Technische Universität Berlin
Fakultät für Elektrotechnik und Informatik
Lehrstuhl für Intelligente Netze und Management Verteilter Systeme

Virtual Network Embeddings: Theoretical Foundations and Provably Good Algorithms

vorgelegt von

Matthias Johannes Rost, M.Sc.

von der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
– Dr. rer. nat. –

genehmigte Dissertation

Promotionsausschuss

Vorsitzender:	Prof. Dr. Rolf Niedermeier
Gutachter:	Prof. Dr. Thomas Zinner
Gutachter:	Prof. Dr. Stefan Schmid
Gutachterin:	Prof. Anja Feldmann, Ph.D.
Gutachter:	Prof. Dr. Harald Räcke

Tag der wissenschaftlichen Aussprache: 25. März 2019

Berlin 2019

Abstract

Virtualization and resource isolation techniques have enabled the efficient sharing of networked resources both inside and across data centers. To guarantee reliable performance for all tenants, while efficiently sharing the available resources, novel resource allocation problems have arisen. Specifically, the core problem, coined the *Virtual Network Embedding Problem* (VNEP), asks for the embedding of several *virtualized networks* to the physical network while not exceeding resource capacities. The VNEP has been studied extensively for more than a decade, yet, few theoretic results pertaining to the VNEP are known.

To facilitate a better understanding of the VNEP as well as to derive novel *efficient* algorithms for the VNEP, this thesis studies the theoretical underpinnings of the VNEP. In particular, the computational complexity of the VNEP is studied and the $\mathcal{NP}$-completeness of the VNEP under various restrictions is proven. Furthermore, the first (fixed-parameter) tractable approximations are obtained for the offline setting. While theoretic in nature, we bridge the gap between theory and practice, and, based on our theoretic insights, derive novel algorithms for the VNEP and show their practical applicability in extensive computational evaluations.

Furthermore, we improve current state-of-the-art solutions and propose novel mechanisms to more efficiently use and share resources. Considering the virtual cluster abstraction for data centers, we give an optimal polynomial-time algorithm and develop a novel model to more efficiently use the scarce bandwidth resources. Additionally, we propose a novel continuous-time approach to *schedule* virtual networks under temporal flexibilities and validate the approach using computational studies.

Zusammenfassung

Virtualisierung sowie Techniken zur Isolierung von Ressourcen haben das effiziente Teilen von vernetzten Ressourcen sowohl innerhalb wie auch zwischen Rechenzentren ermöglicht. Um eine verlässliche Qualität in der Diensterbringung für alle Beteiligten zu garantieren, während die Ressourcen effizient geteilt werden, wurden neue Allokationsprobleme für Ressourcen von Nöten. Konkret verlangt das als Virtual Network Embedding Problem bekannte Kernproblem die Einbettung mehrerer virtualisierte Netzwerke auf einem einzelnen physikalischen Netz ohne dabei Ressourcenkapazitäten zu überschreiten. Dieses Problem wurde im vergangenen Jahrzehnt intensiv studiert, es sind jedoch nur wenige theoretische Resultate bekannt.

Um ein besseres Verständnis des VNEPs zu ermöglichen sowie um neue effiziente Algorithmen für das VNEP zu entwickeln, werden in dieser Dissertation die theoretischen Grundlagen des VNEPs studiert. Insbesondere wird die Komplexität des VNEPs betrachtet und die $\mathcal{NP}$-Vollständigkeit unter verschiedensten Restriktionen gezeigt. Weiterhin werden die ersten (parametrisierten) effizienten offline Approximationen gefunden. Während diese Resultate zunächst theoretischer Natur sind, verbinden wir Theorie und Praxis, und leiten, ausgehend von den theoretischen Einsichten, neuartige Algorithmen für das VNEP ab und zeigen deren Anwendbarkeit in umfangreichen Simulationsstudien.

Weiterhin verbessern wir bekannte Lösungen und schlagen neue Mechanismen vor, um Ressourcen effizienter zu nutzen und zu teilen. Spezifisch für die Abstraktion der Virtual Cluster, welche in Rechenzentren benutzt wird, geben wir den ersten optimalen polynomiellen Algorithmus und zeigen eine neuartige Methode auf, um die knappen Bandbreitenressourcen effektiver zu nutzen. Zusätzlich schlagen wir ein kontinuierliches Zeit-Modell für die Einbettung virtueller Netzwerke unter Berücksichtigung temporaler Flexibilitäten vor und validieren diesen Ansatz mittels Simulationsstudien.

Acknowledgements

Obtaining the Ph.D. has been a strenuous journey, but also one filled with memories I am very fond of. I would like to thank all the people who accompanied me on this journey and whom I had the pleasure to work with, even though I may not have included them in the following.

First of all, I thank Anja Feldmann for including me in her group and giving me the freedom to realize my research agenda. While we have worked together mostly in the realm of teaching, I have always deeply valued her input and the mutual respect when 'fighting' for ideas.

I am most thankful to Stefan Schmid, who took me on as a master student and shaped my research like no other. I have learnt so much from working with him and am still amazed at how productive and quarrel-free our collaboration was throughout the last 7 years. Without his relentless quest to push ideas, I could not have realized my full potential. Thank you!

