Introducing the Virtual Network Mapping Problem with Delay, Routing and Location Constraints

Johannes Inführ and Günther R. Raidl

Abstract Network virtualization is a main paradigm of Future Internet research. It allows for automatic creation of virtual networks with application specific resource management, routing, topology and naming. Since those virtual networks need to be implemented by means of the underlying physical network, the Virtual Network Mapping Problem (VNMP) arises. In this work, we introduce the Virtual Network Mapping Problem with Delay, Routing and Location Constraints (VNMP-DRL), a variant of the VNMP including some practically relevant aspects of Virtual Network Mapping that have not been considered before. We describe the creation of a benchmark set for the VNMP-DRL. The main goal was to include VNMP-DRL instances which are as realistic as possible, a goal we met by using parts of real network topologies to model the physical networks and by using different classes of virtual networks to model possible use-cases, instead of relying on random graphs. As a first approach, we solve the VNMP-DRL benchmark set by means of a multicommodity flow integer linear program.

Key words: Virtual Network Mapping, Network Virtualization, Integer Linear Programming, Multicommodity Flow

1 Introduction

Network virtualization has been identified as a main paradigm of Future Internet research [3, 4] because it helps to overcome the ossification of the internet [15]. In this context, ossification means that it is very hard or even impossible to replace or fundamentally change a widespread technology, such as internet protocols. With the help of virtualization, such changes can be implemented in an incremen-

Johannes Inführ and Günther R. Raidl
Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstraße 9-11, 1040 Vienna, Austria, e-mail: {infuehr|raidl}@ads.tuwien.ac.at

tal and non-disruptive manner. Another viewpoint is that virtualization is not only useful to switch technologies, but allows the coexistence of different technologies with different tradeoffs, each targeting a different user group. Network virtualization techniques have already been successfully used in scientific network testbeds such as GENI [2], PlanetLab [8] or G-Lab [17]. Its area of application is the splitting of a shared underlying network infrastructure (substrate) into virtual networks (slices), which are under the full control of different research groups for their experiments. The properties of the slices can be controlled by the experimenters and are not simply the result of the used substrate network. It is even possible to implement custom resource management, routing and naming on a per slice basis. One main idea of Future Internet research is that these mechanisms can be directly transferred to the internet, to be able to create application specific virtual networks, specifically tailored to each application. Network virtualization also enables application specific choice of internet service provider, depending on the network characteristics of those providers and the application's requirements.

In such a scenario, the question naturally arises of how to map a set of virtual networks, each with its specific performance requirements, onto the existing network, which is the core of the Virtual Network Mapping Problem (VNMP). As is often the case, the problems are hidden in the details. For the VNMP, there is no standard set of requirements of virtual networks or properties of the substrate network, or even a clearly specified aim. For our work, we use the common properties of bandwidth (supplied by links of the substrate network and required by links of virtual networks) and CPU power (supplied by the substrate nodes and required by the virtual network nodes to implement custom protocols). In addition, we use communication delay on arcs (transporting data across a link in the substrate network incurs a delay and each virtual link has a specified maximum allowed value for such delay) and routing capacity on nodes (in most cases, routers can not route the full bandwidth with which they are connected). One additional class of constraints we consider is the possible placement of virtual nodes. In practice, users of a virtual network are located at specific positions in the substrate network and cannot be relocated to positions where it would be more suitable. Our goal will be to use the cheapest subset of substrate resources to satisfy all virtual network demand. We call this problem the Virtual Network Mapping Problem with Delay, Routing and Location constraints (VNMP-DRL).

Formally, the VNMP-DRL is defined as follows: We are given a directed multigraph $G = (V,A)$ with node set V and arc set A representing the substrate network. Additionally given is the available CPU power of a substrate node $c_i \in \mathbb{N}^+,\ \forall i \in V$, the amount of bandwidth units that can be routed by a substrate node $r_i \in \mathbb{N}^+,\ \forall i \in V$, the cost of using a substrate node as host for virtual nodes $p_i^V \in \mathbb{N}^+,\ \forall i \in V$, the delay of a substrate arc $d_e \in \mathbb{N}^+,\ \forall e \in A$, the available bandwidth of a substrate arc $b_e \in \mathbb{N}^+,\ \forall e \in A$ and the cost of using a substrate arc to implement virtual connections $p_e^A \in \mathbb{N}^+,\ \forall e \in A$. The slices are given by (the components of) the directed graph $G' = (V',A')$ (the virtual network graph), with node set V' and arc set A'. Associated with the slices is the required CPU power by a virtual node $c_k \in \mathbb{N}^+,\ \forall k \in V'$, the allowed delay on the path implementing a virtual connec-

