LARAC-SN and Mole in the Hole: Enabling Routing through Service Function Chains

Amaury Van Bemten, Jochen W. Guck, Petra Vizarreta, Carmen Mas Machuca, and Wolfgang Kellerer
Lehrstuhl für Kommunikationsnetze
Technical University of Munich
Email: {amaury.van-bemten, guck, cmas, wolfgang.kellerer}@tum.de, petra.vizarreta@lkn.ei.tum.de

***Abstract*—With the emergence of Software-Defined Networking (SDN) and Network Function Virtualization (NFV), the problem of centralized routing through intermediate specified nodes (for which several candidates can be defined) has become an important issue. Indeed, an SDN controller routing flows through Service Function Chains (SFCs) has to efficiently solve this problem to achieve the online provisioning of routing requests and the online placement of Virtual Network Functions (VNFs). In this paper, we propose two algorithms for solving this problem. First, we propose *LARAC for specified nodes* (LARAC-SN), a fast and close to optimal algorithm for finding the constrained shortest path (CSP) visiting an ordered set of specified nodes. Second, we propose *Mole in the Hole* (MITH), a graph transformation algorithm which can force any state-of-the-art routing algorithm to visit an ordered set of specified nodes. While LARAC-SN is bounded to the specific CSP problem and can only handle one candidate per specified node, MITH can be used for any routing problem and can deal with several candidates per specified node. Through evaluations, we show that LARAC-SN is fast and close to optimal (its optimality gap stays lower than 1.62% in average) and that MITH has the potential of reaching optimality for any problem, but at the cost of a higher runtime.**

Keywords*—*routing, intermediate nodes, specified nodes, service function chaining (SFC), function placement problem (FPP), network function virtualization (NFV), LARAC, graph transformation

I. Introduction

As a fundamental graph theory problem, *routing* is a well-known and important problem [1], [2]. Routing problems vary depending on *(i)* the type of route that has to be found (e.g., a path, a tree), and on *(ii)* the properties that this route has to satisfy (e.g., shortest, delay-constrained). For example, finding the least-cost tree from a node to a set of nodes and finding a path between two nodes whose length and cost are lower than given thresholds are two different routing problems. Routing problems already received a lot of attention in the literature [3].

A. Motivation: Routing through Service Function Chains

Routing problems can be extended by requiring the solution to traverse a given set of n so-called *specified nodes* (denoted by $\{N_1, \dots, N_n\}$). We define such an extension as a *specified nodes extension* (SNE) or as an *ordered SNE* (OSNE) if the set of specified nodes has to be visited in a specific order. For a given specified node N_i, a set of up to c so-called *candidate nodes* (denoted by $\{N_i^a, N_i^b, \dots\}$) can be given, one of them at least having to be visited. If only one candidate is defined for a given specified node N_i, we simply refer to it as N_i.

This type of routing problem is now gaining attention in the context of *Network Function Virtualization* (NFV) [4]. In such

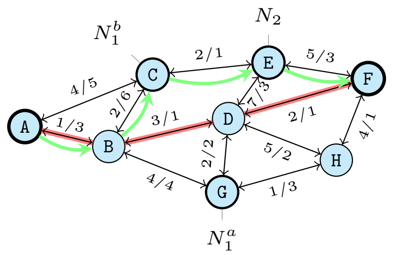

Fig. 1: Illustration of the *ordered specified nodes extension* (OSNE) of the unicast *constrained shortest path* (CSP) problem.

a scenario, routing requests can define a set of *Virtual Network Functions* (VNFs) that have to be traversed (e.g., security functions). This is referred to as *Service Function Chaining* (SFC). Besides, as part of NFV, the *Function Placement Problem* (FPP) [5] consists in finding the optimal locations for hosting the VNFs. The routing of a request through an SFC corresponds to the problem of routing through specified nodes. The FPP corresponds to defining different candidate nodes for each potential VNF location.

