An Efficient Online Heuristic for Mobile Network Slice Embedding

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

Andrea Fendt
Department of Computer Science
Universität Augsburg
Augsburg, Germany
andrea.fendt@informatik.uni-augsburg.de

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

***Abstract*—The fifth generation (5G) of mobile networks facilitates the management of various new use cases like autonomous driving, eHealth and the Internet of Things (IoT) along with dense sensor networks. Those use cases induce diverse requirements on the mobile network infrastructure. Network Slicing is one of the key features in 5G to address this issue. Network Slicing organizes the applications in network slices, whereby they are isolated, logical end-to-end networks for different use cases, sharing a common physical network infrastructure. They can be set up, changed and deactivated dynamically and in real-time. Therefore, efficient Network Slice Embedding algorithms are required for the assignment and allocation of network slice resources. In this paper, a novel, efficient approach for such an algorithm is proposed. It is a heuristic which meets the specific requirements of Network Slice Embedding, based on the algorithm provided by Cheng et al. In addition to their topology-aware node ranking algorithm, three alternative ranking techniques for the node mapping are introduced. The Network Slice Embedding heuristic and its ranking algorithms are evaluated on three generic blueprints of network slices and multiple mobile network sizes. The results show that the proposed heuristic is suitable for real-time embedding as it is robust, flexible and scalable and achieves high acceptance ratios.**

***Index Terms*—5G, Network Slicing, end-to-end mobile networks, Virtual Network Embedding, topology**

I. Introduction

Together with the growing demand for high data rates in mobile networks, new use cases introduce a wide range of requirements for mobile networks [1]. For example, safety critical applications like autonomous cars and eHealth require extremely reliable low latency communication. Other use cases, e.g., sensor networks and the Internet of Things (IoT) need a huge amount of concurrent mobile data connections. Services like video streaming, so called enhanced Mobile Broadband (eMBB) use cases, necessitate high availability and high data-rates. In order to deal with these new use cases, Network Slicing is seen as a key enabling technology for the fifth generation (5G) of mobile networks since network slices offer isolated end-to-end connections on a shared network infrastructure [2]. They contain all required resources as defined in the Service Level Agreements (SLA) between network slice customer or tenant and network slice provider [3]. Network Slicing can simultaneously integrate services and applications with diverse requirements and serve a huge number of connected devices.

978-1-7281-5127-4/20/$31.00 ©2020 IEEE

The variety of network slices that emerge through these use cases needs to be embedded in a common physical network. This means to allocate the resources for services and applications and the required mobile data connections to the user equipments (UEs) (e.g., smartphones, notebooks, connected cars, sensors) on the cloud serves of the wired and wireless communication links of the mobile network infrastructure, also referred to as the substrate. As the architecture of the 5G substrate is currently not clear, the flexibility of our approach is shown by considering two variations of possible 5G substrate topologies proposed by [1] and [3]. They are depicted in Fig. 1. Apart from the two substrate topologies,

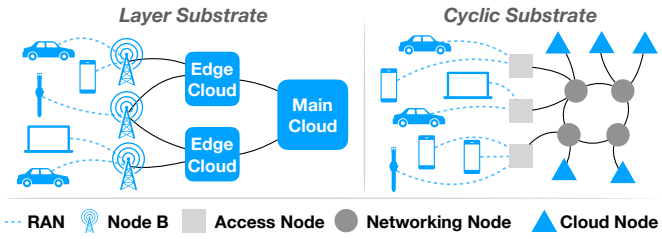

Fig. 1. Layer and cyclic substrate topology

three generic 5G network slice types are considered: ultra Low Latency (uLL), IoT and eMBB slices. uLL slices have extremely strong latency constraints. They might host autonomous driving services or eHealth applications. IoT slices include a large number of UEs, but have relaxed latency, throughput and computing constraints, examples are large sensor networks. eMBB slices are characterized by higher throughput and computing resource demands, together with medium latency constraints. Video streaming services as well as augmented reality applications belong to this class. The process of mapping the virtual resources to the physical resources is called Network Slice Embedding (NSE). Finding the optimal NSE is non-trivial. The problem is strongly related to the Virtual Network Embedding (VNE) problem, which is known to be NP-hard [4]. Removing the UEs from the NSE problem results in a VNE problem, hence the NSE problem is also NP-hard. Therefore, computing the optimal solution is