I am also deeply grateful to Thomas Zinner, who not only filled the lingering void when Anja left, but brought a new light to the group in a way I did not deem possible at that point in time. I will never forget his continuous support during the last months of finishing my thesis.

I am very grateful to Harald Räcke for providing very detailed feedback on my thesis and his idea to again consider Chernoff bounds, which has helped me to improve this thesis significantly.

I thank Georgios Smaragdakis for his numerous advices over the last years and for providing me with the freedom to complete my thesis and other projects on my own terms.

Besides the above mentioned, I would furthermore like to thank all other professors, whose lectures and engagement with me motivated me to pursue a Ph.D. In particular, I want to extend these thanks to Petr Kuznetsov, Uwe Nestmann, Stephan Kreutzer, and Andreas Bley.

INET has truly been more than a workplace. Not only because I spent many nights, but because of the companionship. With Srivatsan I not only shared numerous breaks and late-nights discussions about god and the world, but found somebody as driven by research as I was. While Arne and Carlo did not teach me how to win SET, they did an excellent job in passing the teaching torch to me and I have missed them ever since they left. Nadi showed me how to handle EU projects just right while also having lots of fun, especially in Athens. With Lalith I not only shared some of the darkest jokes and the same taste in music, but also admired his discipline and his directness. Julius showed me that working hard can indeed go hand in hand with still enjoying life. Bala always had an open ear and his aspirations to make an impact in the research community inspired me. While Enric and Philipp at times pretended to dislike 'this place', they truly made the place a more enjoyable and better one. The same goes without question for Thomas, Niklas, and Susanna: thank you for the good times!

I also have to thank the student workers, with whom I had the pleasure to work with over the last couple of years: Elias Döhne, Alexander Elvers, Tom Koch, and Johannes Wortmann. Without their help, I could have never realized all the different projects I was involved in.

Last, but not least, my deepest gratitude goes to my wife and family for always being there and supporting me even during the most stressful times!

Publications

Pre-published Papers

In the following a list of all works co-authored during my time as a PhD student is given. Papers containing results presented in this thesis are highlighted using the • symbol. For all other works the ◦ symbol is used and these works are treated as common related work. Except for the technical reports, all papers have been peer-reviewed. All my collaborators are among my co-authors.

International Conferences

◦ M. Liu, A. W. Richa, M. Rost, and S. Schmid. "A Constant Approximation for Maximum Throughput Multicommodity Routing And Its Application to Delay-Tolerant Network Scheduling". In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. Apr. 2019, pp. 46–54. DOI: 10.1109/INFOCOM.2019.8737402

• M. Rost and S. Schmid. "Charting the Complexity Landscape of Virtual Network Embeddings". In: *2018 IFIP Networking Conference (IFIP Networking)*. May 2018, pp. 1–9. DOI: 10.23919/IFIPNetworking.2018.8696604

• M. Rost and S. Schmid. "Virtual Network Embedding Approximations: Leveraging Randomized Rounding". In: *2018 IFIP Networking Conference (IFIP Networking)*. May 2018, pp. 1–9. DOI: 10.23919/IFIPNetworking.2018.8696623

◦ G. Even, M. Rost, and S. Schmid. "An Approximation Algorithm for Path Computation and Function Placement in SDNs". In: *Structural Information and Communication Complexity*. Ed. by J. Suomela. Cham: Springer International Publishing, 2016, pp. 374–390. ISBN: 978-3-319-48314-6. DOI: 10.1007/978-3-319-48314-6_24

◦ A. Ludwig, S. Dudycz, M. Rost, and S. Schmid. "Transiently Secure Network Updates". In: *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*. SIGMETRICS '16. Antibes Juan-les-Pins, France: ACM, 2016, pp. 273–284. ISBN: 978-1-4503-4266-7. DOI: 10.1145/2896377.2901476

◦ V. Kotronis, R. Klöti, M. Rost, P. Georgopoulos, B. Ager, S. Schmid, and X. Dimitropoulos. "Stitching Inter-Domain Paths over IXPs". In: *Proceedings of the Symposium on SDN Research*. SOSR '16. Santa Clara, CA, USA: ACM, 2016, 17:1–17:12. ISBN: 978-1-4503-4211-7. DOI: 10.1145/2890955.2890960

• M. Rost, S. Schmid, and A. Feldmann. "It's About Time: On Optimal Virtual Network Embeddings under Temporal Flexibilities". In: *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. May 2014, pp. 17–26. DOI: 10.1109/IPDPS.2014.14

Peer-reviewed Journals

• M. Rost and S. Schmid. "Virtual Network Embedding Approximations: Leveraging Randomized Rounding". In: *IEEE/ACM Transactions on Networking (to appear)* (2019)

• M. Rost, E. Döhne, and S. Schmid. "Parametrized Complexity of Virtual Network Embeddings: Dynamic & Linear Programming Approximations". In: *SIGCOMM Comput. Commun. Rev.* 49.1 (Feb. 2019), pp. 3–10. ISSN: 0146-4833. DOI: 10.1145/3314212.3314214

