A Framework for 5G Network Slicing Optimization using 2-Edge-Connected Subgraphs for Path Protection

Original Scientific Paper

Igor Begić*
JP ELEKTROPRIVREDA HZ HB d. d. Mostar,
Development Division, Department of Telecommunications
Ulica kralja Petra Krešimira IV, 6-A, Mostar, Bosnia and Herzegovina
igor.begic@ephzhb.ba

Adrian Satja Kurdija
University of Zagreb,
Faculty of Electrical Engineering and Computing,
Department of Electronics, Microelectronics, Computer and Intelligent Systems
Unska 3, Zagreb, Croatia
adrian.kurdija@fer.hr

Željko Ilić
University of Zagreb,
Faculty of Electrical Engineering and Computing,
Department of Telecommunications
Unska 3, Zagreb, Croatia
zeljko.ilic@fer.hr

*Corresponding author

Abstract *– Emerging telecommunications technologies require robust frameworks for efficient network slicing. We propose a network-slicing model that aims to optimize the deployment of virtual networks on a physical network topology. Our model ensures compliance with 5G requirements, incorporating latency and capacity constraints on virtual links. Selecting slices with cost and resource requirements on the computing nodes is optimized using a Knapsack problem with revenue maximization. We propose a path protection algorithm to deal with link failures by constructing a 2-edge-connected subgraph (or two link-disjoint Steiner trees) for each slice to provide both primary and backup paths. Simulation results include comparison with existing solutions by metrics such as latency, revenue, resource utilization, number of protected slices, and computation time, providing valuable insights for network planners operating in diverse and dynamic environments. Key contributions include efficient resource allocation using the Knapsack problem, enhanced network resilience via 2-edge-connected subgraphs for path protection, and realistic simulation experiments on SNDlib dataset topologies. The simulation results show that the proposed framework improves computational efficiency compared to the recent related solutions, particularly in large network topologies where k-connected function slicing (KC-FS) subgraph embeddings take approximately 3.5 times more computation time.*

Keywords: *network slicing, resource allocation, virtual networks, optimization model, 5G networks, path protection*

Received: April 15, 2024; Received in revised form: July 4, 2024; Accepted: July 4, 2024

1. INTRODUCTION

5G technology has ushered in unprecedented challenges and opportunities in the rapidly evolving telecommunications landscape. As the demand for diverse and high-performance services intensifies, robust frameworks to efficiently manage network resources become crucial. Network slicing, a pivotal concept in the realm of 5G, offers a promising avenue for achieving this efficiency by allowing the creation of virtual networks tailored to specific service requirements [1].

Ensuring compliance with 5G requirements, particularly regarding latency and capacity constraints on virtual links, becomes critical. The telecommunications industry is actively developing models and frameworks that address these challenges. Solutions that optimize the deployment of virtual networks contribute to the overall efficiency and effectiveness of 5G networks [2, 3].

As the demand for 5G services continues to escalate, network operators are confronted with strategically selecting and deploying network slices that comply with stringent cost and resource constraints on computing nodes and maximize revenue potential [4]. The cost-effective utilization of resources is pivotal to ensuring the economic viability of 5G networks, making optimization strategies a focal point in current research and development efforts [5].

Furthermore, in the 5G networks, the seamless and uninterrupted connectivity with this advanced technology promises is contingent upon robust mechanisms for handling network link failures. As 5G networks evolve to support diverse services (e.g., eMBB, uRLLC, and mMTC) and applications with stringent reliability requirements, the industry has recognized the imperative of implementing protection algorithms to safeguard against disruptions caused by link failures. The telecommunications industry has turned to solutions such as dedicated path protection algorithms to mitigate this impact. This approach involves constructing both primary and backup paths, ensuring that communication is seamlessly transmitted to the pre-established backup path in case of a link failure on the primary path [6, 7].

This paper proposes a network-slicing model designed to optimize the deployment of virtual networks on a physical network topology. Our model is crafted to ensure compliance with 5G requirements, addressing crucial aspects such as latency and capacity constraints on virtual links. The model integrates latency constraints on virtual links, ensuring that the constructed subgraphs and selected paths minimize latency, which is essential for meeting the low latency requirements of 5G real-time communications. It enhances network reliability by constructing 2-edge-connected subgraphs, ensuring that an alternative path remains available between any pair of nodes even if a single link fails, thus maintaining uninterrupted service.

By integrating cost and resource requirements considerations on computing nodes, we employ a Knapsack problem with revenue maximization to optimize the selection of slices, striking a balance between resource utilization and service quality. By formulating the slice selection problem as a Knapsack problem, the framework optimizes the allocation of resources (CPU units, memory, etc.) on computing nodes, maximizing revenue while adhering to the nodes' capacity constraints, thereby ensuring efficient use of available resources.

Recognizing the dynamic nature of telecommunications environments, our proposed model employs two variants of path protection. Our primary proposed approach for each slice constructs a subgraph with the 2-edge-connectivity property, which means that at least two distinct paths exist between each pair of slice's nodes. The alternative approach leverages the construction of two link-disjoint Steiner trees for each slice: one for primary paths and the second for backup paths. Both methods are tailored to handle link failures efficiently, providing a robust mechanism for ensuring continuous service availability and mitigating potential disruptions.

We comprehensively evaluated the proposed models through extensive simulations using network topologies from the SNDlib dataset [8]. Metrics such as cost, latency, accepted slices, and resource utilization are systematically analyzed to provide network planners with valuable insights into the performance and resilience of the network slicing framework. These insights are crucial in diverse and dynamic environments where telecommunication networks must adapt to varying demands and unexpected challenges.

This paper aims to contribute to the scientific discourse surrounding optimizing 5G networks, grappling with the complexity of modern and evolving network landscapes. By addressing deployment efficiency, compliance with stringent requirements, and resilience in the face of failures, our model seeks to advance state-of-the-art telecommunications technologies, offering tangible solutions for network planners.

The primary contributions of this paper encompass the following:

- The paper introduces a novel framework for efficient resource allocation and path protection in 5G network slicing deployment using the Knapsack problem and 2-edge-connected subgraphs.
- The paper shows the proposed framework's computational efficiency compared to the recent related solutions through extensive simulations with realistic network topologies.

