End-to-End Mobile Network Slice Embedding Leveraging Edge Computing

Andrea Fendt
Nokia Bell Labs
Munich, Germany
andrea.fendt@nokia-bell-labs.com

Christian Mannweiler
Nokia Bell Labs
Munich, Germany
christian.mannweiler@nokia-bell-labs.com

Katja Ludwig
Department of Computer Science
University of Augsburg
Augsburg, Germany
katja.ludwig@student.uni-augsburg.de

Lars Christoph Schmelz
Nokia Bell Labs
Munich, Germany
christoph.schmelz@nokia-bell-labs.com

Bernhard Bauer
Department of Computer Science
University of Augsburg
Augsburg, Germany
bauer@informatik.uni-augsburg.de

***Abstract*—Network virtualization and network slicing are key-features of the fifth generation (5G) of mobile networks to overcome the challenge of increasingly diverging network requirements emerging from new use-cases like IoT, autonomous driving and Industry 4.0. In particular, low latency network slices require deployment of their services and applications, including Network Functions (NFs), close to the user, i.e., at the edge of the mobile network. Since the users of those services might be widely distributed and mobile, multiple instances of the same application are required to be available on numerous distributed edge clouds. This paper tackles the problem of Network Slice Embedding (NSE) with edge computing. Based on an Integer Linear Program (ILP) formulation of the NSE problem, the optimal set of network slices, in terms of network slice revenue and cost, is determined. The required network slice applications, functions and services are allocated nearly optimally on the 5G end-to-end mobile network infrastructure. The presented solution also provides the optimal number of application instances and their optimal deployment locations on the edge clouds, even for multiple User Equipment (UE) connectivity scenarios. Evaluation shows that the proposed holistic approach for NSE and multiple edge cloud allocation is feasible and efficient for small and medium sized problem instances.**

***Index Terms*—5G, Network Slice, Virtual Network Embedding, End-to-End, Latency, Edge Computing, Resource Allocation, Low Latency, Integer Linear Programming**

I. Introduction

Various 5G use-cases are based on high performance, reliable mobile data connections with a low end-to-end latency. The 5G standardization comities ETSI NFV and 3GPP agree on network slicing as a key feature to overcome these challenges. Network slices are logical end-to-end networks sharing a common mobile network infrastructure. Network slicing is a mean to comply with privacy requirements and isolate services with different resource requirements on the mobile network. [1] [2]

In addition to that, edge computing is seen as a key feature of 5G to enable local communication, with a very low latency and high throughput, while at the same time improving data security and reducing transmission cost. It provides computation power and data storage close to the mobile users, within the 5G Radio Access Network (RAN), at the so-called edge of the mobile network. Edge clouds are typically deployed on 5G gNodeBs, that means macro base stations or the multi-Radio Access Technology cell aggregation sites, instead of centralized core nodes. This leverages location based services, like the communication between co-located devices in car-to-x and smart factory applications as well as, for instance, augmented reality services. By using edge computing the latency and volume of the transferred data is reduced. Hence, the available bandwidth can be used more efficiently. Additionally, edge clouds can be used for computation offloading, enabling services that cannot be performed by most mobile device, e.g., Artificial Intelligence (AI) based services. Furthermore, a better Quality of Experience (QoE) and a lower battery consumption of the User Equipments (UEs) can be provided by edge computing. However, edge clouds come at high Operational Expenditures (OPEX). Therefore, edge cloud resources are limited, while their applications are constantly increasing. [3]

A feasible deployment of one or several Network Slice Instances (NSIs) on a common physical network infrastructure, is referred to as the Network Slice Embedding (NSE) problem in this work. The physical network infrastructure is also called the substrate in the following. The NSE problem is a special case of Virtual Network Embedding (VNE) problem. The VNE problem is a well-researched NP-hard problem, but it does not consider all resources and capabilities required in an end-to-end mixed wired and wireless mobile NSE problem, in particular strong end-to-end latency, availability and reliability requirements as well as numerous resource demands, like throughput, on the RAN, transport and core network communication links as well as computation power and memory requirements on the servers. In addition to that, allocating edge cloud resources is not considered in conventional VNE solutions. Deploying a service close to its users on an edge

cloud often means that the same service has to be provided at different locations, i.e., for one application in an NSI several instances might have to be deployed in the substrate network. Optimizing the NSE in a way, that the most beneficial NSIs and as many NSIs as possible can be deployed and meet their Service Level Agreements (SLAs) while concurrently targeting at OPEX reduction is, to the best of the authors knowledge, currently an unsolved problem in literature. In this paper, a formal, mathematical model of the NSE problem leveraging edge computing is provided. This model can be solved nearly optimally with an out-of-the-box Integer Linear Program (ILP) solver.