tion $d_f \in \mathbb{N}^+,\ \forall f \in A'$ and the required bandwidth on the path implementing a virtual connection $b_f \in \mathbb{N}^+,\ \forall f \in A'$. The set $M \subseteq V' \times V$ defines the allowed mappings between virtual and substrate nodes. The functions $s : A \cup A' \to V \cup V'$ and $t : A \cup A' \to V \cup V'$ associate each arc of G and G' with their source and target nodes respectively. The objective is to find an assignment of the virtual nodes to substrate nodes (subject to the allowed mappings and CPU constraints) and for each virtual arc a path in the substrate network from the location of the virtual source node to the location of the virtual target node in the substrate network (subject to routing, bandwidth and delay constraints), so that the total cost of used substrate nodes and arcs is minimized.

2 Related Work

The VNMP has been solved in diverse variants [6, 10, 11, 14, 16, 19, 20, 22] and under various names (Virtual Network Mapping, Virtual Network Assignment, Network Testbed Mapping) in the literature. The solution methods that have been applied to VNMP variants include (quadratic) mixed integer programming [6, 10, 14], approximation algorithms [10], simulated annealing [16], distributed algorithms [11], multicommodity flow algorithms [19, 20] or algorithms especially tailored to the considered problem variant [22].

One type of virtual network demand considered by nearly all works is the required bandwidth, but how it is taken into account varies. One method is to use traffic bounds to describe a whole range of bandwidth requirements that all have to be feasibly routed (e.g. [10, 14]), another is to specify the node-to-node communication demand in the form of a traffic matrix (e.g. [19]). If another requirement is taken into account, it is usually the required CPU processing power of each virtual node (e.g. [20]).

In most cases, the aim of optimization is a tradeoff between the cost of the mapping and load balancing on the substrate nodes ([6, 11]) but also other aspects are taken into consideration, such as reliability requirements [20], configuration costs [14] or node and link stress (the amount of virtual nodes or links mapped to a single substrate node or link) [22].

The considered substrate sizes vary between 20 [14] and 100 [22] nodes and are either real topologies or generated by tools such as GT-ITM [21]. The requested virtual networks are mostly random graphs and consist of about ten nodes. All the cited works use undirected or directed graphs to model the substrate and virtual networks, to the best of our knowledge this is the first work to consider substrate multigraphs. We chose multigraphs because there may be multiple connections between nodes of the substrate networks with different characteristics, and one has to be able to represent those in a natural manner.

On the tropic of network virtualization, its application and available technologies, see [7] for a survey.

3 The Model

In this section we present a multicommodity flow based mixed integer programming formulation for the VNMP-DRL. It utilizes the decision variables $x_{ki} \in \{0,1\},\ \forall k \in V',\ \forall i \in V$ to indicate where the virtual nodes are located in the substrate graph and $y_e^f \in \{0,1\},\ \forall f \in A',\ \forall e \in A$ to indicate if a virtual connection is implemented by using a substrate connection. Further auxiliary decision variables are $z_i^f \in \{0,1\},\ \forall f \in A',\ \forall i \in V$ to indicate that a substrate node is used to route a virtual connection, $u_i^V \in \{0,1\},\ \forall i \in V$ to indicate that a substrate node hosts at least one virtual node and $u_e^A \in \{0,1\},\ \forall e \in A$ to indicate that a substrate arc is used for at least one virtual connection.

Now follows the multicommoditiy flow (MCF) based integer linear programming model we use to solve the VNMP-DRL.

$$\text{(FLOW)} \quad \min \quad \sum_{i \in V} p_i^V u_i^V + \sum_{e \in A} p_e^A u_e^A \tag{1}$$

$$\sum_{(k,i) \in M} x_{ki} = 1 \qquad \forall k \in V' \tag{2}$$

$$\sum_{e \in A | t(e) = i} y_e^f + x_{s(f)i} - \sum_{e \in A | s(e) = i} y_e^f - x_{t(f)i} = 0 \qquad \forall i \in V,\ \forall f \in A' \tag{3}$$

$$\sum_{e \in A | t(e) = i} y_e^f + x_{s(f)i} \leq z_i^f \qquad \forall i \in V,\ \forall f \in A' \tag{4}$$