The paper is organized as follows: Section II discusses the related works; Section III presents our system model and defines the problem in mathematical terms; Section IV presents the proposed algorithms; Section V presents simulation results. Finally, the conclusion of the paper is presented is Section VI.

2. RELATED WORK

Many papers considered 5G network slicing aspects, attempting to solve optimization problems based on the system model and the assumed constraints and requirements.

To name a few related approaches,the authors in [9] used the Mul tiple Choice Knapsack Problem to address the problem of maximizing the slice allocation in the access network, i.e., a macro cell. In our paper, the 0-1 Knapsack Problem is adapted to find an optimal subset of network slices in each computing node, balancing the trade-off between revenue maximization and adherence to resource constraints.

Recently, the authors in [5] proposed a nested decomposition model for reliable slicing of Virtualized Network Functions (VNFs) running on virtual computing nodes, providing dedicated path protection with primary and backup paths being link-disjoint. Because of the enormous problem size, their solution used integer linear programming (ILP) with heuristic column generation. In [7], the authors employ a heuristic to solve a path-based ILP with bandwidth squeezing and multi-path provisioning. In another recent approach [6], latency-constrained paths between source and target nodes are considered and selected from the shortest to the longest in terms of hops.

In [10], the authors proposed a model that integrates computing and networking resources to minimize costs for the Slice Broker (SB) when purchasing resources from Infrastructure Providers (InPs). In [11], the authors introduce the Topology-Level Based Protection Scheme (TLPS) as an innovative method to improve reliability and minimize redundancy in multi-domain networks. TLPS customizes virtual networks (VNs) to meet precise reliability needs while reducing backup redundancy. It accomplishes this by crafting tailored VNs and deploying them onto the substrate network through a mixed integer linear programming (MILP) model and heuristic approach. In [12], the authors proposed an advanced approach for optimizing VNF allocation within dynamic network slices. By introducing a multilayered Service Function Chain (SFC) formation and utilizing an ILP model, the research addresses the VNF-embedding and allocation problem (VNF-EAP) on a real-world AT&T network topology.

In [13], the authors presented a hierarchical identifier (HID) 5G architecture that simultaneously supports network slicing and SFC. The HID incorporates network slice selection assistance information (NSSAI) and service path ID (SPI), allowing users to attach to specific network slices and enabling the sequential processing of flows by service functions (SFs). To mitigate degradation in quality of service (QoS) caused by unexpected increases in flow to a slice, a backup slice shared among different services is introduced to utilize network resources efficiently. In [14], the authors examined the reliability of 5G transport network slices within elastic optical networks (EON). The primary focus is slicing 5G transport networks, which involves setting up virtual networks on 5G transport infrastructure while ensuring dedicated protection.

Most existing methods solve the Virtual Network Embedding (VNE) problem by first embedding all nodes without considering link embedding information. However, such approaches can lead to suboptimal link embedding due to decisions made without explicitly considering the requirements and characteristics of the links. This can lead to inefficient resource utilization and revenue loss for service providers.

Our approach differs from the conventional approaches that model each slice as a *chain* of virtual network functions (VNFs) implemented by the computing nodes (e.g., [5, 12]). Instead of assuming a fixed order of VNFs within a slice, our approach is more flexible: it only requires that all corresponding computing nodes are connected by paths, allowing for different orderings (chains) of VNFs within the slice. Therefore, instead of constructing separate paths for each pair of consecutive computing nodes in a slice, we build a 2-edge-connected subgraph spanning all computing nodes with minimal total latency of the used links. Path protection results from the 2-edge-connectivity property: the subgraph remains connected whenever an edge is removed, corresponding to a failed link. This approach is compared to our proposal [15], where two link-disjoint Steiner trees spanning all computing nodes are constructed, one for primary paths and the other for backup paths. It is also compared to an approach recently proposed in [16], where each service function (SF) is embedded as a *k-edge*-connected subgraph using the k-connected function slicing (KC-FS) algorithm. When $k = 2$, the KC-FS algorithm can be used for slice subgraph construction in the scenario presented here. However, the simulation results demonstrate that it is computationally much more demanding. Also, in [16] the authors do not consider selecting nodes for an SF: it assumes that the set of node instances (N_{fi}) for the i^{th} SF is given as part of the input. The two link-disjoint trees might not exist for network topologies with low redundancy, so the present approach is more suitable, as we will demonstrate. Another key difference from conventional approaches is that instead of using ILP formulations, which can be inefficient for large problem sizes, we use the classical algorithms for Knapsack and Steiner Tree problems, making the algorithm more straightforward to implement and guaranteeing its polynomial time complexity.

In [17], the authors highlighted the importance of virtualization in modern computing and its pervasive presence across various computing domains, emphasizing the need for efficient management of physical resources. The paper introduced the Virtual Network Embedding (VNE) problem, which entailed allocating physical resources to meet virtual resource requests while adhering to constraints and maximizing resource utilization. It presented the Improved Virtual Network Embedding using Conflict-Based Search (iVNE-CBS) as an improved algorithm for addressing the VNE problem.

In [18], the authors formulated the problem of Service Function Chain (SFC) scheduling in NFV-enabled 5G networks as a mixed integer non-linear programming challenge. They aimed to maximize the number of requests meeting latency and reliability constraints in a dynamic network environment where SFC requests arrive randomly. Additionally, the authors proposed an efficient algorithm to decide VNF redundancy while minimizing delay.

In [19], the authors tackled the challenge of optimally placing SFCs within 5G network slices, considering resource distribution across Edge and Cloud sites. They introduced solutions for the Virtual Network Functions Chain Placement Problem (VNF-CPP) using integer linear programming (ILP) and heuristic algorithms, addressing constraints like end-to-end delay, processing delay, VNF affinity, and traffic requirements between VNFs.

In [20], the authors proposed a virtual network embedding algorithm model incorporating a graph attention mechanism and a multi-layer perceptron. This model determines the weights between nodes by introducing the attention mechanism to measure the relationships between nodes. The proposed models and algorithms effectively reduced resource fragmentation and improved the acceptance rate of virtual network requests.