This paper is structured as follows: In section II an overview over related work and the prior art is given. Section III presents the formal model of the NSE problem. Subsequently, a simple example and an evaluation of the feasibility and runtime of the proposed approach given in IV. The paper is concluded with a summary and outlook on future work in section V.

II. Related Work

The NSE problem is a special case of the well-researched VNE problem. The VNE problem belongs to the class of NP-hard problems, first proved in [4], see also [5]. Nevertheless, there are various algorithms and heuristics for the VNE problem. The challenges of NSE in conjunction with edge computing in 5G and beyond are currently under intense discussion. One of the most recent publication is the paper of Sanguanpuak et al. [6]. The authors are modeling the scenarios of network slice allocation to multiple Mobile Network Operators (MNOs). The allocation problem is solved using the generic Markov Chain Monte Carlo Method. It is targeting at minimizing infrastructure cost and considers latency constraints. With the greedy fractional knapsack algorithm, the optimal mapping solution is calculated. The presented algorithm is focusing on determining the best network slice to MNO resource mapping for co-located resources. In contrast, Xian et al. present a Mixed Integer Nonlinear Problem formulation of resource allocation with edge computing in [7]. The proposed optimization algorithm takes the link and server capacities as well as latency restrictions into account. Sequential Fixing (SF) is used to efficiently compute near-optimal solutions for the mobile edge resource allocation problem. In contrast to the publications [6] and [7], this paper provides an end-to-end NSE solution with distributed UEs using a common network slice service. The number of required instances of a specific application within a network slice is determined automatically, depending on the topology of the network, the end-to-end latency, throughput as well as CPU and memory requirements of the respective application. The approaches mentioned above focus on choosing the optimal edge cloud for a predefined application instance associated with a specific UE group at a specific location. However, they do not provide an automated optimization of Multiple Application Instantiation (MAI) for geographically distributed UEs in consideration of latency requirements.

The authors of the paper in [8] propose a method of allocating Virtual Network Functions (VNFs) in slicing-enabled 5G networks using an edge computing infrastructure. Therefore, the required number of instances of the VNFs are determined. In comparison to this work, the presented approach is not integrated with solving the NSE problem.

Further approaches focus on the allocation of computation tasks on distributed clouds during runtime. For example, Alicherry et al. [9] present an efficient approximation algorithm for allocating computation tasks on a distributed cloud with the objective of minimizing communication cost and latency. Hao et al. [10] provide online heuristics for the resource allocation problem for geographically diversified cloud servers. Both papers are providing algorithms for solving the NSE problem. They are both taking latency and throughput restrictions as well as the network topology into account. However, they are not tackling the challenges of edge computing in conjunction with the NSE problem. For instance, Akhatar et al. [11] present an ILP-based solution for the virtual function placement and traffic steering in 5G networks under the assumption, that every service instance is deployed exactly once. In contrast to that, the algorithm presented in this paper optimizes the number of application instances and their placement in the physical network. To the best of the authors knowledge none of the approaches in literature is tailored to end-to-end NSE leveraging MAI and edge computing.

However, in the field of VNE and NSE without explicitly dealing with the challenges of edge computing further publications can be found. For instance, Richart et al. [12] provide an overview over mobile NSE solutions and the paper of Vassilaras et al. [13] summarizes the algorithmic challenges of efficient NSE. Fischer et al. [14] published a survey on VNE targeted towards wired communication networks. Also, Riggio et al. [15] and Tsompanidis et al. [16], deal with the challenges of VNE in wireless networks. Despotovic et al. [17] present a scalable near-optimal solution for the VNE problem. However, the applicability to NSE is limited, since the model does not consider latency and is not capable of many-to-one virtual to physical node mappings.

Further publications on specific runtime aspects of NSE can be found in literature, e.g., Zhang et al. [18] are focusing on the runtime optimization of throughput resources in the context of NSE. Jiang et al. [19] present a UE admission control to avoid network overloading at runtime and Wang et al. [20] present a resource price balancing method considering dynamic offer and demand. In contrast, this paper focuses on NSE for end-to-end mixed wired and wireless networks in the preparation phase.

III. Problem Formalization

In this section a formal mathematical specification of the NSE problem with respected to MAI is presented.