$$\sum_{(k,i) \in M} c_k x_{ki} \leq c_i \qquad \forall i \in V \tag{5}$$

$$\sum_{f \in A'} b_f z_i^f \leq r_i \qquad \forall i \in V \tag{6}$$

$$\sum_{f \in A'} b_f y_e^f \leq b_e \qquad \forall e \in A \tag{7}$$

$$\sum_{e \in A} d_e y_e^f \leq d_f \qquad \forall f \in A' \tag{8}$$

$$x_{ki} \leq u_i^V \qquad \forall i \in V,\ \forall k \in V' \tag{9}$$

$$y_e^f \leq u_e^A \qquad \forall e \in A,\ \forall f \in A' \tag{10}$$

$$x_{ki} = 0 \qquad \forall (k,i) \in (V' \times V) \backslash M \tag{11}$$

$$x_{ki} \in \{0,1\} \qquad \forall (k,i) \in M \tag{12}$$

$$y_e^f \in \{0,1\} \qquad \forall e \in A,\ \forall f \in A' \tag{13}$$

$$z_i^f \in \{0,1\} \qquad \forall i \in V,\ \forall f \in A' \tag{14}$$

Equalities (2) ensure that each virtual node is mapped to exactly one substrate node, subject to the mapping constraints. The flow conservation constraints (3) make sure that for each virtual connection there is a connected path in the substrate net-

Table 1: Summary of the used variables, constants and functions of the MCF formulation of the VNMP-DRL ($i \in V$, $e \in A$, $k \in V'$, $f \in A'$, $l \in A \cup A'$)

Symbol	Meaning	Symbol	Meaning	Symbol	Meaning
$G(V,A)$	Substrate graph	$G'(V',A')$	Virtual graph	x_{ki}	Map node k to i
c_i	Av. CPU	c_k	Req. CPU	y_e^f	Use arc e for f
d_e	Delay	d_f	Max. allowed delay	z_i^f	Use node i for f
b_e	Av. bandwidth	b_f	Req. bandwidth	u_i^V	Use node i
r_i	Av. routing capacity	M	Set of allowed mappings	u_e^A	Use arc e
p_i^V	Node price	$s(l)$	Source node of arc l		
p_e^A	Arc price	$t(l)$	Target node of arc l		

work. Linking constraints (4) make certain that variables z_i^f are equal to one when the corresponding node is used to route the traffic of a particular virtual connection. Inequalities (5)–(8) ensure that the solutions are valid with regard to CPU, routing capacity, bandwidth and delay constraints. Linking constraints (9) and (10) force variables u_i^V and u_e^A to be (at least) one when the corresponding substrate node or arc is used by any virtual node or arc. Constraints (11) exclude any forbidden mappings from the solution. Note that while the model only includes integrality constraints for x_{ki}, y_e^f and z_i^f (12)–(14), constraints (9) and (10) together with the objective function (1) also cause variables u_i^V and u_e^A to be integral (and binary). Table 1 gives a short reference of the used variables, constants and functions.

4 Generating the Benchmark-Instances

This section describes how the VNMP-DRL benchmark set was created, by first illustrating how the components of a benchmark instance were created and then how they were combined to form a complete instance. The main goal was to create hard VNMP-DRL instances which are as realistic as possible, to serve as a common basis for the comparison of different VNMP-DRL solution approaches. The benchmark set can be obtained at [13].

4.1 Substrate Network

The substrate networks were based on real internet topology maps. The base data came from the Rocketfuel project [18] and the scan-lucent map (the union of the topologies measured by the SCAN [9] and Lucent [5] internet mapping projects). To create networks of the required size, nem-0.9.6 was used to extract subgraphs which retain the main characteristics of the source graph. In the cases where that was not possible (when the required network size exceeded 30 percent of the source network

size), nodes with in- and out-degree of at most one were randomly deleted until the target size was reached. In absence of such nodes, random nodes were deleted. No measures to ensure connectivity were taken because the source networks were not always connected themselves.

Since the Rocketfuel graphs (rf) have assigned latencies to their arcs, those were used as delay values d_e. For the graphs generated from the scan-lucent (sl) map delay values were chosen uniformly at random between 1 and 10. Arcs which were the only connection of a node were assigned a bandwidth b_e of 25. The other arcs were assigned a bandwidth value of 25 times the minimum of the in-degree of their source node and the out-degree of their target node, but at least 25. The routing capacity r_i of a node was calculated as the minimum of the sum of the incoming and outgoing bandwidth. The available CPU capacity of a node c_i was set to be the same as the routing capacity. The costs of nodes and edges are chosen uniformly at random between 1 and 20.