In [21], the authors employed a reinforcement learning (RL) approach to address the embedding problem. The approach integrated shareable virtual network functions into an existing RL scheme designed for virtual node embedding, achieving this with minimal additional computation.

3. FRAMEWORK OVERVIEW

In our framework, we focus on virtual computing nodes connected by links in the network topology. We optimize the slice selection in the computing nodes and the construction of slices' subgraphs in the network topology graph.

As an illustration of our Steiner-tree approach [15], Fig. 1 depicts the nodes selected for a specific slice (nodes 4, 8, and 9, in red color) along with the primary tree (red links) and the backup tree (blue links) connecting these selected nodes. Backup links can still connect each pair of the selected nodes in case of any primary link failure.

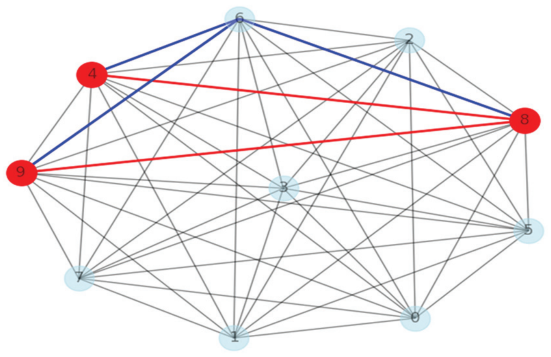

Fig. 1. Primary (red) and backup (blue) trees spanning the slice embedded in the nodes 4, 8, and 9

However, this paper aims for more efficient path protection. Instead of two spanning trees, one being able to replace the other, we will construct a single spanning subgraph, which is "larger" than one tree but "smaller" than two trees (in terms of the number of links). The subgraph must have the following property to allow for link failures: whenever a single edge from the subgraph is removed, there is still a path between any pair of its nodes. Therefore, the subgraph provides both primary and backup paths between the nodes. The required property of such a graph is called *2-edge-connectivity*.

To illustrate the difference between the two proposals, consider Fig. 1 and 2. In Fig. 1, the (red) primary and (blue) backup trees contain five edges. In Fig. 2, however, the single (red) subgraph includes four edges serving the same slice. The subgraph has the required property: if any of its edges is removed, it still contains paths between all its slice nodes (4, 8, and 9). For instance, if the (primary) link between nodes 4 and 8 fails, there is a backup path: 4-6-9-8. In terms of graph theory, the subgraph in Fig. 3 is 2-edge-connected. It is more efficient since it provides both primary and backup paths between its nodes and requires fewer resources than the two Steiner trees in Fig.1.

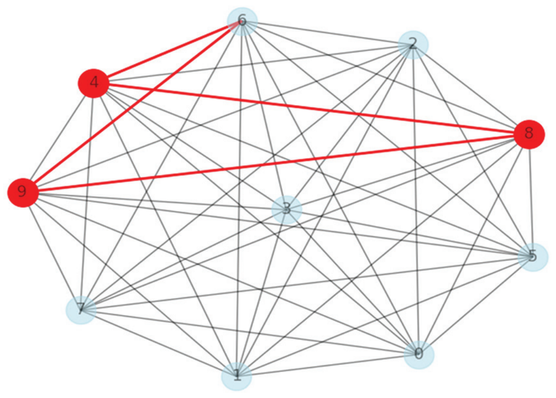

Fig. 2. A 2-edge-connected subgraph spanning the slice embedded in the nodes 4, 8, and 9, serving both primary and backup paths

This difference is significant in networks with low redundancy, where the number of disjoint paths between the nodes is small. In such networks, the previous proposal might not be able to construct two disjoint Steiner trees spanning each slice. For instance, Fig. 3 depicts the primary tree constructed for a slice spanning the nodes 1, 5, 7, and 8 in a topology with low redundancy ("*abilene*" *network topology from the* SNDlib dataset [11]). In this network, there is no way to construct a backup tree spanning the same nodes while avoiding the links of the primary tree. Therefore, this slice would not have path protection. On the other hand, the path protection can be achieved if a 2-edge-connected subgraph is constructed instead of a tree, serving both primary and backup paths, as depicted in Fig. 4. Such 2-edge-connected subgraph is inherently easier to construct than two spanning trees, which in most cases require more links and the existence of disjoint paths (high redundancy). The evaluation will present a numerical comparison (Sect. 5).

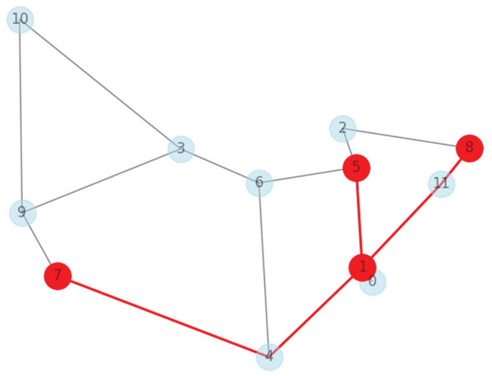

Fig. 3. Primary Steiner tree spanning the slice embedded in the nodes 1, 5, 7, and 8. A link-disjoint backup tree cannot be constructed (no path protection).

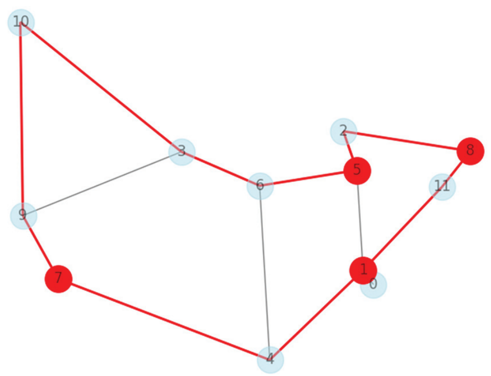

Fig. 4. Path protection of the slice embedded in the nodes 1, 5, 7, and 8 using a 2-edge-connected subgraph, allowing for a single link failure

To formalize the proposed framework, let N be the set of computing nodes, S the set of slices, and E the set of bidirectional links between node pairs (i, j). We assume that each computing node i has R_i available resource elements (representing CPU units, memory units, available energy, or any other resource measure).