An example of this problem is depicted in Fig. 1. Let us consider a *constrained shortest path* (CSP) [6] problem from `A` to `F` that has to visit two ordered specified nodes N_1 and N_2 with a maximum constraint value of 15. N_1 has two candidates `G` and `C` and N_2 only has one candidate `E`. For example, in Fig. 1, if N_1^a and N_1^b correspond to two nodes at which a firewall VNF can be deployed, and N_2 to a node hosting a deep packet inspection (DPI) VNF, the illustrated request corresponds to routing a flow through an SFC consisting of a firewall and a DPI and placing the firewall VNF at one out of two possible locations (i.e., solving the FPP). In Fig. 1, each edge is labeled with its cost and constraint metric values separated by a slash. Without considering the specified nodes, the optimal CSP (highlighted in red) is `A-B-D-F` with a cost of 6. This path can be found by traditional optimal CSP algorithms [7]. Considering the specified nodes, the optimal CSP (shown in green) is `A-B-C-E-F` with a cost of 10. While the definition of the OSNE of the CSP problem is straightforward, traditional CSP algorithms [7] are not able to solve this extension and alternative solutions are hence needed.

B. Contributions: LARAC-SN and MITH, Solutions for Routing through Ordered Service Function Chains

First, in Sec. III, we propose *LARAC for specified nodes* (LARAC-SN), a heuristic for solving the *ordered specified nodes extension* (OSNE) of the unicast *constrained shortest path* (CSP) problem based on the state-of-the-art LARAC heuristic [6], [8]. In our evaluations (Sec. V), its optimality gap remains lower than 1.62% in average. However, LARAC-SN can only handle one candidate per specified node ($c = 1$) and is relevant only for unicast CSP problems.

Second, in Sec. IV, for problems different than the unicast CSP problem and/or for problems considering several candidates per specified node ($c > 1$), we propose *Mole in the Hole* (MITH), a graph transformation algorithm that allows any routing algorithm to be able to solve the OSNE of the routing problem it is originally solving. The power of MITH resides in the fact that it is algorithm-agnostic and can hence be used by any routing algorithm.

For the specific case of the CSP problem with a single candidate per specified node, we show in Sec. V that LARAC-SN presents a much better runtime than algorithms extended with MITH (at least around 10 times faster in our simulations). However, while LARAC-SN is slightly sub-optimal, MITH allows the optimal solution to be found.

II. Related Work

Since the emergence of the SDN and NFV technologies, a wide range of work has been addressing the problem of routing flows through specified nodes. We classify the existing approaches in six overlapping categories for which we list the most representative examples and with respect to which we highlight the contributions of the work in the present paper.

1) Unordered Specified Nodes: The original literature on routing through specified nodes considered the problem of routing through an unordered set of nodes [9]–[11]. That is, the set of specified nodes can be visited in any order. Recent work also studied this problem [12]–[15]. In this paper, we consider that the order in which the nodes have to be visited is fixed. Further, these proposals do not consider candidate nodes for the specified nodes, which our MITH proposal does.

2) Offline Solutions: Several offline algorithms have been proposed [16]–[19]. These algorithms tackle the problem globally, i.e., optimize the solution for a set of routing requests. In this paper, we consider the online problem, i.e., we aim at solving the problem independently for each routing request.

3) Solutions based on Integer Linear Programming: Some proposals model the problem as an *integer linear program* (ILP) and obtain a solution using an ILP solver [5], [14], [16], [17]. In this paper, we avoid formulating the problem as an ILP. Indeed, this often leads to high runtime and requires a new ILP formulation for every different type of routing problem.

4) Special Purpose Algorithms: Several papers propose a special-purpose algorithm for dealing with the SNE of specific problems [9]–[15], [18]–[21]. Our proposed LARAC-SN algorithm falls into this category for the constrained shortest path (CSP) problem. On the contrary, our MITH approach is independent of the routing problem and can be used along with any state-of-the-art routing algorithm.

5) Solutions based on Layering: A couple of solutions duplicate the subject graph into different layers whose interconnections represent the different functions [17], [18], [22]. Our MITH algorithm belongs to this category. Existing proposals based on layering however then run a special-purpose algorithm for the specific problem the authors are dealing with. In contrast, our proposed MITH algorithm runs any state-of-the-art algorithm on the layered graph. As such, any routing algorithm can use MITH to deal with the OSNE of an existing problem, making our proposal more general.

6) Loop-free Solutions: Some proposals require the solution to be loop-free [12]–[16]. As other works [9], [21], we do not define such a constraint and we consider that paths can contain loops.