A. Definitions

The NSE model uses the following graph theoretical definitions from [21]. An undirected Graph $G = (\mathcal{V}, \mathcal{E})$ is defined by a set of $n \in \mathbb{N}$ vertices $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ and

a set of $m \in \mathbb{N}$ edges $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$. Every edge e_k for $k = 1, \ldots, m$ has two ends, $v_i \in \mathcal{V}$ and $v_j \in \mathcal{V}$ for $i, j = 1, \ldots, n$, which can be denoted as $e_k := \{v_i, v_j\}$ or as $e_k := v_i v_j$. Note that $v_i v_j = v_j v_i$ in undirected graphs.
A path $P = (\mathcal{V}, \mathcal{E})$ in the network is defined as a subset of an undirected Graph G, $P \subseteq G$. Paths consist of a set of successive edges. Their length is defined as the number of contained edges. The set of paths within an undirected Graph G sharing the same end nodes $v_i \in \mathcal{V}$ and $v_j \in \mathcal{V}$ is defined as $\mathcal{P}_{v_i v_j}$ which is equal to $\mathcal{P}_{v_j v_i}$.
In 5G mobile networks a variety of services are deployed on a shared mobile network infrastructure. This includes the RAN, transport and core network as well as edge, aggregation and central cloud servers. Fig. 1 illustrates a model of the end-to-end mobile network infrastructure with edge computing.
The following model is based on our previous work on NSE

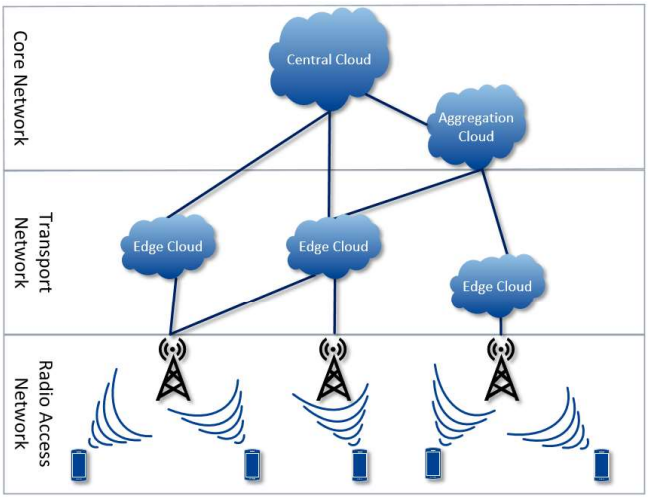

Fig. 1. End-to-End Mobile Network Infrastructure Model

without MAI in [22]. Also, selected principles from [17] are reused. The mobile network infrastructure, also called substrate, is defined as a network graph $\mathcal{N} = (\mathcal{U}, \mathcal{C}, \mathcal{E})$, i.e., as an undirected Graph $G = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} := \mathcal{U} \cup \mathcal{C}$. The vertices of the network graph are of two different types, UE groups $u_v \in \mathcal{U}$ and cloud nodes $c_w \in \mathcal{C}$ (which include edge clouds as well as central and aggregation clouds). Due to the end-to-end mobile network topology, communication links $e_j \in \mathcal{E}$ in the network can be defined between a UE group and a cloud node or between two cloud nodes $\mathcal{E} \subseteq \{u_i c_j, c_v c_w\}$ with $v \neq w$. The same applies to communication paths $\mathcal{P}$ in $\mathcal{N}$, $P_r \in \mathcal{P}$ can connect a UE group with a cloud node or two cloud nodes, but not two UE groups.
$n \in \mathbb{N}$ NSI requests shall be embedded into the mobile network infrastructure. They are modeled as virtual network graphs $\mathcal{N}_k = (\mathcal{U}_k, \mathcal{A}_k, \mathcal{L}_k)$ for $k = 1, .., n$ with $\mathcal{U}_k \subseteq \mathcal{U}$, that means the UE groups are already embedded in the physical network by definition. To compare the importance of the NSIs relative to each other, for instance, the utility of the NSIs or their revenue for the infrastructure provider, weights ω_k are assigned to them. $\mathcal{A}_k$ is the set of application nodes of the k-th NSI, with $a_m^k \in \mathcal{A}_k$ defined as the m-th application node of $\mathcal{N}_k$. To guarantee NSI isolation, NSIs never share application instances. Even if they require the same type of application, distinct instances of the same application are defined for the NSIs. The virtual communication links connecting UE groups and application nodes, are defined as $l_i^k \in \mathcal{L}_k$ for the k-th NSI. Also, in the NSI no direct UE group to UE group connections are considered. A communication connection, like a phone call, between two UE groups is modeled as two connections, one for each UE group towards a common application. Without loss of generality, every l_i^k can be written as $l_i^k = \{u_v, a_m^k\}$ if it is a UE group to application connection or as $l_i^k = \{a_q^k, a_m^k\}$ if it is an application to application connection for distinct applications with $q \neq m$.
Embedding an NSI into the mobile network infrastructure means to map each application node a_m^k on at least one cloud nodes c_w such that each virtual link l_i^k can be mapped on a suitable path in the substrate and all resource requirements and network capabilities are fulfilled. The resources and capabilities considered in this work are restricted to the communication link throughput, latency as well as the node computation power and memory. However, this list can easily be extended by additional resources and parameters. For the mobile substrate network the relevant resources and parameters on the nodes are the computation power D_w^s and the memory capacity M_w^s of the cloud server nodes $c_w \in \mathcal{C}$. The communication links are characterized by their available throughput of the NSI T_j^s, and their maximum latency L_j^s. The available throughput is defined as the maximum possible throughput on the wired or wireless communication link $e_j \in \mathcal{E}$ of the network infrastructure. To keep the model simple, uplink and downlink are not distinguished. For RAN connections an expected Channel Quality Index (CQI) has to be defined in advance in order to determine the maximum throughput based on the frequency bandwidth. For simplicity, the maximum latency is modeled as an upper bound for the data transmission time if the link is on full load, but not overloaded.
In contrast, the NSIs define the throughput T_i^k requirements and the maximum allowed latency L_j^s for each link $l_i^k \in \mathcal{L}_k$. Virtual nodes require specified computation and memory capacities, denoted as D_m^k and M_m^k for the k-th NSI.
Virtual links and nodes can only be mapped on network paths and substrate nodes which fulfill their requirements.

