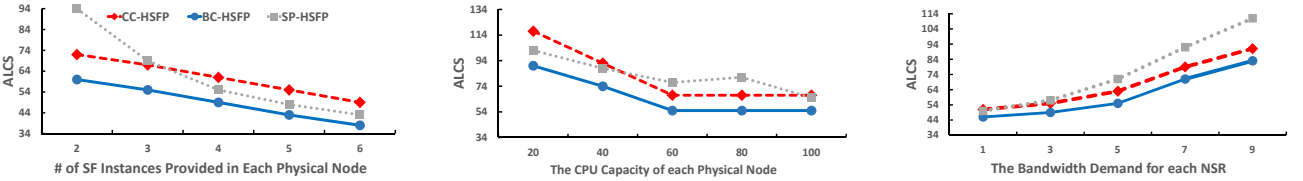
In Fig. 9c, when increasing the number of forward SFs, EC-HSFP and the sum lengths of MSTs increase. When the number of forward SFs is small (e.g., 4), the probability that the backward SFs belong to the forward SFs is small. Thus, the common node that the forward and backward MSTs share likely is the source node, which results in a high AR. When increasing the number of forward SFs, the probability that the backward SFs are included in the forward SFs increases, which reduces the latency of the constructed hybrid SFP by potentially increasing the length of the deleted paths.

Overall, the AR is no more than 2 in any situation, which verifies that the proposed EC-HSFP algorithm guarantees the 2-approximation performance.



(a) Backward SFs $\subseteq$ Forward SFs (b) Backward SFs $\not\subset$ Forward SFs (c) Fixed # of Backward SFs

Fig. 9: Approximation analysis in UFSN



(a) # of SF instances (b) # of computing resource (c) # of bandwidth demand

Fig. 10: Performance analysis in MFSN

D. Performance Analysis in MFSN

Fig. 10 shows the performance of CC-HSFP, BC-HSFP and SP-HSFP in MFSNs. In Fig. 10, the red dashed curve, grey dotted curve and blue solid curve represent the performance of CC-HSFP, SP-HSFP and BC-HSFP, respectively.

Fig. 10a demonstrates that the latency from all three schemes decreases when increasing the number of SF instances in each substrate node. This is because the probability to generate the inner connection becomes larger when the SF instances provided in each substrate node are more. Note that, BC-HSFP outperforms both SP-HSFP and CC-HSFP. This is because BC-HSFP can jointly optimize SFC composition and embedding, whereas SP-HSFP is limited by the given SFCs. The CC technique in CC-HSFP will likely create the MST as a "star". This is because the central node of a graph has the highest CC value, which implies it is the nearest one to other substrate nodes that host the required SFs. Thus, the MST is likely to be created by connecting the central node with the other substrate nodes via the shortest path, which is a "star". However, from the experimental results, the BC technique will likely create the MST as a "path". Fig. 11 and 12 show an example of the difference between "star" and "path" topologies. For a star MST in Fig. 11, the process in Line 8-9 of Algorithm 2 will delete the paths $A \rightarrow D$ and $D \dashrightarrow A$, removing 4 hops out of final hybrid SFP. However, when a path MST is constructed by BC as shown in Fig. 12, the process in Line 8-9 of Algorithm 2 will remove 8 hops, resulting in a shorter hybrid SFP. This difference can be even larger in an SN with more nodes.

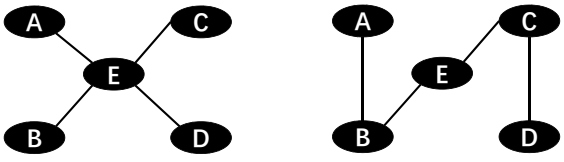


Fig. 11: A star MST from CC Fig. 12: A path MST from BC

When the number of SF instances provided by each substrate node is 3, Fig. 10b demonstrates that the more computing resource provided by each substrate node, the less latency is required for all algorithms. The performance of BC-HSFP and CC-HSFP is flat when the computing resource provided by each substrate node is larger than 60. This is because 60 computing capacity will allow that all SF instances are available in each substrate node. Thus, further increasing the number of computing capacity does not change the performance of CC-HSFP and BC-HSFP.

Fig. 10c shows the performance of the proposed algorithms when varying the number of bandwidth demand. As one can see that, the more bandwidth requested by the NSR, the more latency is required. This is because when increasing the bandwidth demand, some links become unavailable, and a longer path may have to be employed. Note that, CC-HSFP and BC-HSFP increase slower than SP-HSFP. This is because, as a joint h-SFC composition and embedding process, the routing technique (i.e., EC-HSFP) of BC-HSFP/CC-HSFP does not introduce multi-detours that bring large latency in a bandwidth resource-limited network.

Overall, the proposed BC-HSFP algorithm averagely outperforms CC-HSFP by 20% and outperforms SP-HSFP as much as 50%.