III. LARAC-SN: OSNE of the CSP Problem

In this section, we consider the specific unicast *constrained shortest path* (CSP) problem and its OSNE with a single candidate per specified node ($c = 1$). We propose *LARAC for specified nodes* (LARAC-SN), a heuristic for this problem. After mathematically defining the *shortest path* (SP) and CSP problems and their respective OSNEs (Sec. III-A), we first consider the OSNE of the SP problem (Sec. III-B). Indeed, our proposed LARAC-SN heuristic is based on this problem. Then, we present the LARAC-SN heuristic in Sec. III-C.

A. Preliminary Definitions

The unicast *constrained shortest path* (CSP) problem (simply referred to as CSP problem in this section) requires two metrics to be defined at each edge. The problem consists in finding the route minimizing the first metric (the *optimized metric* – often referred to as *cost*) while keeping the second metric (the *constraint metric* – often referred to as *delay*) lower than a given bound. Mathematically, consider routing to be performed on a network graph $G = \{V, E\}$, where V is the set of vertices (or nodes) and E is the set of directed edges (with $|E|$ denoting the number of edges in the graph). The vector of costs of the edges is denoted by c, $\mathsf{c} \in \mathbb{R}_+^{|E|}$. Let $d \in \mathbb{R}_+$ denote the bound for the constraint metric. Let d, $\mathsf{d} \in \mathbb{R}_+^{|E|}$, denote the vector of the constraint values for the individual edges. Let P_{sd}, $P_{sd} \subseteq \{0,1\}^{|E|}$, denote the set of paths from the source node s to the destination node d (where a value of 1 for an edge means that the edge belongs to the path). For additive metrics, the CSP problem can then be formulated as

$$z_{\text{opt}} = \min_{\mathsf{x} \in P_{sd}} \mathsf{c}^{\mathsf{T}}\mathsf{x} \tag{1}$$

$$\text{s.t.} \quad \mathsf{d}^{\mathsf{T}}\mathsf{x} \leq d. \tag{2}$$

The unicast *shortest path* (SP) problem (simply referred to as SP problem in this section), for its part, consists in finding the route minimizing a single metric (Eqn. 1 only).

The formulation of the OSNE of the CSP problem with one candidate per specified node is then straightforward. The solution has further to belong to the set of paths that traverse nodes $N_1, \ldots, N_n$. Let $P_{sd}^{N_1,\ldots,N_n}$ denote the intersection of this set of paths and P_{sd}. The OSNE of the CSP problem with one candidate per specified node can then be formulated as

$$z_{\text{opt}} = \min_{\mathsf{x} \in P_{sd}^{N_1,\ldots,N_n}} \mathsf{c}^{\mathsf{T}}\mathsf{x} \tag{3}$$

$$\text{s.t.} \quad \mathsf{d}^{\mathsf{T}}\mathsf{x} \leq d. \tag{4}$$

The OSNE extension of the SP problem with one candidate per specified node then corresponds to Eqn. 3 only.

B. Solving the OSNE of the SP Problem

The adaptation of any SP algorithm to the OSNE of the SP problem with one candidate per specified node is straightforward. The solution corresponds to the concatenation

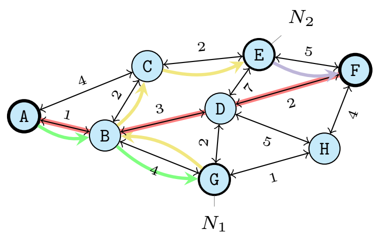

Fig. 2: Illustration of the OSNE of the SP problem with one candidate per specified node. The solution of the SP problem is shown in red (A−B−D−F). The solution of the OSNE of the SP problem corresponds to the concatenation of the SPs (green (A−B−G), yellow (G−B−C−E), purple (E−F)) between the successive pairs of specified nodes.

of the SPs between the successive pairs of specified nodes (Fig. 2). We refer to such an algorithm as a *shortest-path specified nodes* (SP-SN) algorithm, which can be implemented using any SP algorithm (e.g., the *Dijkstra algorithm* [23]).

C. Description of LARAC-SN

The CSP problem has been thoroughly investigated in the literature and a wide range of algorithms have been proposed [3], [7]. However, unlike for the SP problem, the adaptation of CSP algorithms for visiting a set of ordered specified nodes is not straightforward. Indeed, because of the constraint metric, the search cannot be split as for the SP problem, as this would require to determine the distribution of the constraint among the different segments, which is not straightforward.