Figure 1 shows a generated substrate graph based upon the rf1221 topology of size 20.

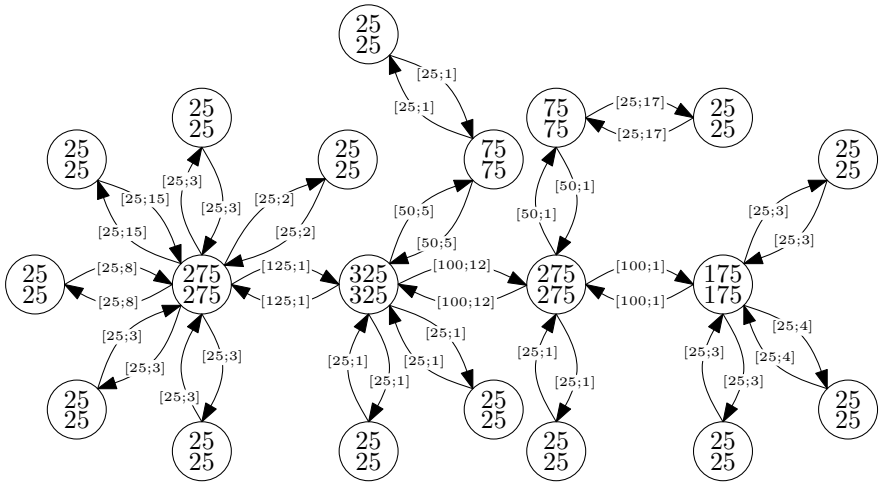

Fig. 1: Example substrate graph of size 20, generated from the rf1221 topology map. The node labels are the routing capacity r_i and the available CPU c_i, the arc labels are the bandwidth b_e and the delay values d_e.

4.2 Slices

In this work, we used four different slice-types to represent possible use-cases in the virtual network setting with different sets of requirements regarding needed band-

width, needed processing power per node and maximum delay. Those types were web slices to represent general http-traffic, stream slices to represent video streaming, P2P slices to represent P2P networks and VoIP slices to represent voice chat.

Web slice The general characteristics of web slices were chosen to be low bandwidth requirements, short delays (for fast responses) and no special CPU requirements. Web slices were modeled by a star graph, where the central node represented a web server and the leafs were the users. All edges were assigned a bandwidth requirement of 1 and a maximum allowed delay of 25. All nodes were assigned a CPU requirement of 1, except the root node which was assigned the sum of the outgoing bandwidth as CPU requirement. The allowed mapping calculated for web slices placed the leaf nodes of the star at a random location at the edge of the substrate network (which was defined as the set of nodes with minimal degree), while the central node was placed at a random location at the core of the network (which was defined as not being the edge). Figure 2a shows a generated web slice of size 5.

Stream slice The general characteristics of stream slices were chosen to be medium to high bandwidth requirements, no relevant delay bounds and 3 units of CPU processing power per routed bandwidth. The idea of the stream slices was to use a random tree graph, where the root node is the source of the video stream, the leafs are the customers receiving specific channels of the video stream and the intermediate nodes split the video stream and forward only the channels which are watched by the customers. The stream splitting is the reason for the high CPU requirements in relation to bandwidth units. This is an example of a customized routing protocol which delivers more features than currently possible. The delay bound of the arcs of the stream slices was set to 1000, which effectively means that they are not relevant. The number of channels in the video stream was chosen uniformly at random between 10 and 20, while the total bandwidth requirement of the stream was chosen from a discrete $N(5,1)$ distribution, but at least 3 and at most 7. Each child node in the stream network only received a random fraction of the channels the parent received (between 0.3 and 1), but it was made sure that all channels that a node received were forwarded. The bandwidth requirements of each arc were set to $\lceil(\text{bandwidthPerChannel} * \text{forwardedChannels}\rceil$ and the CPU requirement of each node to three times the received bandwidth. The calculated allowed mapping for stream slices placed the root node at a random location at the core of the substrate network and the leaf nodes at a random location at the edge of the substrate network. The leaf nodes were placed in a way so that the distance between siblings was less or equal to four hops in the substrate network. The placement of the intermediate node was not constrained, i.e. they were allowed to be mapped anywhere. Figure 2b shows a generated stream slice of size 5.