time-consuming. In contrast to VNE, NSE deals with end-to-end networks including UEs. In our previous work [5], the NSE problem is solved by a nearly-optimal Integer Linear Programming (ILP) approach which is slow for large problem instances and only works in offline scenarios. In this paper, four heuristic algorithms providing an efficient and faster online Network Slice Embedding with high acceptance ratios are implemented.

II. Related Work

In contrast to VNE, which is a well researched problem, there are few publications on the NSE problem. Vassilaras et al. [2] concentrate on the algorithmic challenges that arise with efficient Network Slicing, e.g., efficient allocation, management and control of the slices in real time. Furthermore, they define the optimal, offline optimization problem as well as the online, dynamic version and state that heuristic algorithms are unavoidable for solving these problems in real-life. In previous work, we concentrate on an offline resource allocation and optimization model for Network Slicing, formalized as an ILP [5]. A nearly optimal solution can be calculated with generic solvers. In contrast, this paper focuses on heuristic algorithms that generate good but not necessarily optimal solutions in a fraction of the time that is needed for solving the corresponding ILP. The presented heuristic algorithm is based on rankings of substrate and network slice elements. Furthermore, this paper takes the online version of the NSE problem into account, meaning that the slices are not known in advance and the requests for embeddings arrive over time. Embedded slices can be decommissioned dynamically. The ranking-based heuristics presented in this paper are similar to the VNE heuristics with rankings in [6]–[8]. However, the proposed approaches are not tailored to NSE in end-to-end 5G mobile networks. Especially, latency requirements are not taken into account. Similar to the approach of Cheng et al. [6] for solving the VNE problem, topology-aware ranking methods are considered. In this paper, we evaluate the ranking technique of Cheng et al. [6] which is called NodeRank (NR) as well as three additional ranking algorithms. A simple resource dependent algorithm called ResourceRank (RR), the PageRank (PR) algorithm by Page et al. [9] and a combination of RR and PR, which is called PageResourceRank (PRR) is introduced. To the best of the authors knowledge, RR and PRR are novel and unevaluated ranking techniques for end-to-end NSE.

Furthermore, Liao et al. [8] propose different online, heuristic algorithms as well. Their ranking algorithms are similar to the ones analyzed in this paper. However, They pursue a different approach in analyzing the topology of the networks, based on parameters like degree, closeness (measuring the centrality of the nodes) and the so called betweenness, quantifying how many times a node appears along the shortest path of two other nodes. However, for 5G NSE, resource centered and connectivity-based approaches (like PageRank) seem to be more promising since the network topologies are rather homogeneous.

III. Formal Problem Definition

The model used in this paper is losely related to the models presented in Cheng et al. [6] and Fendt et al. [5]. Substrate networks and network slices are both modeled as instances of undirected graphs $G = (V, E)$. Let $v_i, v_j \in V$ be two vertices of the graph, then the notation $e_{ij} = (v_i, v_j)$ for the edge between these vertices is used. Two subtypes of vertices are distinguished: user vertices V_U and central vertices that are cloud nodes in substrate networks (V_C) and applications (V_A) in network slices. For nodes $c_i \in V_C \cup V_A$, their CPU power is denoted by $P(c_i)$ and their memory capacity by $C(c_i)$. The resources of links $e_{ij} \in E$ are characterized by their throughput $T(e_{ij})$ and their latency $L(e_{ij})$. In substrate networks, these attributes indicate the currently available resources and capabilities of the cloud nodes and links, i.e. embedding a new network slice reduces the remaining resources in the substrate, while deactivating a network slice frees the previously occupied resources. In network slices, the attributes specify the required resources and capabilities. Note that the model could easily be complemented by additional resources or capability if needed. The NSE problem is modeled as follows: A network slice $N = (V_N, V_A, E_N)$ shall be mapped on a substrate network $S = (V_U, V_C, E_S)$, where other slices might already be embedded. All slices are embedded sequentially. If two or more network slice requests arrive at once, they are processed one after another. V_U is the set of all UEs, V_C are the cloud nodes of the substrate and E_S is the sef of the substrate links. For the network slice, $V_N \subset V_U$. The applications V_A and the virtual links E_N should be embedded in the substrate. The embedding is a mapping $M : \mathcal{N} \to \{0, 1\}$ with $M(N) = 1$, if slice N_i is embedded, and $M(N) = 0$, if not. $\mathcal{N}$ is the set of all network slices. In this paper, a is used for application vertices, c for cloud node vertices, u for user vertices and v for vertices in substrate networks as well as network slices.