1) The State-of-the-Art LARAC Algorithm: The CSP problem being NP-complete [1], a wide range of heuristics have been proposed [7]. Among these, the LARAC heuristic [6], [8] generally achieves the best trade-off between runtime and optimality gap [7]. The LARAC algorithm is based on the *Lagrange relaxation technique*, which allows to solve a constrained problem by removing some of the constraints and by introducing them in the optimization objective [24]. For example, the Lagrange relaxation of problem (1)–(2) is

$$L(\lambda) = \min_{\mathsf{x} \in P_{sd}} \quad \mathsf{c}^\mathsf{T}\mathsf{x} + \lambda(\mathsf{d}^\mathsf{T}\mathsf{x} - d). \tag{5}$$

It can be shown that, if the original problem is feasible, then there is an optimal solution to

$$z_L = \max_{\lambda \in \mathbb{R}_+} L(\lambda), \tag{6}$$

that is a feasible solution of the original problem. The idea of the Lagrange relaxation technique is then to obtain a solution to problem (1)–(2) by solving problem (6), which is potentially easier. Solving problem (6) requires to solve the relaxed problem (5) several times in order to find the λ maximizing $L(\lambda)$. For the CSP problem, the relaxed problem corresponds to an SP problem with a modified cost function $c'_i(\lambda) = c_i + \lambda d_i$. This means that, for solving the CSP problem, the LARAC algorithm subsequently runs several SP searches optimizing the combined metric $c'_i(\lambda)$ with different λ values. The computation of the λ values and the terminating condition can be found in the original references of the LARAC algorithm [6], [8]. The algorithm always finds a solution if one exists (i.e., it is *complete*) but is, by the properties of the Lagrange relaxation technique [24], not optimal.

2) Adaptation of the LARAC Algorithm: We propose to also use the Lagrange relaxation technique for the OSNE of the CSP problem with one candidate per specified node. The Lagrange relaxation of problem (3)–(4) is

$$L(\lambda) = \min_{\mathsf{x} \in P_{sd}^{N_1,\dots,N_n}} \quad \mathsf{c}^\mathsf{T}\mathsf{x} + \lambda(\mathsf{d}^\mathsf{T}\mathsf{x} - d), \tag{7}$$

which corresponds to the OSNE of the SP problem with the combined metric $c'(\lambda)$. As such, the LARAC algorithm can simply be adapted by running an SP-SN (see III-B) algorithm at each iteration instead of a simple SP algorithm. The computation of the λ values and the terminating condition are identical to the original LARAC algorithm. We refer to this algorithm as *LARAC for specified nodes* (LARAC-SN). By the properties of the original LARAC algorithm, LARAC-SN is complete but not optimal. By including the constraint metric in the combined optimization metric $c'(\lambda)$ of the SP-SN runs, the LARAC-SN algorithm automatically distributes the usage of the constraint metric budget along the different segments between the different specified nodes.

IV. Mole in the Hole (MITH)

While LARAC-SN is an interesting solution for the OSNE of the CSP problem with one candidate per specified node, it presents several drawbacks: *(i)* it is not optimal (though close to optimal – see Sec. V-B), *(ii)* it cannot deal with several candidates per specified node and *(iii)* it is tailored to the unicast CSP problem. In this section, we present *Mole in the Hole* (MITH), our solution to overcome the limitations of LARAC-SN. First, MITH allows to obtain an optimal solution for the OSNE of any routing problem, overcoming limitations *(i)* and *(iii)* of LARAC-SN. Second, MITH allows to deal with any number of candidate nodes per specified node, thereby overcoming limitation *(ii)* of LARAC-SN.

MITH is a graph transformation algorithm based on layering. The introduced layers correspond to copies of the original graph for routing before, between and after the specified nodes. The transformed graph enables state-of-the-art algorithms to solve the OSNE of their original problem.

A. Graph Transformation

MITH defines a new graph consisting of several layers. Each layer corresponds to an exact copy of the original graph, including the metrics associated to the different edges. The different layers are ordered and correspond to the routing before, between and after the different specified nodes. That is, if there are n specified nodes, $n + 1$ layers are defined. Subsequent layers are interconnected by connecting pairs of nodes corresponding to the same original node. An interconnecting edge corresponds to the visit of the original node corresponding to the nodes used for the interconnection. That is, the interconnections between the layers correspond to the visit of the different specified nodes. For a specified node with several candidates, the corresponding layers are simply interconnected several times through the corresponding nodes. The edge(s) created for interconnecting the layers can be assigned metric values corresponding to the visit of this node.