B. NSE Model

We define the following embedding variables:

$$y_k := \begin{cases} 1 & \text{if } \mathcal{N}_k \text{ is embedded into } \mathcal{N} \\ 0 & \text{otherwise} \end{cases}$$

$$a2c^k_{mw} := \begin{cases} 1 & \text{if } a^k_m \text{ is mapped on } c_w \\ 0 & \text{otherwise} \end{cases}$$

$$l2p^k_{ir} := \begin{cases} 1 & \text{if } l^k_i \text{ is mapped on } P_r \\ 0 & \text{otherwise} \end{cases}$$

For better readability an additional path to edge mapping, which is constant for the mobile network substrate, is defined:

$$p2e_{rj} := \begin{cases} 1 & \text{if } e_j \text{ is used in } P_r \\ 0 & \text{otherwise} \end{cases}$$

Together with the $l2p$ mapping variables, the $l2e$ mapping can be derived without creating any additional variables:

$$l2e^k_{ij} := \sum_r (l2p^k_{ir} \cdot p2e_{rj})$$

To maximize the revenue minus the cost of embedding as many beneficial NSIs as possible, the following revenue and cost objective function is used in this paper:

$$\begin{aligned} \max \rho_1 \cdot \frac{\sum_k \omega_k \cdot y_k}{\sum_k \omega_k} - \rho_2 \cdot \sum_{k,m,w} \frac{a2c^k_{mw} \cdot D^k_m}{\sum_w D^k_w} - \\ \rho_3 \cdot \sum_{k,m,w} \frac{a2c^k_{mw} \cdot M^k_m}{\sum_w M^s_w} - \rho_4 \cdot \sum_{i,j,k} \frac{l2e^k_{ij} \cdot T^k_i}{\sum_j T^s_j} \end{aligned} \tag{1}$$

It maximized the weighted, normalized revenue of all embedded NSIs, represented by the NSI weights ω_k, minus the weighted, cost represented by the overall utilization percentage of the CPU, the memory and throughput capacities. The weights ρ_1, ρ_2, ρ_3 and ρ_4 can be used if the absolute revenue and costs are unknown and have to be estimated. The objective function is maximized under the following capability, resource, mapping and graph constraints.