We assume that a slice s requires $R_{s,i}$ resource elements when running on node i if those resources are allocated to s, in other words, if s is embedded at i. We assume that the fixed cost of embedding slice s at node i is $F_{s,i}$ and that revenue generated per unit of resource allocated to slice s equals Us (Table 1).

Table 1. Notation

Symbol	Meaning	Symbol	Meaning
N	set of computing nodes	E	set of bidirectional links $(i, j) \in N \times N$
S	set of slices	$L_{i,j}$	latency of link (i, j)
R_i	available resource elements of node i	$C_{i,j}$	capacity of link (i, j)
$R_{s,i}$	resource requirement of slice s at node i	$R_{s,i,j}$	capacity requirement for slice s on link (i, j)
$F_{s,i}$	cost of embedding slice s at node i	N_s	set of nodes selected for slice s
U_s	revenue per unit of resource of slice s	P_s	links of the primary tree for slice s
$x_{s,i}$	whether slice s is embedded at node i	B_s	links of the backup tree for slice s
$y_{s,i,j}$	whether link (i, j) is included in the primary subgraph for slice s	$z_{s,i,j}$	whether link (i, j) is included in the backup tree for slice s

We can now formalize the problem of slice selection in each computing node. Let $x_{s,i} \in \{0,1\}$ be a binary decision variable indicating whether slice s is allocated resources at node i.

For each node $i \in N$, maximize the total slices' revenue:

$$\sum_{s \in S} x_{s,i} \left(R_{s,i} U_s - F_{s,i}\right) \qquad (1)$$

With respect to available resources:

$$\sum_{s \in S} x_{s,i} R_{s,i} \leq R_i \qquad (2)$$

Where,

$x_{s,i}$	Binary decision variable indicating whether slice s is allocated resources at node i
$R_{s,i}$	Resource requirement of slice s at node i
U_s	Revenue per unit of resource of slice s
$F_{s,i}$	Cost of embedding slice s at node i
S	Set of slices
R_i	Available resource elements of node i.

After the slice selection variables xs,i are determined, the set of slice nodes is defined as the set of all nodes where slice s is embedded:

$$N_s = \{i \in N : x_{s,i} = 1\} \qquad (3)$$

Where,

N_s	Set of nodes selected for slice s
N	Set of computing nodes.

We assume that each link $(i, j) \in E$ has known latency $L_{i,j}$ and capacity $C_{i,j}$. Furthermore, if this link is embedded for slice s, each slice has a capacity requirement $R_{s,i,j}$.

We now jointly define the problems of path selection and path protection. For each slice s, a subgraph $P_s (P_s \subset E)$ should be constructed so that all nodes in N_s are connected (directly or indirectly) by links from P_s. This subgraph should be 2-edge-connected to enable path protection in case of a single link failure. Alternatively, it can be a tree, and then it should be augmented with another backup tree $B_s \subset E$, which ensures path protection by providing backup paths in case of a single link failure in P_s. While constructing the subgraphs, the capacity constraints for each link should be met, and total latency should be minimal. We can formalize the path selection and path protection problem as follows.

$$\sum_{s\in S}\sum_{(i,j)\in E} y_{s,i,j}L_{i,j} \tag{4a}$$

$$\sum_{s\in S} y_{s,i,j}R_{s,i,j} \leq C_{i,j} \tag{4b}$$

Where,

$L_{i,j}$	Latency of link (i, j)
$y_{s,i,j}$	a binary decision variable indicating whether link (i, j) is included in the primary subgraph for slice s
$R_{s,i,j}$	The capacity requirement for slice s on link (i, j)
$C_{i,j}$	The capacity of link (i, j).

These equations correspond to a choice of 2-edge-connected subgraphs for P_s. Therefore, in the rest of the paper, P_s is the set of links (i, j) for which $y_{s,i,j} = 1$. In the case of trees, there are two subgraphs, so we modify the equations as follows.

Minimize:

$$\sum_{s\in S}\sum_{(i,j)\in E} (y_{s,i,j} + z_{s,i,j})L_{i,j} \tag{5a}$$

$$\sum_{s\in S} (y_{s,i,j} + z_{s,i,j})R_{s,i,j} \leq C_{i,j} \tag{5b}$$

Where $z_{s,i,j}$ is a binary decision variable indicating whether link (i, j) is included in the backup tree for slice s. Therefore, in the rest of the paper, B_s is the set of links (i, j) for which $z_{s,i,j} = 1$.

4. PROPOSED ALGORITHMS

The proposed framework is divided into two steps. The first step performs the slice selection in the computing nodes, determining the decision variables $x_{s,i}$. The second step performs the construction of slices' subgraphs. In both steps, we consider three variants of the proposed framework. Two of them construct primary and backup Steiner trees and are included mostly for evaluation and comparison purposes, while the third variant is our proposed approach based on 2-edge connectivity. More specifically:

- Steiner-balanced algorithm seeks to optimize the total revenue, as well as the number of accepted slices, i.e., the number of slices for which the primary tree is successfully constructed.
- Steiner-protection algorithm seeks to maximize the number of selected slices per node, as well as the number of protected slices, i.e., the number of slices for which both the primary tree and the backup tree are successfully constructed.
- Proposed variant constructs a 2-edge-connected subgraph for each slice instead of two Steiner trees, while also maximizing total revenue.

4.1. SELECTING SLICES ON THE COMPUTING NODES

In the first step, since $x_s, i \in \{0,1\}$, we reduce the problem to a classical 0-1 Knapsack Problem and solve it by the corresponding dynamic programming algorithm.

Namely, for each computing node i, we consider all slices as potential items in the Knapsack with weight capacity R_i. Following Eq. (1), for an item corresponding to slice s, its value v_s equals the revenue $R_{s,i}\,U_s$-$F_{s,i}$, while its weight w_s equals its resource requirement $R_{s,i}$. The goal of the 0-1 Knapsack Problem is to select items (without repetitions) to put in the knapsack to maximize the total value of selected items without exceeding the knapsack's weight capacity. If the corresponding item gets selected in the knapsack solution, then $x_{s,i} = 1$.