The procedure is illustrated in Fig. 3. Let us consider a request with two specified nodes ($n = 2$), the first one having

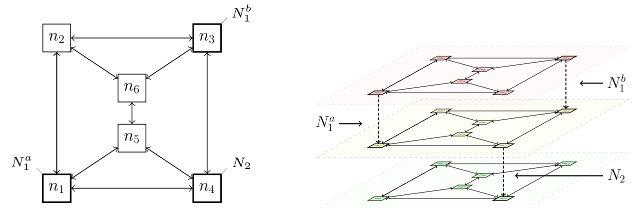

Fig. 3: On the left, example graph for the illustration of the *Mole in the Hole* (MITH) graph transformation algorithm. On the right, obtained transformed graph for a request with two specified nodes, the first one having two candidates (n_1 and n_3) and the second one having only one candidate (n_4).

two candidates ($N_1^a = n_1$ and $N_1^b = n_3$) and the second one having only a single candidate ($N_2 = n_4$). As there are two specified nodes, three layers are defined. The first two layers are interconnected via nodes n_1 and n_3, corresponding to the two candidates of the first specified node (N_1^a and N_1^b), and the last two layers are interconnected via node n_4, corresponding to the single candidate of the second specified node (N_2).

B. Request Transformation

The original request has to be mapped to the transformed graph. From a graph point of view, any routing request without specified intermediate nodes can be represented by a set of source nodes and a set of destination nodes. As the source nodes have to be visited before the first specified node, the source nodes in the transformed graph correspond to the copy of the source nodes of the original request in the first layer. Similarly, as the destination nodes have to be visited after the last specified node, the destination nodes in the transformed graph correspond to the copy of the destination nodes of the original request in the last layer.

C. Solution Transformation

Once the state-of-the-art algorithm returns a solution on the transformed graph, the latter has to be mapped back to the original graph. The solution in the original graph corresponds to the concatenation of all the original edges corresponding to the edges used in the different layers.

D. Limitations of MITH

While MITH presents itself as a general algorithm for enabling routing through specified nodes for any routing algorithm, it presents some limitations. First, if the original graph includes local constraint metrics (e.g., finite bandwidth usage capacity), the routing algorithm extended with MITH could lose its completeness or optimality [25]. Second, node- or edge-disjointness on the transformed graph does not ensure node- or edge-disjointness on the original graph. Hence, MITH does not support multipath routing. Third, as mentioned, MITH can only deal with an *ordered* set of specified nodes.

V. Evaluation

The goal of our evaluation is twofold. First, in Sec. V-A, we give an insight on the impact of MITH on the performance of state-of-the-art routing algorithms. Second, in Sec. V-B, as LARAC-SN and MITH can both solve the OSNE of the unicast CSP problem with one candidate per specified node, we compare the performance of both algorithms for this specific problem.

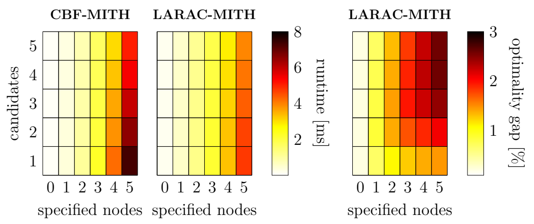

Fig. 4: Runtime and optimality gap of CBF with MITH (CBF-MITH) and LARAC with MITH (LARAC-MITH) for different numbers of specified nodes and candidates per specified node.

We perform both evaluations on the example of the unicast CSP problem. We use the topologies of the Topology Zoo [26] which are connected, have between 10 and 100 vertices and less than 200 edges. The delay d_i of an edge is defined as the propagation delay and its cost c_i is defined as $1+1/d_i$. This ensures that least-cost and least-delay paths are not identical. For each combination of topology, number of specified nodes and number of candidate nodes per specified node, we generate 100 random sets of candidate nodes. For each of these, we generate 100 random unicast requests (random source and destination) with a delay bound randomly uniformly distributed between the minimum (delay of the least-delay path) and maximum possible values (delay of the least-cost path). This setup leads, for each scenario, to 10.000 requests per combination of topology and number of specified and candidate nodes. The evaluations were ran on an Ubuntu 16.04 PC equipped with an Intel Core i7-4790 CPU @ 3.60GHz.