If $M(N) = 1$, two further mapping functions $M_A : V_A \to V_C$ and $M_E : E_N \to \mathcal{P}$ are defined. $\mathcal{P}$ is a set of paths in the substrate, $p = (e_{ij}, e_{jk}, ..., e_{lm}, e_{mn}), p \in \mathcal{P}, \forall e \in p : e \in E_S$. We write $e \in p$ if the edge e is part of the path p. Since M_A and M_E are functions, each slice element is mapped to exactly one substrate element (a path is considered as one element here). The mapping is subject to the following constraints: A network slice may either be mapped completely or not at all, i.e., if $M(N) = 1$, all slice applications are mapped to cloud nodes in V_C and all virtual links are mapped to substrate paths from $\mathcal{P}$. If the slice is not mapped, $M(N) = 0$, no element of the slice is embedded. Furthermore, the embedding must respect the topology of the slices. If a link connects two nodes in the slice, the mapped path must connect the mapped nodes. With $p = (e_{ij}, ..., e_{mn})$, this can be formalized as

$$\forall e_{kl} \in E_N : M_E(e_{kl}) = p \Rightarrow M_A(a_k) = c_i \wedge M_A(a_l) = c_n \quad (1)$$

When trying to embed a network slice, the following resource and capability constraints need to be satisfied.

$$\forall a \in V_A, X \in \{P, C\} : X(a) \leq X(M_A(a)) \quad (2)$$

$$\forall e \in E_N \; \forall e' \in M_E(e) : T(e) \leq T(e') \tag{3}$$

$$\forall e \in E_N : \sum_{e' \in M_E(e)} L(e') \leq L(e) \tag{4}$$

Similar to the work of Cheng et al. [6], the utility of an embedding is measured with the so called *acceptance ratio*. It is defined as the number of embedded slices divided by the number of all network slice requests. Let $Emb(\mathcal{N}) = \{N \in \mathcal{N} : M(N) = 1\}$, then

$$AccR(M) = \frac{|Emb(\mathcal{N})|}{|\mathcal{N}|} \tag{5}$$

IV. Algorithms

The embedding heuristic is based on the *RW-BFS* method proposed by Cheng et al. [6]. As we deal with end-to-end networks, it is extended to include the UEs. The *RW-BFS* technique requires a ranking algorithm. Apart from the ranking NR that is proposed by Cheng et al. [6], we consider three additional ranking algorithms: RR, PR and PRR. The ranking results are needed multiple times during the embedding process. Rankings that depend only on the topology of the physical network and network slice do not change over time since the network topology is fixed. Therefore, this paper extends the approach of Cheng et al. with storing reusable ranking results for higher runtime efficiency. Additionally, in layer substrates, the Dijkstra algorithm [10] with equal edge weights is used to determine the shortest paths between two nodes. For cyclic substrates we propose a faster approach that takes advantage of the cycle to find the shortest path between two nodes. For two arbitrary nodes v, w, it works as follows. At first, all direct paths P_v from v to Networking Nodes v' and all paths P_w from w to Networking Nodes w' are calculated. If v or w is a UE, the paths include a link to an Access Node and a link from this Access Node to a Networking Node. If v or w is a Cloud Node, the path includes the link to its only neighbor which is a Networking Node. The path is empty, if v or w is a Networking Node itself. Afterwards, both directions of paths through the cycle from v' to w' are computed and included in the set $P_{v'w'}$. The lengths of all possible paths (v to all possible v' via P_v, then to all possible w' via $P_{v'w'}$ and finally via P_w to w) are compared. The shortest one in terms of number of hops is chosen. This concatenated path is the shortest one from node v to node w.