P2P slice The general characteristics of P2P slices were chosen to be medium bandwidth requirements, no relevant delay bounds and medium CPU requirements. The network structure of a P2P slice was generated by the small_world_iterator of the boost graph library [1], after which every arc was duplicated and reversed. The bandwidth requirement of each arc was chosen uniformly at random between 1 and

3, the delay bound was set to 1000 and the CPU requirement of the nodes was chosen uniformly at random between 1 and 5. The CPU requirement was chosen independent of the routed bandwidth to model that some traffic may be encrypted or compressed and therefore has a higher computational demand. All nodes of the P2P slice were only allowed to be mapped to one random node at the edge of the substrate network. Figure 2c shows a generated P2P slice of size 5.

VoIP slice The general characteristics of VoIP slices were chosen to be medium bandwidth requirements, medium delays and high CPU requirements. The networks structure of VoIP slices was generated in the same way as P2P slices. The bandwidth requirement of each arc was chosen uniformly at random between 1 and 3 and the delay bound was set to 50. The CPU requirement was set to the minimum of incoming and outgoing bandwidth, so that "super-nodes" (which route a lot of VoIP traffic) have high CPU requirements. Figure 2d shows a generated VoIP slice of size 5.

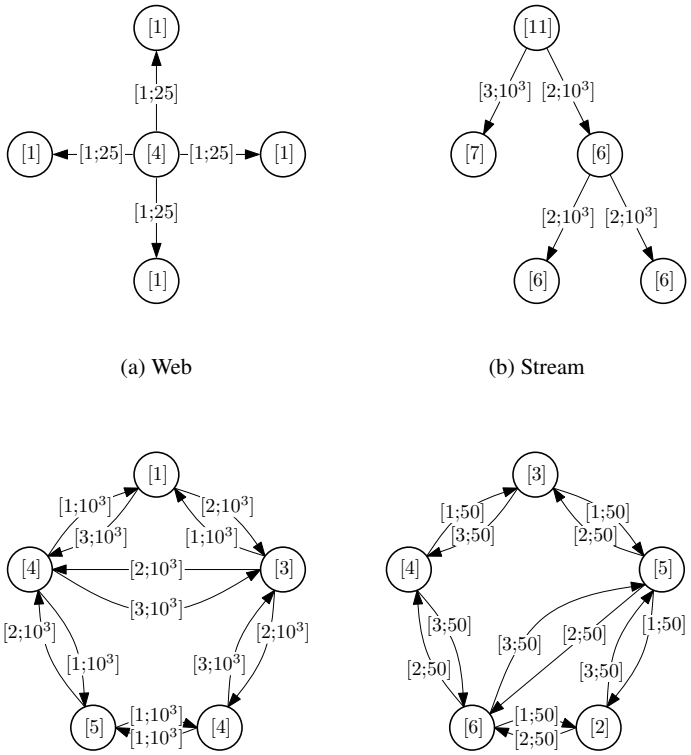

Fig. 2: Examples of generated slices of size 5

4.3 VNMP-DRL Instance

To generate a complete VNMP-DRL instance, first a specific topology map is used to create a substrate graph (in conjunction with its associated costs) of the required size. Then the virtual graph is build by adding slices (of random type and size between 10 and 20 percent of the substrate graph size) to the virtual graph, until the problem becomes "too hard" (see next paragraph). If that happens, the last added slice is removed and a new one is generated and added. If this process fails to find an addable slice 40 times in a row, the generation process is finished. From this generated instance, five variants are constructed by only using 50 to 90 percent of the added slices (in ten percent increments, each variant includes all the slices used by a smaller variant), so the complete generation process creates six problem instances of incremental difficulty.

We tried different criteria to define "too hard", for instance only solving the LP-relaxation of FLOW and rejecting virtual graphs which cause the relaxation to be unsolvable or only calculating the root node of the branch-and-bound tree and rejecting virtual graphs which are by then proven to be unsolvable. However, preliminary runs showed that the instances that were generated in this way were either extremely easy to solve to optimality (without branching) or provably unsolvable. Especially for larger instance sizes the fraction of generated unsolvable instances became unmanageable. The hardness definition we used in the end was that if CPLEX 12.2 [12] is not able to find an integer solution after 300 seconds once in five tries, the virtual graph is rejected as "too hard". In each of the five trials, the sequence in which the slices are added to the virtual graph is permuted, because preliminary runs showed that in some cases a bad sequence can double the time needed to solve the LP relaxation and that time is missed afterwards when trying to find an integer solution.