VII. Conclusion

In this paper, for the first time, we have comprehensively studied a new set of *Hybrid SFC composition and Embedding* (HSFCE) problems in different network scenarios. When the computing capacity provided by the Substrate Network (SN) is limited, we have investigated the Unique Function SN (UFSN), where each substrate node only provides one unique type of SF. We have proved the NP-hardness of HSFCE in UFSN and proposed a 2-approximation algorithm to jointly composite and embed a hybrid SFC, called Eulerian Circuit based Hybrid SFP optimization (EC-HSFP). We have also studied HSFCE in Multi-Functions SN (MFSN), where each substrate node provides various SFs, and extended the EC-HSFP with the betweenness centrality technique to optimize HSFCE in MFSN. Our extensive analysis and simulation results have shown that the EC-HSFP algorithm can guarantee the 2-approximation boundary, and the proposed BC-HSFP outperforms the algorithms directly extended from the state-of-art techniques by an average of 20%.

REFERENCES

[1] B. Yi, X. Wang, K. Li, S. K. Das, and M. Huang, "A Comprehensive Survey of Network Function Virtualization," *Comput. Netw.*, vol. 133, pp. 212–262, 2018.

[2] J. G. Herrera and J. F. Botero, "Resource Allocation in NFV: A Comprehensive Survey," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 518–532, 2016.

[3] D. Zheng, E. Guler, C. Peng, G. Luo, L. Tian, and X. Cao, "Dependence-Aware Service Function Chain Embedding in Optical Networks," in *IEEE ICC*, 2019, pp. 1–6.

[4] P. Quinn and T. Nadeau, "Problem Statement for Service Function Chaining," https://tools.ietf.org/html/rfc7498, 2015.

[5] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual Network Embedding: A Survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 2013.

[6] A. J. Ferrer, J. M. Marquès, and J. Jorba, "Towards the Decentralised Cloud: Survey on Approaches and Challenges for Mobile, Ad hoc, and Edge Computing," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 111:1–111:36, 2019.

[7] J. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture," https://tools.ietf.org/html/rfc7665, 2015.

[8] W. Zhou, Y. Yang, M. Xu, and H. Chen, "Accommodating Dynamic Traffic Immediately: a VNF Placement Approach," in *IEEE ICC*, 2019, pp. 1–6.

[9] A. Boubendir, E. Bertin, and N. Simoni, "A VNF-As-A-Service Design through Micro-Services Disassembling the IMS," in *ICIN*, 2017, pp. 203–210.

[10] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near Optimal Placement of Virtual Network Functions," in *IEEE INFOCOM*, 2015, pp. 1346–1354.

[11] A. Tomassilli, F. Giroire, N. Huin, and S. Pérennes, "Provably Efficient Algorithms for Placement of Service Function Chains with Ordering Constraints," in *IEEE INFOCOM*, 2018, pp. 774–782.

[12] G. Sallam, G. R. Gupta, B. Li, and B. Ji, "Shortest Path and Maximum Flow Problems Under Service Function Chaining Constraints," in *IEEE INFOCOM*, 2018, pp. 2132–2140.

[13] L. Guo, J. Z. T. Pang, and A. Walid, "Joint Placement and Routing of Network Function Chains in Data Centers," in *IEEE INFOCOM*, 2018, pp. 612–620.

[14] J.-J. Kuo, S.-H. Shen, H.-Y. Kang, D.-N. Yang, M.-J. Tsai, and W.-T. Chen, "Service chain embedding with maximum flow in software defined network and application to the next-generation cellular network architecture," in *IEEE INFOCOM*, 2017, pp. 1–9.

[15] M. C. Luizelli, D. Raz, and Y. Sa'ar, "Optimizing NFV Chain Deployment through Minimizing the Cost of Virtual Switching," in *IEEE INFOCOM*, 2018, pp. 2150–2158.

[16] Z. Zheng, J. Bi, H. Yu, H. Wang, C. Sun, H. Hu, and J. Wu, "Octans: Optimal Placement of Service Function Chains in Many-Core Systems," in *IEEE INFOCOM*, 2019, pp. 307–315.

[17] P. K. Vairam, G. Mitra, V. Manoharan, C. Rebeiro, B. Ramamurthy, and K. Veezhinathan, "Towards Measuring Quality of Service in Untrusted Multi-Vendor Service Function Chains: Balancing Security and Resource Consumption," in *IEEE INFOCOM*, 2019, pp. 163–171.

[18] D. Zheng, C. Peng, X. Liao, G. Luo, L. Tian, and X. Cao, "Service Function Chaining and Embedding with Spanning Closed Walk," in *IEEE HPSR*, 2019, pp. 1–5.

[19] D. Dolson, M. Boucadair, S. Homma, and D. Lopez, "Hierarchical Service Function Chaining (hSFC)," https://tools.ietf.org/html/rfc8459, 2018.

[20] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 2017.

[21] F. Lam and A. Newman, "Traveling Salesman Path problems," *Mathematical Programming*, vol. 113, no. 1, pp. 39–59, 2008.

[22] D. B. West *et al.*, *Introduction to Graph Theory*. Prentice hall Upper Saddle River, NJ, 1996, vol. 2.

[23] E. Guler, D. Zheng, G. Luo, L. Tian, and X. Cao, "Virtual Multicast Tree Embedding over Elastic Optical Networks," in *IEEE GLOBECOM*, 2017, pp. 1–6.