We solve the 0-1 Knapsack Problem independently for each node using the standard dynamic programming algorithm. The key idea is to build a 2D array dp where $dp[k][w]$ represents the maximum value that can be achieved with the first k items and a knapsack capacity of w. We iterate through the items, updating the array based on whether we exclude or include the current k-th item with value v_s and weight w_s (if $w \geq w_s$):

$$dp[k][w] = max\begin{cases} dp[k-1][w] & \text{(exclude)} \\ v_s + dp[k-1][w-w_s] & \text{(include)} \end{cases} \tag{6}$$

To extend the solution to return which items are selected, we need to modify the algorithm to track the decisions made during the construction of the dp table. This reconstruction can be done by examining the dp table after it has been filled out and tracing back the choices that led to the maximum value.

The algorithm complexity is $O(|S| \cdot |R_i|)$ for each of the $|N|$ computing nodes. We thus obtain selected slices at each node and calculate the sets N_s using Eq. (3).

The described solution is used in the Proposed and Steiner-balanced variants. In the Steiner-protection variant, the 0-1 Knapsack is solved greedily by repeatedly selecting the item with the lowest weight, corresponding to the slice with the lowest resource requirement. This approach maximizes the number of slices embedded in the node.

After the computing nodes N_s for each slice have been determined, they should be connected by paths with respect to the latency requirements and capacity constraints. Namely, we are searching for a set of links $P_s \subset E$ spanning all nodes in N_s with minimal total latency, which is the second step of the algorithm. The following two subsections describe the considered variants of this step, differing in the chosen structure of P_s.

4.2. PATH CONSTRUCTION: STEINER TREE APPROACHES

A natural idea is for P_s to be a tree because such a subgraph uses the smallest number of edges. In that case, however, path protection in case of a link failure requires one more tree for each slice – a backup tree B_s, which is link-disjoint with P_s ($P_s \cap B_s = \emptyset$).

In a given graph, the construction of a tree that spans a given subset of nodes (terminals) is a standard *Steiner tree problem in graphs*. Its optimization variant which minimizes the total weight of the selected tree's edges (corresponding to link latencies $L_{i,j}$) can be solved by e.g. Kou's algorithm [22] which is implemented in a NetworX library [23]. For this algorithm, we consider only the links with enough capacity ($C_{i,j} \geq R_{s,i,j}$). In this way, we obtain the primary tree P_s for each slice. Afterward, we construct the backup tree B_s in the same manner, considering only the links not selected in P_s to ensure link disjointness. The algorithm complexity is $O(|N_s| \cdot |E|2)$ for each of the $O(|S|)$ Steiner trees.

Whenever a primary or a backup tree is constructed, we must update the total link capacities by deducting the occupied capacity of the corresponding slice:

$$C_{i,j} := C_{i,j} - R_{s,i,j} \quad (7)$$

For this reason, the order in which the Steiner trees are constructed is not irrelevant. To maximize the number of accepted slices, we first sort the slices ascendingly by the number of embedding nodes $|N_s|$, since it is easier to construct a spanning tree for a smaller number of terminals.

A slice is *accepted* if its primary tree is successfully constructed and protected if its backup tree is successfully constructed as well. In the Steiner-balanced variant, we attempt to construct all primary trees for slices in the order described above, and then we attempt to construct all backup trees in the same order, optimizing the number of accepted slices by prioritizing the construction of primary trees. In the Protection-based variant, instead of constructing the backup trees after all primary trees, each backup tree is constructed immediately after the corresponding primary tree, attempting to maximize the number of protected slices.

4.3. PATH CONSTRUCTION: 2-EDGE-CONNECTED SUBGRAPH

The subgraph P_s spanning the nodes (terminals) selected for slice s does not have to be a tree. If P_s already contains multiple paths between each pair of terminals, path protection is ensured without constructing additional backup paths. This is the 2-edge-connectedness property.

To find such a subgraph, the following algorithm is applied for each slice:

1. We initialize P_s to be maximal, consisting of all edges (*i, j*) $\in E$ with enough capacity for slice s ($C_{i,j} \geq R_{s,i,j}$).
2. If this subgraph P_s is not 2-edge-connected, the slice is rejected.
3. Otherwise, we go over all edges in P_s, sorted by latency in the decreasing order.
    1) We check if P_s would still be 2-edge-connected if the current edge was removed.
    2) If the answer is yes, we remove the current edge and thus reduce the total latency of P_s.
4. Return the resulting 2-edge-connected subgraph P_s.

Since edges with higher latency are removed first, the subgraph obtained after removing all unnecessary edges will have minimal latency. It remains to describe how to check if a graph is 2-edge-connected. Verifying the 2-edge-connectedness of Ps is done in the following way:

1. For each pair of terminals u and v in P_s, the function *edge connectivity* from the NetworX library [23] is called to compute the minimum number of edges that must be removed to break all paths from u to v.
2. If the obtained edge connectivity number is greater than 1 for all pairs of terminals u and v, the subgraph P_s is 2-edge-connected.

After constructing each P_s, link capacities are updated by Eq. (7). The order in which the slices are processed is the same as in the previous variants (increasing $|N_s|$). Since each subgraph provides both primary and backup paths for the corresponding slice, the number of accepted and protected slices is equal in this variant. It equals the number of successfully constructed 2-edge-connected subgraphs.

5. EVALUATION

This section describes the simulation results. All experiments were implemented in Python 3 on Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz. We used the network topologies from the SNDlib dataset [8]. We note that most topologies were not suitable for the Steiner-based approaches because of their low redundancy which disabled the construction of link-disjoint trees. The suitable high-redundancy topologies were *pdh, dfn-bwin, newyork*, and *pioro40*, denoted by T1, T2, T3, and T4 in the rest of this section. The link capacity values (in Mbit/s) by individual topologies are: T1: 30, T2: 80000, T3: 1000 and T4: 155.

Apart from using realistic network topologies, we used the link capacities from the same SNDlib files and artificially generated other parameters. We assumed $|S| = 8$ slices, the resource capacity of each node was 100 elements, while the resource requirement per slice was a random integer uniformly chosen from [10, 100]. Revenue per resource unit and embedding costs were random integers uniformly chosen from [10, 100] and [100, 500], respectively. Link latencies were uniformly chosen between 1 and 20 ms. Slices' capacity requirements were uniformly chosen between 1 and the average link capacity divided by 2. Each experiment was repeated 30 times, and the results were averaged.