The used substrate sizes for instance creation were 20, 30, 40, 50, 70 and 100. For each of the seven topology maps (six rf and one sl) the problem generation procedure described previously was used ten times, so in total 420 instances per size were created, with the exception of substrate size 100, for which only 300 instances were created because two of the rocketfuel topology maps contain less than 100 nodes.

5 Computational Results

All results presented in this section have been achieved by using CPLEX 12.2 to solve the (FLOW) problem. Each computational experiment has been performed on one core of an Intel Xeon E5540 multi-core system with 2.53 GHz and 3 GB RAM per core. A time limit of 10000 seconds was used. The reported gaps are the optimality gaps calculated by CPLEX.

5.1 Created Benchmark-Set

The basic properties of the generated benchmark instances are shown in Table 2. It can be seen that while the substrate grows, the number of slices contained in the virtual network graph does not grow and for the sizes 70 and 100 is even less than the number of slices of instances of size 20. The total size of the virtual network however does not shrink. From this we can conclude that because larger instances include larger slices (keep in mind that slices are created with a size between 10 and 20% of the substrate size), fewer slices can be mapped onto the substrate network, even though it is bigger and should be able to carry more slices. Also noteworthy is the slice composition development. From the description of the instance generation procedure, one would expect an equal number of slice types in every virtual network graph. It seems to be the case that it is harder to find P2P and VoIP slices that can be mapped, than it is for the other slice types. For VoIP slices, this effect gets worse as the substrate size grows. Also finding mappable Web slices gets harder with increasing substrate size. The reason for this behaviour could be, that Stream slices are not delay bound, while Web and VoIP slices are, so when the substrate grows, so does the average delay between two points at the edge of the substrate network and so paths within the delay bounds become more scarce.

The influence of the choice of topology source for the substrate network on the generated benchmark instances can be seen in Table 3. Note that the average substrate size for the rf1775 and rf3967 instances is smaller than those of the other instances because those topologies were too small to create instances of size 100. It can be seen that the sl instances are the source of the sparsest substrate graphs in the benchmark set. This leads to the highest number of slices in the virtual network graph, but web and stream slices are highly overrepresented. Also note that it seems to be very hard to find mappable web slices when using rf3967 and rf6461 as substrate topology source, even though it is not problematic for all other topology sources.

Table 2: Overview of the properties of the created benchmark instances: Average size of substrate graph $G = (V,A)$, average size of virtual network graph $G' = (V',A')$, average number of slices contained in G' ($\#S$) and average fraction of slice types contained in the virtual network graph (Web, Stream, P2P, VoIP)

Size	$\lvert V\rvert$	$\overline{\lvert A\rvert}$	$\overline{\lvert V'\rvert}$	$\overline{\lvert A'\rvert}$	$\overline{\#S}$	$\overline{\text{Web}}$	$\overline{\text{Stream}}$	$\overline{\text{P2P}}$	$\overline{\text{VoIP}}$
20	20	53.3	72.4	81.9	14.5	0.36	0.36	0.13	0.15
30	30	87.3	101.1	117.9	20.2	0.31	0.37	0.19	0.14
40	40	126.9	104.8	145.7	18.7	0.28	0.42	0.18	0.12
50	50	172.9	111.8	174.2	16.7	0.27	0.44	0.19	0.11
70	70	253.1	100.8	140.9	10.5	0.25	0.54	0.14	0.07
100	100	386.4	119.0	156.9	8.7	0.19	0.59	0.16	0.06

Table 3: Influence of the chosen topology source on the created benchmark instances: Average size of substrate graph $G = (V,A)$, average size of virtual network graph $G' = (V',A')$, average number of slices contained in G' (#S) and average fraction of slice types contained in the virtual network graph (Web, Stream, P2P, VoIP)

Top. source	$\overline{\lvert V\rvert}$	$\overline{\lvert A\rvert}$	$\overline{\lvert V'\rvert}$	$\overline{\lvert A'\rvert}$	$\overline{\#S}$	$\overline{\text{Web}}$	$\overline{\text{Stream}}$	$\overline{\text{P2P}}$	$\overline{\text{VoIP}}$
rf1221	51.7	164.7	120.5	163.3	17.4	0.34	0.43	0.12	0.11
rf1239	51.7	202.0	75.8	95.3	11.1	0.35	0.44	0.10	0.12
rf1755	42.0	141.2	108.6	147.0	18.4	0.31	0.42	0.14	0.13
rf3257	51.7	197.4	76.5	112.9	11.3	0.29	0.38	0.19	0.14
rf3967	42.0	140.4	93.7	141.6	16.1	0.07	0.56	0.26	0.12
rf6461	51.7	207.3	73.3	113.1	11.0	0.10	0.54	0.29	0.07
sl	51.7	125.0	157.2	176.4	21.7	0.47	0.37	0.09	0.07