Capability constraints:
The sum of the latencies of all links along the mapped paths must not exceed the allowed virtual link latency:

$$\sum_j l2e^k_{ij} \cdot L^s_j \leq l2p^k_{ir} \cdot L^k_i, \; \forall k, i, P_r \in \mathcal{P} \tag{2}$$

Resource constraints:
The sum of required throughputs T^k_i of all links l^k_i mapped to an edge e_j must not exceed the available throughput on this edge T^s_j:

$$\sum_{k,i} l2e^k_{ij} \cdot T^k_i \leq T^s_j, \; \forall j \tag{3}$$

Also, the sum of the required CPU resources D^k_m of all applications a^k_m allocated on a cloud node c_w must not exceed the available CPU D^s_w on c_w:

$$\sum_{k,m} a2c^k_{mw} \cdot D^k_m \leq D^s_w, \; \forall w \tag{4}$$

The same applies to the required memory resources M^k_m of all applications a^k_m deployed on the cloud node c_w:

$$\sum_{k.m} a2c^k_{mw} \cdot M^k_m \leq M^s_w, \; \forall w \tag{5}$$

Graph constraints:
Every application has to be instantiated at least once, if the NSI is embedded into the substrate network:

$$\sum_w a2c^k_{mw} \geq y_k, \; \forall k, m \tag{6}$$

All communication links connecting two applications must be mapped suitably:

$$\begin{aligned} \sum_{P_r \in \mathcal{P}_{c_v c_w} = \mathcal{P}_{c_w c_v}} l2p^k_{ir} \geq a2c^k_{mw} \\ \forall k, i, w \text{ if } l^k_i = \{a^k_m, a^k_b\} \text{ or } l^k_i = \{a^k_b, a^k_m\} \end{aligned} \tag{7}$$

If an NSI application a^k_m is mapped on a cloud node c_w in the substrate network, then every link connected to a^k_m in the NSI specification must be mapped on at least one path P_r with $P_r \in \mathcal{P}_{c_v c_w} = \mathcal{P}_{c_w c_v}$. That means a path which has c_w as one of its end-nodes.
The same is achieved for the UE group to application links with:

$$\begin{aligned} \sum_{P_r \in \mathcal{P}_{u_v c_w} = \mathcal{P}_{c_w u_v}} l2p^k_{ir} \geq y_k \\ \forall k, i \text{ if } l^k_i = \{u_v, a^k_b\} \end{aligned} \tag{8}$$

Also, all nodes adjacent to a link must be mapped suitably:

$$\begin{aligned} a2c^k_{mv} + a2c^k_{mw} \geq l2p^k_{ir} \\ \forall k, i, r \text{ with } l^k_i = \{a^k_m, a^k_q\} \\ \text{and } P_r \in \mathcal{P}_{c_v c_w} = \mathcal{P}_{c_w c_v} \end{aligned} \tag{9}$$

$$\begin{aligned} a2c^k_{qv} + a2c^k_{qw} \geq l2p^k_{ir} \\ \forall k, i, r \text{ with } l^k_i = \{a^k_m, a^k_q\} \\ \text{and } P_r \in \mathcal{P}_{c_v c_w} = \mathcal{P}_{c_w c_v} \end{aligned} \tag{10}$$

The set of inequations defined in 9 and 10 concern the link mapping, if $l^k_i = \{a^k_m, a^k_q\}$ and l^k_i is mapped on $P_r \in \mathcal{P}_{c_v c_w} = \mathcal{P}_{c_w c_v}$ with the cloud nodes c_v and c_w as its end-nodes. Then a^k_m must be mapped on c_w or c_v and a^k_q must be mapped on c_v or c_w or vice versa. Note that, the inequations 8 and 9 do not exclude that both ends of the virtual link a^k_m and a^k_q are mapped on the same physical end-node of P_r. That means, both could be mapped on c_w or both on c_v. However, this is prevented by the inequations 7.

$$\begin{aligned} a2c^k_{mw} \geq l2p^k_{ir} \\ \forall k, i, r \text{ with } l^k_i = \{u_v, a^k_m\} \text{ and } P_r \in \mathcal{P}_{c_v c_w} = \mathcal{P}_{c_w c_v} \end{aligned} \tag{11}$$

If $l^k_i = \{u_v, a^k_m\}$ and l^k_i is mapped on $P_r \in \mathcal{P}_{c_v c_w} = \mathcal{P}_{c_w c_v}$ with the adjacent nodes u_v and c_w, then a^k_m must be mapped on c_w.

IV. Example and Evaluation

In this section the proposed model formalization is illustrated with a simple example. Subsequently the evaluation setup and results are described.