Fig. 5 compares the algorithm variants and the KC-FS algorithm [16] across several measures. Since the KC-FS performs the second step of the present framework and not the first, the same knaspack-based selection of slice nodes (or SF candidates) is performed as the first step of KC-FS. For its second step, the slices are sorted descendingly by the number of embedding nodes, following the recommendation from Algorithm 4 in [16] which starts from the SF with the most candidates in the embedding network. The main KC-FS algorithm is implemented according to the pseudocode of Algorithm 3 in [16], including Algorithm 1 (*k-Connected Network Slicing Technique*) and Algorithm 2 (*i-Based Node Degree Balancing*) as subroutines. The following paragraphs describe the obtained results by subfigures of Fig. 5.

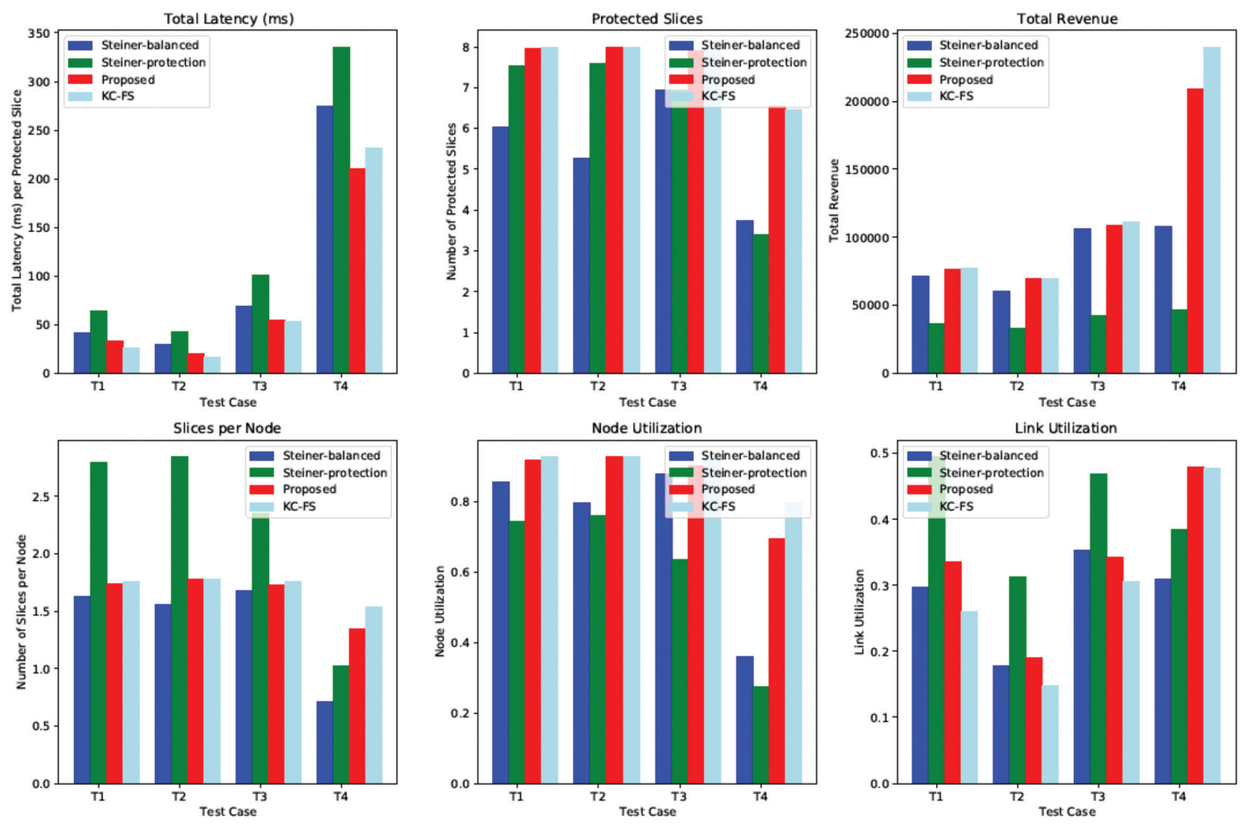

Fig. 5. Comparison between different approaches

The *Total Latency* for each subgraph was calculated by adding up the latencies of its links, and the results were averaged for all protected slices. The Proposed and KC-FS approaches achieve the best results in terms of total latency, with KC-FS giving slightly lower latencies in T1 and T2 and higher latencies in T4. The Steiner-protection variant achieves higher latencies than the Balanced-protection variant because it has the highest number of *slices per node* in all test cases, resulting from its approach to 0-1 Knapsack.

The Steiner-protection algorithm, however, has a higher number of *Protected Slices* in T1 and T2 compared to the Steiner-balanced. The Proposed and KC-FS approaches are the best in this respect, behaving similarly and enabling the highest number of protected slices.

Total revenue (of accepted slices) is expectedly lowest for the Steiner-protection approach, which does not take revenue into account when solving the 0-1 Knapsack. The Proposed approach is more successful than Steiner-balanced, especially in T4, because of the much higher number of protected slices. The KC-FS approach is even more successful in T4 because it starts from the slices with the most nodes.

Node utilization in Fig. 5 was calculated as the average percentage of utilized required resource elements by the slices embedded in each node $i \in N$. Link utilization was calculated as the average percentage of utilized capacities by the slices for which a link $(i,j) \in E$ is chosen for the subgraph/tree. Steiner-protection variant achieves lower node utilization compared to other approaches because of its selection of low-requirement slices when solving the 0-1 Knapsack. The opposite is true for link utilization since the Protection-based approach constructs larger trees because of the higher number of nodes per slice. The Proposed variant achieves slightly higher results than the Steiner-balanced with respect to node and link utilization. The Proposed and KC-FS approaches again behave similarly, with KC-FS utilizing slightly more nodes because of its slice ordering.