5.2 Solving the VNMP-DRL

The results of solving the benchmark instances with the FLOW formulation are summarized in Table 4. It can be seen that in general, the generated instances are not very hard to solve, with at least 74.3% instances solved to optimality per substrate size. Interestingly, the hardest instances turned out to be those of size 50 (or 70 when regarding the average gap). In total, the results show that up to substrate sizes of 100 nodes it is possible to solve the VNMP-DRL to proven optimality within one hour on average.

Table 5 shows the influence of the chosen source topology on the solution characteristics of the instances. Unsurprisingly, instances based on the sl topology map are the easiest to solve, as the substrate graphs are very sparse, which reduces the routing and mapping possibilities. But also the rf instances vary a lot, which indicates that the hardness of the VNMP-DRL is very sensitive to the choice of the substrate topology.

The influence of the fraction of the set of slices (denoted as pS), created during instance creation, on the hardness of the created instances is presented in Table 6. It can be seen that the complexity of the instances can be selected very well by an appropriate choice of pS. From another point of view this means that the more slices

Table 4: Results of solving the VNMP-DRL with the FLOW formulation: Number of instances (# Inst.), fraction of instances solved to optimality (# Opt [%]), average gap, average number of branch-and-bound nodes and average required CPU time

Size	# Inst.	# Opt [%]	$\overline{\text{gap [\%]}}$	$\overline{\text{BB-Nodes}}$	$\overline{\text{t [s]}}$
20	420	100.0	0.00	53.5	8
30	420	93.8	0.12	1352.3	758
40	420	85.7	0.36	1955.3	1842
50	420	74.3	0.52	1176.8	3320
70	420	82.9	0.60	420.0	2465
100	300	90.7	0.20	218.1	1859

Table 5: Influence of the substrate topology source: Number of instances (# Inst.), fraction of instances solved to optimality (# Opt [%]), average gap, average number of branch-and-bound nodes and average required CPU time

Top. source	# Inst.	#Opt [%]	$\overline{\text{gap [\%]}}$	$\overline{\text{BB-Nodes}}$	$\overline{\text{t [s]}}$
rf1221	360	87.8	0.19	961.2	1781
rf1239	360	94.2	0.14	193.0	902
rf1755	300	80.7	0.70	2336.7	2736
rf3257	360	85.3	0.29	400.9	2087
rf3967	300	73.7	0.79	2153.1	3145
rf6461	360	89.2	0.17	644.8	1650
sl	360	100.0	0.00	24.7	20

Table 6: Influence of pS: Number of instances (# Inst.), fraction of instances solved to optimality (# Opt [%]), average gap, average number of branch-and-bound nodes and average required CPU time

pS	# Inst.	#Opt [%]	$\overline{\text{gap [\%]}}$	$\overline{\text{BB-Nodes}}$	$\overline{\text{t [s]}}$
0.5	400	97.2	0.05	547.1	555
0.6	400	95.5	0.07	766.8	862
0.7	400	91.2	0.17	912.7	1449
0.8	400	85.8	0.33	941.3	1935
0.9	400	81.8	0.44	1026.1	2397
1	400	75.0	0.77	1175.4	3010

are added, the harder the instance gets. This seems like an obvious statement, but there were some instances in the benchmark set for which the pS=0.8 or pS=0.9 variants were harder to solve than the pS=1 variant, probably because the pS=1 variants were so densely packed with slices that the number of routing possibilities was greatly reduced.

6 Conclusion and Future Work

In this work, we introduced the Virtual Network Mapping Problem with Delay, Routing and Location constraints (VNMP-DRL). We presented a method for creating realistic benchmark instances for this problem and solved those instances by means of a multicommodity flow based integer linear program. Even this simple approach was able to solve more than 74% of problem instances to proven optimality in less than one hour on average. The biggest influence on instance hardness was shown to be the topology map used to create the instance. Larger problem instances were not harder to solve than smaller instances on average, instances of size 100 were about as easily solved as instances of size 40. We were able to show that the fraction of slices used of the complete set of slices found during instance creation can be used to control instance hardness.