A. Example

Fig. 2 shows the network graph of a simple substrate network, consisting of two UEs with single RAN connections to the two edge clouds c_0 and c_1. In this example the edge clouds provide a computation power and a memory of in each case 10 units. The edge clouds are linked to the central cloud c_2, with a computational and memory capacity of in each case 1000 units. In this simple example, only one NSI, see Fig. 3, shall be embedded. It consists of the two UEs connected to an application chain with two elements. The applications a_0^0 and a_1^0 both have a memory and computational capacity consumption of 10 units and the allowed latency on all virtual links is 1.5 units with a throughput of 100 units. Compared to that, the actual latency on the all edges of the substrate is 1 unit and their provided throughputs are 100 units. The optimal

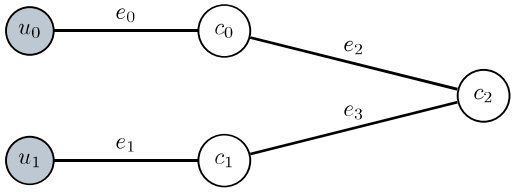

Fig. 2. Simple Example: Substrate Network

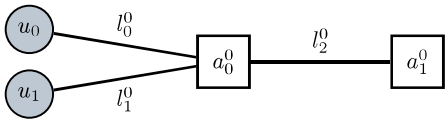

Fig. 3. Simple Example: Network Slice 0

embedding of slice 0 in the substrate network is displayed in Fig. 4. Due to the requirement of low latency of only 1.5 on the links l_0^0 and l_0^1, the application a_0^0 has to be deployed on the edge clouds. Since both UEs use the service a_0^0, it has to be made available on both edge clouds. Therefore, the application a_0^0 is instantiated twice, on the two edge clouds c_0 and c_1. Both instances of a_0^0 are connected to the second application in the function chain a_1^0, which is allocated on the central cloud c_2.

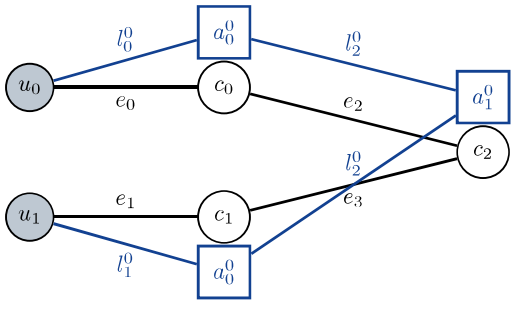

Fig. 4. Simple Example: Network Slice Embedding

B. Evaluation

In this section, two different characteristics of the proposed nearly optimal NSE algorithm with MAI are evaluated. First of all, the dependency between the allowed virtual link latency as well as the actual latency provided by the mobile network infrastructure and the number of application instances created for the NSIs in average are analyzed. In a second analysis, the runtime and scalability limits of the ILP-based approach are analyzed.

For the evaluation of the number of created application instances depending on the latency characteristics of the NSIs, 10 randomly generated NSE problems, similar to the simple example above, are analyzed. For the generation of random NSE problems, the mobile end-to-end network and cloud infrastructure is modeled as a star network topology, using the structure sketched in Fig. 1. UEs are connected via the RAN with antennas and their base stations, which have fast access to an edge cloud installed close to them. Single connectivity is assumed, i.e., every UE is connected to exactly one edge cloud. In addition to that, there are connections to the core network, where they share aggregation and central cloud servers providing high computational and memory capacities. Obviously, the aggregation and central cloud servers are further away from the UE and therefore the data connections have a higher latency than those to the edge clouds. However, deploying servers close to the RAN is typically expensive and therefore the available low latency edge cloud resources are very limited. In this evaluation, we assume that there are 30 UEs groups located in different cells, each connected with one of the 10 edge clouds. Furthermore, there are 4 aggregation and 1 central cloud. Every edge cloud is connected to exactly one aggregation cloud, while every aggregation cloud is connected to the central cloud. The capacity of the edge cloud in memory and CPU is randomly chosen from a uniform distribution across the interval $[80, 100]$. The aggregation clouds have much higher CPU and memory capacities of between 150 and 200 units, while the central cloud provides a memory and CPU capacity of in each case 2000. The link throughputs increase from the edge towards the core of the network. The throughputs on the air interface are assumed to be constant. A random throughput is chosen from a uniform distribution between 20 and 30 units for the RAN connections. It is assumed that the channel qualities remain constant for the UE groups in the network in average. The throughput capacity can be seen as an average, or expected wireless connection capacity. The transport links between the edge cloud and the aggregation clouds also have a throughput between 20 and 30. The core links provide throughputs between 50 and 100. The latencies for all connections in the RAN, transport and core network are set to 1 unit. The latency is also assumed to be constant, which is feasible as long as the channel is not overloaded.