The most important difference between Proposed and KC-FS is depicted in Fig. 6 which shows the average execution time per test case. For small networks such as T1, T2, and T3 the execution times are negligible, but for a larger network such as T4 the KC-FS subgraph embeddings take ≈ 3.5× more computation time in comparison to Proposed. This is in line with the complexity of the procedures in Algorithms 1-3 of [16].

The results show the overall dominance of the Proposed approach in terms of execution time, latency, number of protected slices, total revenue, and node utilization. In terms of other measures, the Steiner-protection variant has the highest edge utilization (except in T4), as well as the highest number of slices per node and, equivalently, nodes per slice. However, it is dominated by the Steiner-balanced approach in terms of total revenue. In cases where execution time is irrelevant, the KC-FS approach with knapsack-based node selection can be the best choice in terms of total revenue (T4).

To visualize the effect of network topology, Fig. 7 depicts the resulting subgraphs of specific slices on 12 different low-redundancy topologies. Only the Proposed variant achieved path protection on these topologies.

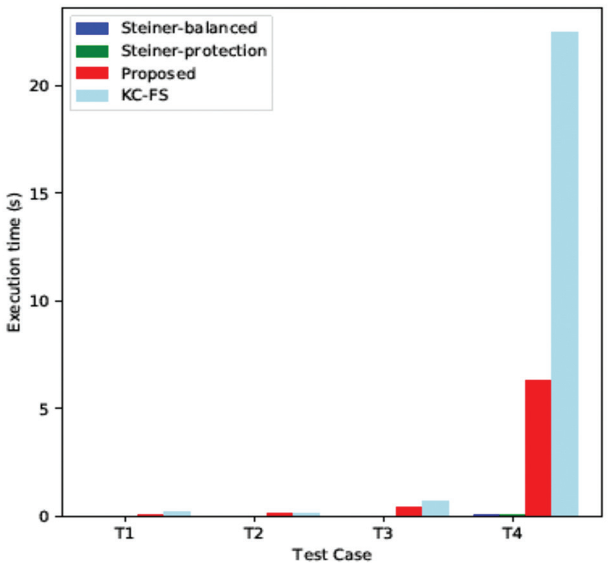

Fig. 6. Average execution time

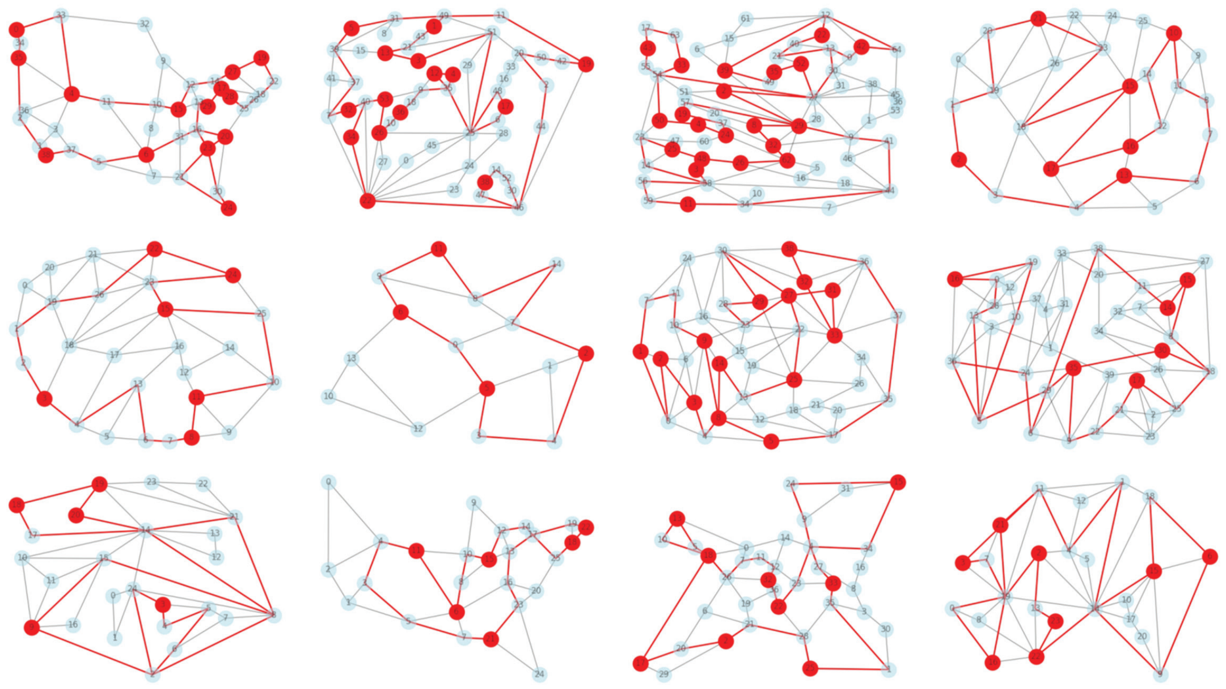

Fig. 7. Slice subgraphs on low-redundancy topologies (row by row: *janos-us-ca*, *zib54*, *ta2*, *norway*, *sun*, *atlanta*, *giul39*, *pioro40*, *france*, *janos-us*, *cost266*, and *ta1* from SNDlib dataset [8]).

6. CONCLUSION

In this paper, we presented a framework with several variants for slicing optimization and path protection in 5G networks, focusing on balancing factors like execution time, revenue, latency, slice protection, and resource utilization.

The proposed approach based on 2-edge-connected subgraphs emerged as the most effective, achieving the lowest total latency, highest number of protected slices, highest node utilization, and highest total revenue along with the existing KC-FS approach which has a significantly higher execution time. This shows that the proposed algorithm created slice subgraphs more efficiently than the recent related approach. The Steiner-protection approach created larger and more protected slices than the Steiner-balanced approach, emphasizing the importance of prioritizing network resiliency with dedicated path protection. However, this comes at the expense of the total revenue, which is dominated by the

Steiner-balanced approach, underscoring the efficacy of dynamic programming in optimizing network resources and showing a tradeoff between revenue maximization and other optimization aspects, such as slice protection and the number of nodes per slice.

7. REFERENCES

[1] D. Irawan, N. R. Syambas, A. A. N. Ananda Kusuma, E. Mulyana, "Network Slicing Algorithms Case Study: Virtual Network Embedding", Proceedings of the 14th International Conference on Telecommunication Systems, Services, and Applications, Bandung, Indonesia, 4-5 November 2020, pp. 1-5.