A. Influence of MITH on the Performance of Algorithms

We observe the runtime and optimality gap of *Constrained Bellman-Ford* (CBF) [27], an optimal CSP algorithm, and LARAC [6], [8], a CSP heuristic, for different numbers of specified nodes (from 0, i.e., without MITH, to 5) and for different numbers of candidate nodes per specified node (from 1 to 5). The results are shown in Fig. 4. Each point is represented by the average runtime or optimality gap observed for the 10.000 runs performed for this point.

1) Runtime: We observe that the impact of MITH on the runtime of the algorithms increases with the number of specified nodes. However, surprisingly, the runtime of the algorithms decreases with the number of candidate nodes. This is due to the fact that, because of the randomness of our evaluation, adding candidates potentially allows the algorithms to reach the last layer with shorter paths (i.e., faster).

2) Optimality Gap: In terms of optimality gap, with CBF as optimal benchmark, we observe that increasing the number of specified nodes and the number of candidate nodes per specified node both increase the optimality gap of LARAC.

B. LARAC-SN vs LARAC with MITH

Fig. 5 shows the runtime and optimality gap of LARAC-SN and LARAC and CBF extended with MITH for different numbers of specified nodes (from 0 to 8). Each point in the graph corresponds to the average of the values observed for the 10.000 runs corresponding to this point.

1) Runtime: As expected, since increasing the number of specified nodes increases the size of the graph on which path finding is performed, the runtime of CBF and LARAC with MITH increases exponentially with the number of specified

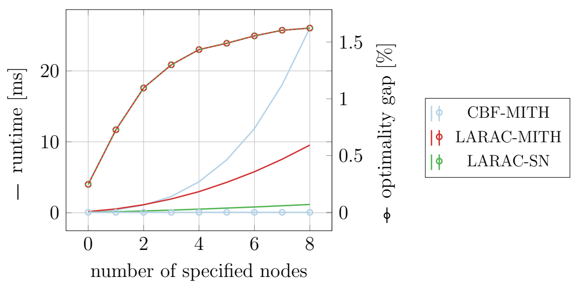

Fig. 5: Runtime and optimality gap of LARAC-SN, CBF with MITH (CBF-MITH) and LARAC with MITH (LARAC-MITH) for different numbers of specified nodes and one candidate per specified node.

nodes. As already observed by previous studies [7], LARAC scales better than CBF. We observe that LARAC-SN presents a much better runtime behavior. Indeed, the runtime of LARAC-SN is around 10% of the runtime of LARAC with MITH.

2) Optimality Gap: At the cost of its high runtime, CBF with MITH is optimal. On the contrary, LARAC-SN and LARAC with MITH are both sub-optimal but their optimality gap stays lower than 1.62% in average. They both exhibit the exact same optimality gap behavior. Interestingly, while the optimality gap increases with the number of specified nodes, the increase seems to saturate.

VI. Conclusions

The problem of routing through intermediate specified nodes has become more and more an important issue. Indeed, the problem finds applications in new emerging fields such as *Software-Defined Networking* (SDN), *Network Function Virtualization* (NFV) and *Service Function Chaining* (SFC). In this paper, we propose two solutions for this problem. First, we propose *LARAC for specified nodes* (LARAC-SN), an extension of the state-of-the-art LARAC *constrained shortest path* (CSP) algorithm which allows to find a CSP visiting specified intermediate nodes. While sub-optimal, we show that LARAC-SN exhibits a small optimality gap at a low runtime cost. Second, we propose *Mole in the Hole* (MITH), a graph transformation algorithm forcing any state-of-the-art algorithm to visit an ordered set of specified nodes. MITH further allows to define a set of candidate nodes per specified node such that the state-of-the-art routing algorithm used automatically selects the best candidate. We show that MITH has the potential of finding the optimal solution to any problem, however at the cost of an increased runtime. We further show that the performance of both LARAC-SN and MITH degrades as the number of specified nodes increases. As LARAC-SN shows a very good trade-off between optimality gap and runtime, future work should consider the extension of LARAC-SN to deal with several candidates per specified node. Further, MITH generates graphs with a specific layered structure. The thorough evaluation of the performance of state-of-the-art algorithms on MITH-like graphs is also an interesting research direction.