Future work will include improvements of the instance creation method, so that larger instances are actually harder than smaller instances and scaling the substrate network size up to 1000 nodes. Another venue is the application of more advanced ILP solution methods like branch-and-price to improve solution time and quality and eventually of heuristic methods to solve large problem instances. One simplifying assumption of VNMP-DRL was that the set of slices is known in advance, so future research could target the online aspect of this problem. Another interesting research direction could be analyzing the effect of rising delays with rising link utilization, which is currently not modelled.

Acknowledgements We would like to thank Kurt Tutschku and David Stezenbach for their invaluable support and helpful discussions. This research was funded by the WWTF, project number ICT10-027.

References

1. Boost C++ Libraries, http://www.boost.org
2. GENI.net Global Environment for Network Innovations, http://www.geni.net
3. Anderson, T., Peterson, L., Shenker, S., Turner, J.: Overcoming the Internet impasse through virtualization. Computer **38**(4), 34 – 41 (2005)
4. Berl, A., Fischer, A., de Meer, H.: Virtualisierung im future internet. Informatik-Spektrum **33**, 186–194 (2010)
5. Burch, H., Cheswick, B.: Mapping the internet. Computer **32**(4), 97 –98, 102 (1999)
6. Chowdhury, N., Rahman, M., Boutaba, R.: Virtual network embedding with coordinated node and link mapping. In: INFOCOM 2009, IEEE, pp. 783 –791 (2009)
7. Chowdhury, N.M.K., Boutaba, R.: A survey of network virtualization. Computer Networks **54**(5), 862 – 876 (2010)
8. Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., Bowman, M.: Planetlab: an overlay testbed for broad-coverage services. SIGCOMM Comput. Commun. Rev. **33**, 3–12 (2003)
9. Govindan, R., Tangmunarunkit, H.: Heuristics for internet map discovery. In: INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 3, pp. 1371 –1380 vol.3 (2000)
10. Gupta, A., Kleinberg, J., Kumar, A., Rastogi, R., Yener, B.: Provisioning a virtual private network: a network design problem for multicommodity flow. In: Proceedings of the thirty-third annual ACM symposium on Theory of computing, STOC '01, pp. 389–398. ACM, New York, NY, USA (2001)
11. Houidi, I., Louati, W., Zeghlache, D.: A distributed virtual network mapping algorithm. In: Communications, 2008. ICC '08. IEEE International Conference on, pp. 5634 –5640 (2008)
12. IBM ILOG: CPLEX 12.2. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer
13. Inführ, J., Raidl, G.R.: The VNMP-DRL benchmark set. https://www.ads.tuwien.ac.at/projects/optFI/
14. Lu, J., Turner, J.: Efficient mapping of virtual networks onto a shared substrate. Tech. rep., Washington University in St. Louis (2006)
15. National Research Council: Looking Over the Fence at Networks. National Academy Press (2001)
16. Ricci, R., Alfeld, C., Lepreau, J.: A solver for the network testbed mapping problem. SIGCOMM Comput. Commun. Rev. **33**(2), 65–81 (2003)

17. Schwerdel, D., Günther, D., Henjes, R., Reuther, B., Müller, P.: German-lab experimental facility. In: A. Berre, A. Gómez-Pérez, K. Tutschku, D. Fensel (eds.) Future Internet - FIS 2010, *Lecture Notes in Computer Science*, vol. 6369, pp. 1–10. Springer (2010)
18. Spring, N., Mahajan, R., Wetherall, D.: Measuring ISP topologies with rocketfuel. In: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications, SIGCOMM '02, pp. 133–145. ACM, New York, NY, USA (2002)
19. Szeto, W., Iraqi, Y., Boutaba, R.: A multi-commodity flow based approach to virtual network resource allocation. In: Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE, vol. 6, pp. 3004 – 3008 vol.6 (2003)
20. Yeow, W.L., Westphal, C., Kozat, U.: Designing and embedding reliable virtual infrastructures. In: Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures, VISA '10, pp. 33–40. ACM, New York, NY, USA (2010)
21. Zegura, E., Calvert, K., Bhattacharjee, S.: How to model an internetwork. In: INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE, vol. 2, pp. 594 –602 vol.2 (1996)
22. Zhu, Y., Ammar, M.: Algorithms for assigning substrate network resources to virtual network components. In: INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, pp. 1 –12 (2006)