[2] M. Ait aba, M. Elkael, B. Jouaber, H. Castel-Taleb, A. Araldo, D. Olivier, "A two-stage algorithm for the Virtual Network Embedding problem", Proceedings of the 46th Conference on Local Computer Networks, Edmonton, AB, Canada, 4-7 October 2021, pp. 395-398.

[3] M. K. Singh, S. Vittal, A. A. Franklin, "SERENS: Self Regulating Network Slicing in 5G for Efficient Resource Utilization", Proceedings of the 3rd 5G World Forum, Bangalore, India, 10-12 September 2020, pp. 590-595.

[4] V. Balasubramanian, M. Aloqaily, M. Reisslein, "Mutes: Multi-Tenant Switching for 5G Network Slice Revenue Maximization", Proceedings of the International Wireless Communications and Mobile Computing, Dubrovnik, Croatia, 30 May - 3 June 2022, pp. 590-595.

[5] B. Jaumard, Q. H. Duong, "A Nested Decomposition Model for Reliable NFV 5G Network Slicing", IEEE Transactions on Network and Service Management, Vol. 20, No. 3, 2023, pp. 2186-2200.

[6] C. Raffaelli, E. Amato, P. Monti, F. Tonini, "Reliable Slicing in Optical Metro Networks with Reconfigurable Backup Resources", Proceedings of the International Symposium on Communication Systems, Networks and Digital Signal Processing, Porto, Portugal, 20-22 July 2022, pp. 863-866.

[7] N. Shahriar et al. "Reliable Slicing of 5G Transport Networks with Bandwidth Squeezing and Multi-Path Provisioning", IEEE Transactions on Network and Service Management, Vol. 17, No. 3, 2020, pp. 1418-1431.

[8] S. Orlowski, R. Wessäly, M. Pióro, A. Tomaszewski, "SNDlib 1.0—Survivable Network Design Library", Networks, Vol. 55, No. 3, 2010, pp. 276-286.

[9] M. K. Lee, C. S. Hong, "Efficient Slice Allocation for Novel 5G Services", Proceedings of the 10th International Conference on Ubiquitous and Future Networks, Prague, Czech Republic, 3-6 July 2018, pp. 625-629.

[10] A. Sarah, G. Nencioni, “Resource allocation for cost minimization of a slice broker in a 5G-MEC scenario”, Computer Communications, Vol. 213, No. 3, 2024, pp. 331-344.

[11] Y. Xiao, J. Zhang, P. Zhu, H. Wu, C. Zhang, "Customized Topology-Level Protection for Reliable Slicing in 5G/B5G Metro Access/Aggregation Networks", Journal of Lightwave Technology, Vol. 42, No. 9, 2024, pp. 3068-3080.

[12] D. Basu, S. Kal, U. Ghosh, R. Datta, "DRIVE: Dynamic Resource Introspection and VNF Embedding for 5G Using Machine Learning", IEEE Internet of Things Journal, Vol. 10, No. 21, 2023, pp. 18971-18979.

[13] H. Ko, J. Lee, H. Choi, S. Pack, "Hierarchical Identifier (HID)-based 5G Architecture with Backup Slice", Proceedings of the 21st Asia-Pacific Network Operations and Management Symposium, Daegu, Korea, 22-25 September 2020, pp. 291-293.

[14] N. Shahriar et al. "Reliable Slicing of 5G Transport Networks with Dedicated Protection", Proceedings of the 15th International Conference on Network and Service Management, Halifax, NS, Canada, 21-25 October 2019, pp. 1-9.

[15] I. Begić, A. S. Kurdija, J. Matuško, “A Framework for 5G Network Slicing Optimization with Path Protection", Proceedings of the 47th International Convention on Information and Communication Technology, Electronics and Microelectronics, Opatija, Croatia, 22-26 May 2024.

[16] D. Zheng, G. Shen, Y. Li, X. Cao, B. Mukherjee, "Service Function Chaining and Embedding with Heterogeneous Faults Tolerance in Edge Networks", IEEE Transactions on Network and Service Management, Vol. 20, No. 3, 2023, pp. 2157-2171.

[17] Y. Zheng, S. Ravi, E. Kline, L. Thurlow, S. Koenig, T. K. S. Kumar, "Improved Conflict-Based Search for

the Virtual Network Embedding Problem", Proceedings of the 32nd International Conference on Computer Communications and Networks, Honolulu, HI, USA, 24-27 July 2023, pp. 1-10.

[18] L. Yang, J. Jia, H. Lin, J. Cao, "Reliable Dynamic Service Chain Scheduling in 5G Networks", IEEE Transactions on Mobile Computing, Vol. 22, No. 8, 2023, pp. 4898-4911.

[19] R. Mohamed, A. Leivadeas, I. Lambadaris, T. Morris, P. Djukic, "Online and Scalable Virtual Network Functions Chain Placement for Emerging 5G Networks", Proceedings of the IEEE International Mediterranean Conference on Communications and Networking, Athens, Greece, 5-8 September 2022, pp. 255-260.

[20] H. Liu, "Research on Virtual Network Embedding Model and Algorithm Based on Graph Attention Network and Multi-Layer Perceptron", Proceedings of the 5th International Conference on Artificial Intelligence and Computer Applications, Dalian, China, 2023, pp. 812-818.

[21] A. Samar, K. M. Sivalingam, "RL-based Virtual Network Embedding using VNF Sharing for Network Slicing in 5G Networks", Proceedings of the IEEE/IFIP Network Operations and Management Symposium, Miami, FL, USA, 8-12 May 2023, pp. 1-7.

[22] L. Kou, G. Markowsky, L. Berman, “A Fast Algorithm for Steiner Trees”, Acta Informatica, Vol. 15, No. 2, 1981, pp. 141-145.

[23] A. A. Hagberg, D. A. Schult, P. J. Swart, “Exploring network structure, dynamics, and function using NetworkX”, Proceedings of the 7th Python in Science Conference, Pasadena, CA, USA, 19-24 August 2008, pp. 11-15.