1) ResourceRank (RR): RR ranks the network nodes according to their own resources and to their accessible resources associated with its adjacent communication links. It assigns a memory (f_1), CPU (f_2) and throughput value (f_3) to each node. The memory and CPU values are computed by dividing the respective resources of the considered node by the total amount of required/provided resources in the network slice/substrate. For the throughput value, the sum of the required/provided throughputs of all links connected to the node is divided by the sum of the throughputs of all links, whereby the resources of the links between two non-user nodes are doubled as they have to be considered twice, once for each connected node. All f_i values are multiplied with an importance factor α_k, $k = 1, 2, 3$, $\sum_{k=1}^{3} \alpha_k = 1$. In this paper, node and link resources are considered to be equally important, therefore the importance factors are set to 0.25 for memory and CPU and 0.5 for the link throughput. RR is computed as the weighted sum of all factors:

$$RR(v) = \sum\nolimits_{k=1}^{3} \alpha_k f_k \tag{6}$$

2) PageRank (PR): PR only considers the topology of the network. It is the work of Page et al. [9] and Brin et al. [11] and was initially designed for web pages and links. The hyperlinks in the web are directed, whereas the links of the networks in this paper are undirected. Hence, the number of backlinks used in [9] cannot be used here and is replaced by the number of neighbors: For a vertice $v \in V$, $Neighbor(v) = \{w \in V \,|\, \exists e_{ij} \in E : v_i = v \wedge v_j = w\}$. Then,

$$PR(v) = c \sum_{v' \in Neighbor(v)} \frac{PR(v')}{|Neighbor(v')|} + s \tag{7}$$

The PR of a node is high if it is highly connected to other nodes, including connections over multiple hops. s in Equation 7 is the rank source that is needed for dealing with rank sinks in this recursive calculation. Not including s would lead to unwanted high ranks in cycles [9]. As proposed in [11], s is set to $\frac{1-c}{|U \cup V|}$ and the damping factor c is set to 0.85. A rank has to be assigned to all nodes, including the UEs. These UE ranks are required to receive accurate importance ranks, especially for their direct neighbors.

Evaluations show that PR calculations are slow for network slices. Since the network slices have a quite simple topology, a faster approximation is used for the network slices:

$$PR_N(v) = \frac{|Neighbor(a)|}{|E_i|} \tag{8}$$

This version assigns the highest rank to the node with the highest number of connecting links. Due to the simple structure of the slices, the result of the ranking is similar to the original method. An analysis shows that it is nearly 5 times faster than PR and only 0.07 percentage points behind in terms of acceptance ratio. Therefore, in the evaluation, the original PR method is used for the substrate and combined with the simplified PR_N algorithm for the network slices.

3) PageResourceRank (PRR): PRR combines RR and PR:

$$PRR(v) = 0.5 \cdot PR(v) + 0.5 \cdot RR(v) \tag{9}$$

With the factor 0.5 for both PR and RR, topology and resource characteristics are weighted equally for the final rank. Combining these two rankings seems to be a promising approach, since the topology as well as the resources are crucial aspects for a successful and efficient NSE.

4) NodeRank (NR): NR is an alternative approach for combining resource and topology characteristics in one ranking. NR is proposed by Cheng et al. [6]. Only bandwidth (or throughput in our case) and CPU are taken into account, memory and latency are not considered.

V. Evaluation

For the evaluation of the NSE heuristic, a dedicated Java program is used. All results are collected on a MacBook Air 2013 with a 1.3 GHz Intel Core i5 and a 4 GB 1600 MHz DDR3. The evaluated substrate networks as well as the network slices are generated randomly. That means, the resources of each node and link, the link latencies and the number of links are selected randomly from defined ranges.