10 NSIs with equal importance weights shall be embedded in one substrate network. Also, the NSI parameters are randomly chosen from uniform distributions within the given ranges. For simplicity, each NSI contains exactly 1 application or function chain, consisting of 2 applications each. Each slice hosts 5 UEs. Distributing an overall number of 30 UEs in the substrate network uniformly across 10 NSIs with 5 UEs per slice means

that statistically each UE uses services of 1.67 NSIs. The memory and computation power of the applications is between 5 and 10 units. The required throughput interval of a NSI link is set to a value between 1 and 2. For evaluating the number of application instances deployed in the substrate network, the allowed latency on the virtual links in the network slices are increased step by step. In the first step, a latency of only 1 is set. This requires the first application in every application chain to be deployed on the closest edge cloud, while the second application of each chain can either be deployed on the same edge cloud or on the closest aggregation cloud. This is due to the actual latencies of 1 on each link in the substrate network. Setting the allowed latency of the virtual links to 1 results in a high number of application instances of 3.46 per application in average. When increasing the latency to 2 also the closest aggregation clouds can be used for the first application in the chain as well as any cloud node for the second application in the chain. This results in an average number of application instances of 1.92 in this evaluation setup. Finally, if the latency of all virtual links is set to 3, latency is no longer an actual restriction for selecting a suitable cloud node for the deployment. However, the link and node resource availability and cost still lead to an average of 1.15 application instance in this evaluation. Although the results inevitably depend on the actual network topology, they show that low latency requirements lead to an increasing number of application instances to be deployed in the network.

The runtime and scalability of the ILP-based nearly optimal NSE algorithm introduced in this paper is implemented and evaluated with a dedicated java program, running on a Mac Book Pro 2015 with a 3,1 GHz Intel Core i7 and a 16 GB 1867 MHz DDR3. For solving the ILP the SCIP [23] is used. For this evaluation NSE scenarios with randomly generated substrate networks and NSIs as explained above are used. In the NSIs a constant latency of 2 on every virtual link in each NSI is set. While the size of the substrate network, modeling a given mobile network infrastructure or subnet is assumed to be fixed, the number of NSIs is increased from 10 in the evaluation above to 50, 100 and 150. For each problem size, 5 different, randomly generated problem instances are evaluated. The average preparation and embedding runtimes

TABLE I
RUNTIME AND NSI ACCEPTANCE EVALUATION

No. Slices	Avg. Prep. Time	Avg. Emb. Time	Avg. No. Constr.
10	16.2 s	4.6 s	49,092
50	645.2 s	45.9 s	245,280
100	2,540.1 s	102.0 s	490,595
150	10,350.7 s	173.5 s	738,368

are summarized in Tab. I. The preparation time includes the transformation of an object-orientated model of the NSE problem into a solver-readable ILP problem with the variables, the objective function and the constraints. The main share of the preparation time (e.g., more than 99% of the preparation time of the scenario with 150 network slices) is consumed by creating the complete constraint matrix, with many zero entries, and feeding it into the solver line by line. This is necessary for the current setup, because for large problem instance, e.g., with hundreds of thousands of constraints, the constraint matrix would quickly overload the memory of ordinary computers. The embedding time refers to the time the solver needs for solving the embedding problem nearly optimally, when the constraint matrix and the objective function is given in the required format. It increases over-linearly for an increasing number of network slices and a constant substrate size. In the evaluated scenarios with a constant substrate size, the number of constraints approximately grows linearly in the number of NSI requests. This will not hold, if the number of substrate elements (nodes and links) as well as the number of accumulated NSI elements over all NSI requests are increased simultaneously. A higher increase in the number of constraints will directly affect the preparation time and also lead to a higher increase in the embedding times.

For medium sized problem instances, as evaluated above, the embedding runtimes are fairly quick considering that a nearly optimal solution for the NP-hard NSE problem with MAI is provided within only a few minutes.

V. CONCLUSION AND OUTLOOK

This work studies resource allocation in a sliced 5G mobile network. It models the NSE problem with MAI taking edge computing as well as the relevant resources and network quality constrains computation power, memory, throughput and latency into account and aims at determining the optimal NSE with respect to revenue and cost. A nearly optimal ILP formalization and implementation is provided. The evaluation of medium sized problem instances shows that the proposed model is realistic and can be solved with an out-of-the-box ILP solver within only a few minutes.

In future work, VNE heuristics should be evaluated and adapted to solve the NSE problem with edge computing. Heuristics can help to solve large problem instances within a short runtime. However, the accuracy of the heuristic solutions should be compared to the nearly optimal solution presented in this paper. Additionally, the approach can be enhanced to address a dynamic NSE problem considering the life-time of the NSIs. This means to solve the NSE problem, as described in this paper, for every time frame, i.e., after each change in the NSI requests.