Acknowledgments

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 671648 (VirtuWind).

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] J. L. Gross and J. Yellen, *Graph theory and its applications*. CRC Press, 2005.

[3] P. Paul and S. Raghavan, "Survey of QoS routing," in *Proc. of the Int. Conference on Computer Comm.*, vol. 15, no. 1, 2002, p. 50.

[4] R. Mijumbi *et al.*, "Network function virtualization: state-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

[5] A. Basta *et al.*, "Applying NFV and SDN to LTE mobile core gateways, the functions placement problem," in *Proc. of the 4th SIGCOMM Workshop on All Things Cellular*. ACM, 2014, pp. 33–38.

[6] Y. P. Aneja and K. P. Nair, "The constrained shortest path problem," *Naval Research Logistics (NRL)*, vol. 25, no. 3, pp. 549–555, 1978.

[7] J. W. Guck, A. Van Bemten, M. Reisslein, and W. Kellerer, "Unicast QoS routing algorithms for SDN: A comprehensive survey and performance evaluation," *IEEE Communications Surveys & Tutorials*, 2017.

[8] A. Jüttner *et al.*, "Lagrange relaxation based method for the QoS routing problem," in *Proc. of IEEE INFOCOM*, vol. 2, 2001, pp. 859–868.

[9] J. Saksena and S. Kumar, "The routing problem with k specified nodes," *Operations Research*, vol. 14, no. 5, pp. 909–913, 1966.

[10] S. E. Dreyfus, "An appraisal of some shortest-path algorithms," *Operations research*, vol. 17, no. 3, pp. 395–412, 1969.

[11] T. Ibaraki, "Algorithms for obtaining shortest paths visiting specified nodes," *Siam Review*, vol. 15, no. 2, pp. 309–317, 1973.

[12] T. Gomes *et al.*, "Protected shortest path visiting specified nodes," in *7th International Workshop on Reliable Networks Design and Modeling (RNDM)*. IEEE, 2015, pp. 120–127.

[13] R. C. de Andrade, "New formulations for the elementary shortest-path problem visiting a given set of nodes," *European Journal of Operational Research*, vol. 254, no. 3, pp. 755–768, 2016.

[14] L. Martins *et al.*, "An efficient heuristic for calculating a protected path with specified nodes," in *8th International Workshop on Resilient Networks Design and Modeling (RNDM)*. IEEE, 2016, pp. 150–157.

[15] T. Gomes, L. Martins, S. Ferreira, M. Pascoal, and D. Tipper, "Algorithms for determining a node-disjoint path pair visiting specified nodes," *Optical Switching and Networking*, vol. 23, pp. 189–204, 2017.

[16] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *IEEE 4th International Conference on Cloud Networking (CloudNet)*, Oct 2015, pp. 171–177.

[17] M. Rost and S. Schmid, "Service chain and virtual network embeddings: Approximations using randomized rounding," *arXiv preprint arXiv:1604.02180*, 2016.

[18] M. Ghaznavi, N. Shahriar, R. Ahmed, and R. Boutaba, "Service function chaining simplified," *arXiv preprint arXiv:1601.00751*, 2016.

[19] P. Vizarreta *et al.*, "QoS-driven function placement reducing expenditures in NFV deployments," in *Proceedings of IEEE ICC*, 2017.

[20] W. Ma, C. Medina, and D. Pan, "Traffic-aware placement of NFV middleboxes," in *Proc. of GLOBECOM*. IEEE, 2015, pp. 1–6.

[21] T. W. Kuo *et al.*, "Deploying chains of virtual network functions: On the relation between link and server usage," in *IEEE INFOCOM 2016*, April 2016, pp. 1–9.

[22] M. F. Bari *et al.*, "On orchestrating virtual network functions," in *Proc. of CNSM*. IEEE, 2015, pp. 50–56.

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

[24] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[25] A. Van Bemten *et al.*, "Routing metrics depending on previous edges: The Mn taxonomy and its corresponding solutions," in *Proceedings of IEEE ICC*, 2018.

[26] S. Knight *et al.*, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765 –1775, October 2011.

[27] R. Widyono, "The design and evaluation of routing algorithms for real-time channels," International Computer Science Institute Berkeley, Tech. Rep. TR-94-024, 1994.