The ILP-based approach of [5] is used for a first runtime comparison. It computes a nearly optimal solution. A small layer substrate network is used in this simulation (40 UEs, 20 Nodes B, 5 Edge Clouds, 1 Main Cloud) with small (1–10 UEs and 1–5 applications) slices of all three network slice types. Each run of the evaluation has a new substrate and one slice. The runtime of the algorithms is divided into two parts: the time taken by the ranking and by the embedding. Both together form the total time. The result computed from 120 runs can be found in Fig. 2. As the ILP-based NSE

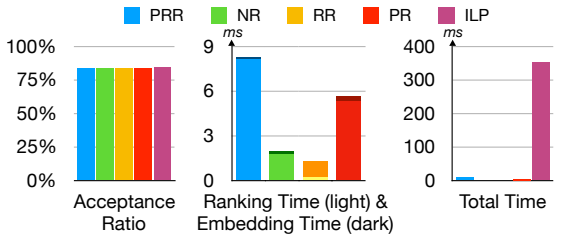

Fig. 2. Runtime and acceptance ratio evaluation with ILP

is a nearly optimal approach, it is capable of embedding in average 84.2 % of the slices, whereas the other algorithms have average acceptance ratios of 83.3 %, but this comes at a high expense. The heuristic algorithms have average ranking times between only a fraction of a millisecond and around 8 ms. Their average embedding times are around 1.1 ms. This evaluation shows that the ILP-based NSE, with an average total runtime of around 350 ms, is far slower than the heuristic approaches.

For a deeper evaluation of the runtime and scalability of the proposed NSE heuristic, compared to the near-optimal ILP-based solution, they are evaluated on differently sized substrates. 20 runs, each including three slices, are executed

TABLE I
Runtime with different substrate sizes

	PRR	**NR**	**RR**	**PR**	**ILP**
S_A	1.733 ms	2.067 ms	1.217 ms	1.667 ms	772.7 ms
S_B	2.983 ms	2.317 ms	0.800 ms	2.233 ms	2127 ms
S_C	3.500 ms	2.867 ms	1.200 ms	3.733 ms	6071 ms
S_D	6.133 ms	4.350 ms	1.450 ms	4.317 ms	12.02 s
S_E	10.87 ms	8.400 ms	2.667 ms	9.267 ms	15.96 s

to determine the average values. The slices are approximately 50% smaller than in the evaluations above in order to also fit into S_A, the smallest tested substrate. S_A contains 20 UEs, 10 Nodes B, 3 Edge Clouds and 1 Main Cloud. The size of the substrates is increased in each step, 20 UEs, 10 Nodes B and 2.5 Edge Clouds (properly rounded) are added. Hence, the substrate sizes increase nearly linearly in the table. S_E, the largest substrate, has around 5 times the size of S_A. The results of this evaluation clearly show that the ILP-based approach is not scalable on large problem instances. On S_E, it is nearly 1500 times slower than the heuristic with the slowest ranking algorithm and nearly 6000 times slower than the heuristic with the fastest ranking algorithm. Therefore, ILP-based NSE is not applicable in real world scenarios.