References

[1] *Next Generation Protocols (NGP), E2E Network Slicing Reference Framework and Information Model*, ETSI GR NGP 011 V1.1.1, ETSI, Sep. 2018.

[2] *Study on management and orchestration of network slicing for next generation network*, TR 28.801 V15.1.0, 3GPP, 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, Jan. 2018.

[3] S. K. et al., "MEC in 5g networks," *ETSI White Paper No. 28*, Jun. 2018.

[4] D. G. Andersen, "Theoretical approaches to node assignment," Dec. 2002, unpublished.

[5] M. Rost and S. Schmid, "NP-completeness and inapproximability of the virtual network embedding problem and its variants," *CoRR*, vol. abs/1801.03162, 2018. [Online]. Available: http://arxiv.org/abs/1801.03162.

[6] T. Sanguanpuak, N. Rajatheva, D. Niyato, and M. Latva-aho, "Network slicing with mobile edge computing for micro-operator networks in beyond 5g," in *2018 21st International Symposium on Wireless Personal Multimedia Communications (WPMC)*, Chiang Rai, Thailand: IEEE, Nov. 2018, pp. 352–357.

[7] B. Xiang, J. Elias, F. Martignon, and E. Di Nitto, "Joint network slicing and mobile edge computing in 5g networks," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, Shanghai, China: IEEE, May 2019, pp. 1–7.

[8] S. Song and J. Chung, "Sliced NFV service chaining in mobile edge clouds," in *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Sep. 2017, pp. 292–294.

[9] M. Alicherry and T. Lakshman, "Network aware resource allocation in distributed clouds," in *2012 Proceedings IEEE INFOCOM*, Mar. 2012, pp. 963–971.

[10] F. Hao, M. Kodialam, T. V. Lakshman, S. Mukherjee, and B. Laboratories, "Online allocation of virtual machines in a distributed cloud," p. 9, Feb. 2017.

[11] N. Akhtar, I. Matta, A. Raza, L. Goratti, T. Braun, and F. Esposito, "Virtual function placement and traffic steering over 5g multi-technology networks," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, Jun. 2018, pp. 114–122.

[12] M. Richart, J. Baliosian, J. Serrat, and J.-L. Gorricho, "Resource slicing in virtual wireless networks: A survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 462–476, Sep. 2016.

[13] S. Vassilaras, L. Gkatzikis, N. Liakopoulos, I. N. Stiakogiannakis, M. Qi, L. Shi, L. Liu, M. Debbah, and G. S. Paschos, "The algorithmic aspects of network slicing," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 112–119, Aug. 2017.

[14] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[15] R. Riggio, A. Bradai, D. Harutyunyan, T. Rasheed, and T. Ahmed, "Scheduling wireless virtual networks functions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, Jun. 2016.

[16] I. Tsompanidis, A. H. Zahran, and C. J. Sreenan, "A utility-based resource and network assignment framework for heterogeneous mobile networks," *2015 IEEE Global Communications Conference*, p. 6, 2015.

[17] Z. Despotovic, A. Hecker, A. N. Malik, R. Guerzoni, I. Vaishnavi, R. Trivisonno, and S. A. Beker, "VNetMapper: A fast and scalable approach to virtual networks embedding," in *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, China: IEEE, Aug. 2014, pp. 1–6.

[18] H. Zhang, N. Liu, X. Chu, K. Long, A. Aghvami, and V. C. M. Leung, "Network slicing based 5g and future mobile networks: Mobility, resource management, and challenges," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 138–145, Aug. 2017.

[19] M. Jiang, M. Condoluci, and T. Mahmoodi, "Network slicing management amp; prioritization in 5g mobile systems," in *European Wireless 2016; 22th European Wireless Conference*, May 2016, pp. 1–6.

[20] G. Wang, G. Feng, W. Tan, S. Qin, R. Wen, and S. Sun, "Resource allocation for network slices in 5g with network resource pricing," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec. 2017, pp. 1–6.

[21] R. Diestel, *Graphentheorie*, 3., neu bearb. und erw. Aufl. Berlin: Springer, 2006.

[22] A. Fendt, C. Mannweiler, L. C. Schmelz, and B. Bauer, "A formal optimization model for 5G mobile network slice resource allocation," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Nov. 2018, pp. 101–106.

[23] A. Gleixner et al., "The SCIP Optimization Suite 6.0," Optimization Online, Technical Report, Jul. 2018. [Online]. Available: http://www.optimization-online.org/DB_HTML/2018/07/6692.html.