Next, the differences between the ranking algorithms are analyzed. The network configuration is as follows. The layer substrates have 50 UEs, 30 Nodes B ($P, C \in [100, 200]$), 10 Edge Clouds ($P, C \in [200, 700]$) and 1 Main Cloud ($P, C \in [5000, 10000]$). There are 1–3 links per user to Nodes B with $T \in [30, 80], L \in [3, 7]$, 2–6 links per Node B to Edge Clouds ($T \in [80, 150], L \in [3, 5]$) and every Edge Cloud is linked to the Main Cloud ($T \in [200, 500], L \in [2, 4]$). The cyclic substrates contain 50 UEs as well, 5 Access Nodes ($P, C \in [200, 500]$), 20 Networking Nodes ($P, C \in [50, 200]$) and 25 Cloud Nodes ($P, C \in [500, 5000]$). Users are connected to 1–3 Access Nodes with $T \in [50, 100], L \in [3, 8]$ and Access Nodes are linked to 3–5 Networking Nodes ($T \in [80, 150], L \in [2, 3]$). The links on the network circle have $T \in [300, 500]$ and $L \in [1, 2]$. Each Networking Node may be linked uniquely to 1–4 Cloud Nodes ($T \in [100, 500], L \in [1, 2]$). The evaluation includes three types of slices. The uLL slices contain 1–10 UEs, 1–5 applications with $P, C \in [3, 15]$ and 1–3 links per app with medium throughput and strong latency constraints ($T \in [10, 40], L \in [10, 30]$). The eMBB slices include 1–10 UEs and applications, whereby the applications have medium to high CPU and memory requirements ($P, C \in [10, 40]$). There are 1–3 links per application with medium latency and medium to high throughput constraints ($T \in [10, 40], L \in [25, 50]$). The IoT slices contain more users, 15–30, and 1–5 applications with $P, C \in [1, 3]$. Each application can have 5–20 links with low throughput and high latency constraints ($T \in [1, 5], L \in [50, 100]$). The slices arrive over time and are deactivated after a predefined lifetime. For the evaluation results, 50 runs are executed. Table II shows the average acceptance ratios for each substrate topology (*C-* for the cyclic and *L-* for the layer substrate), each slice type and each algorithm. The average standard deviations are between 0.028 for *L-IoT* and 0.080 for *C-IoT* and PR.

TABLE II
Average Acceptance Ratio in %

	C-IoT	**L-IoT**	**C-uLL**	**L-uLL**	**C-eMBB**	**L-eMBB**
NR	63.83	96.06	69.34	56.62	65.02	61.83
PR	60.09	96.10	73.09	56.59	71.19	61.69
RR	62.58	96.06	70.22	56.68	65.68	61.87
PRR	60.97	96.03	72.90	56.48	70.65	61.65

In general, all algorithms achieve high acceptance ratios within short runtimes, no matter which slice type or substrate topology is selected. However, some small differences

between the ranking algorithms are recognizable. The largest difference is observed with eMBB slices on cyclic substrates: PR and PRR perform notably better than RR and NR, the difference is 6.17 %. The results of the evaluation with IoT slices on the cyclic substrate also show notable differences and are contrary to the ones of the eMBB evaluation. NR and RR perform well with a big gap to PRR and PR is the worst. The difference in the average acceptance ratio over all network slice types between the best (NR) and the worst ranking algorithm (PR) is 3.74 %. This difference might not sound large, but compared to the difference between ILP-based NSE and the heuristic approach of 0.9 % in the first evaluation, the difference here is up to 6 times higher. Regarding the total number of slices of around 2000, 6.17 % mean that NR embedded around 120 more slices. The highest total runtimes in this evaluation are observed with the IoT slices on the cyclic substrate: PRR is the slowest algorithm with 9.0 ms total time per slice. On the layer substrate, the IoT slices have the fastest runtimes. All algorithms need only around 0.05 ms in average per slice. With the same substrates and slices, ILP-based NSE has the fastest results for low latency slices with 2.5 s and the highest total runtimes for IoT slices with 77 s. This shows again that the runtime of the ILP-based approach increases enormously with larger network instances, whereas the times of the heuristic algorithms stay in the same magnitude.

The heuristic algorithms also scale for even larger, randomly generated substrates. Based on the substrate sizes of the previous evaluation (Table II), the substrate size for this evaluation is denoted by $U_x N_y$. This means that the number of UEs is increased x times and the number of all other nodes except from the main cloud is increased y times. For example, $U_{10}N_5$ has 500 UEs and its layer substrates 150 Nodes B and 50 Edge Clouds. For each substrate size, the averages across 10 runs with 12 slices from all three types are taken. Fig. 3 shows that the runtime for NR is always the highest and it grows exponentially for larger numbers of cloud nodes. Therefore, it might not be applicable for large real world scenarios. All other algorithms scale quite well. RR is the fastest algorithm in most cases. On the layer

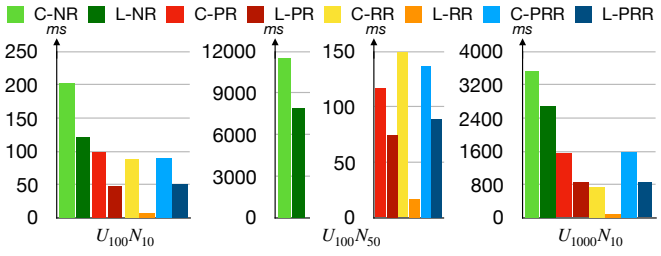

Fig. 3. Runtime evalutation with large substrates

substrate, its average total time is 81.28 ms even for substrate size $U_{1000}N_{10}$. The total times of all other ranking algorithms are more than 10 times higher. Only on cyclic substrates with the size $U_{100}N_{50}$ PR performs better. Hence, for even larger scenarios, RR and PR are the most promising approaches.

VI. Conclusion and Outlook

This paper proposes an online, ranking based heuristic for solving the NSE problem, which is based on the VNE heuristic provided by Cheng et al., but considers the specific end-to-end requirements of NSE. Three new ranking algorithms, a new RR mechanism, a topology-aware PR algorithm and a combination of them, are presented. These ranking algorithms and the NR algorithm provided by Cheng et al. [6] are evaluated. Extensive simulations show that the proposed heuristic is robust to various network slice types, including eMBB , IoT and uLL network slices and different mobile network topologies. All ranking techniques achieve similar acceptance ratios, whereby the best algorithm is dependent on the slice and substrate type. Beyond that, the proposed four variants of the NSE heuristic, especially RR and PR, are scalable for large, real world problem instances. Evaluations show that the NSE heuristic is up to 6000 times faster than the near-optimal ILP NSE solution.

Since none of the proposed ranking algorithms is best for every network slice type and substrate topology, a conditional ranking might be useful. In future work, the best ranking algorithm could be chosen by analyzing the characteristics of the underlying embedding problem. By means of, for instance, Machine Learning (ML), such a classification system cloud be established. Furthermore, future work should support dynamic changes of network slices, e.g. in resource demands, or deployment of new services within a slice. This might result in the necessity to rearrange network slices during runtime. This reconfiguration of the resource assignment should also be used for optimizing the embedding of the network slices in the mobile network infrastructure.

References

[1] NGMN, "NGMN 5G white paper," *NGMN 5G Initiative*, 2015.

[2] S. Vassilaras *et al.*, "The algorithmic aspects of network slicing," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 112–119, 2017.

[3] I. Labrador Pavon *et al.*, "5g norma network architecture – intermediate report," deliverable D3.2, 5G NORMA, 2017.

[4] M. Rost and S. Schmid, "$\mathcal{NP}$-completeness and inapproximability of the virtual network embedding problem and its variants," *arXiv preprint arXiv:1801.03162*, 2018.

[5] A. Fendt, S. Lohmuller, L. C. Schmelz, and B. Bauer, "A network slice resource allocation and optimization model for end-to-end mobile networks," *5G World Forum*, 2018.

[6] X. Cheng *et al.*, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 38–47, 2011.

[7] Z. Zhang, A. X. Liu, X. Cheng, Y. Wang, X. Zhao, and S. Su, "Energy-aware virtual network embedding," *IEEE/ACM Transactions on Networking (TON)*, vol. 22, no. 5, pp. 1607–1620, 2014.

[8] J. Liao, M. Feng, T. Li, J. Wang, and S. Qing, "Topology-aware virtual network embedding using multiple characteristics," *KSII Transactions on Internet & Information Systems*, vol. 8, no. 1, 2014.

[9] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," tech. rep., Stanford InfoLab, 1998.

[10] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[11] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer networks and ISDN systems*, vol. 30, no. 1-7, pp. 107–117, 1998.