- A. Ludwig, S. Dudycz, M. Rost, and S. Schmid. "Transiently Policy-Compliant Network Updates". In: *IEEE/ACM Transactions on Networking* 26.6 (Dec. 2018), pp. 2569–2582. ISSN: 1063-6692. DOI: 10.1109/TNET.2018.2871023
- T. Lukovszki, M. Rost, and S. Schmid. "Approximate and incremental network function placement". In: *Journal of Parallel and Distributed Computing* 120 (2018), pp. 159–169. ISSN: 0743-7315. DOI: https://doi.org/10.1016/j.jpdc.2018.06.006
- T. Lukovszki, M. Rost, and S. Schmid. "It's a Match!: Near-Optimal and Incremental Middlebox Deployment". In: *SIGCOMM Computer Communication Review (CCR)* 46.1 (Jan. 2016), pp. 30–36. ISSN: 0146-4833. DOI: 10.1145/2875951.2875956

- M. Rost, C. Fuerst, and S. Schmid. "Beyond the Stars: Revisiting Virtual Cluster Embeddings". In: *SIGCOMM Computer Communication Review (CCR)* 45.3 (July 2015), pp. 12–18. ISSN: 0146-4833. DOI: 10.1145/2805789.2805792

- S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet, and P. Demeester. "Network service chaining with optimized network function embedding supporting service decompositions". In: *Computer Networks* 93 (2015). Cloud Networking and Communications II, pp. 492–505. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2015.09.035

Workshops and Poster Sessions

- B. Németh, M. Szalay, J. Dóka, M. Rost, S. Schmid, L. Toka, and B. Sonkoly. "Fast and efficient network service embedding method with adaptive offloading to the edge". In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. Apr. 2018, pp. 178–183. DOI: 10.1109/INFCOMW.2018.8406882
- B. Németh, B. Sonkoly, M. Rost, and S. Schmid. "Efficient service graph embedding: A practical approach". In: *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. Nov. 2016, pp. 19–25. DOI: 10.1109/NFV-SDN.2016.7919470
- V. Kotronis, R. Klöti, M. Rost, P. Georgopoulos, B. Ager, S. Schmid, and X. Dimitropoulos. "Investigating the Potential of the Inter-IXP Multigraph for the Provisioning of Guaranteed End-to-End Services". In: *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. SIGMETRICS '15. Portland, Oregon, USA: ACM, 2015, pp. 429–430. ISBN: 978-1-4503-3486-0. DOI: 10.1145/2745844.2745877
- A. Ludwig, M. Rost, D. Foucard, and S. Schmid. "Good Network Updates for Bad Packets: Waypoint Enforcement Beyond Destination-Based Routing Policies". In: *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*. HotNets-XIII. Los Angeles, CA, USA: ACM, 2014, 15:1–15:7. ISBN: 978-1-4503-3256-9. DOI: 10.1145/2670518.2673873
- P. Sköldström, B. Sonkoly, A. Gulyás, F. Németh, M. Kind, F.-J. Westphal, W. John, J. Garay, E. Jacob, D. Jocha, J. Elek, R. Szabó, W. Tavernier, G. Agapiou, A. Manzalini, M. Rost, N. Sarrar, and S. Schmid. "Towards Unified Programmability of Cloud and Carrier Infrastructure". In: *2014 Third European Workshop on Software Defined Networks*. Sept. 2014, pp. 55–60. DOI: 10.1109/EWSDN.2014.18

Technical Reports

- M. Rost and S. Schmid. "NP-Completeness and Inapproximability of the Virtual Network Embedding Problem and Its Variants". In: *CoRR* abs/1801.03162 (2018). URL: http://arxiv.org/abs/1801.03162 (visited on Sept. 19, 2019)

- M. Rost and S. Schmid. "Virtual Network Embedding Approximations: Leveraging Randomized Rounding". In: *CoRR* abs/1803.03622 (2018). URL: `http://arxiv.org/abs/1803.03622` (visited on Sept. 19, 2019)
- ◦ T. Lukovszki, M. Rost, and S. Schmid. "Approximate and Incremental Network Function Placement". In: *CoRR* abs/1706.06496 (2017). URL: `http://arxiv.org/abs/1706.06496` (visited on Sept. 19, 2019)
- ◦ V. Kotronis, R. Klöti, M. Rost, P. Georgopoulos, B. Ager, S. Schmid, and X. A. Dimitropoulos. "Investigating the Potential of the Inter-IXP Multigraph for the Provisioning of Guaranteed End-to-End Services". In: *CoRR* abs/1611.03407 (2016). URL: `http://arxiv.org/abs/1611.03407` (visited on Sept. 19, 2019)
- ◦ V. Kotronis, R. Klöti, M. Rost, P. Georgopoulos, B. Ager, S. Schmid, and X. A. Dimitropoulos. "Stitching Inter-Domain Paths over IXPs". In: *CoRR* abs/1611.02642 (2016). URL: `http://arxiv.org/abs/1611.02642` (visited on Sept. 19, 2019)
- ◦ G. Even, M. Rost, and S. Schmid. "An Approximation Algorithm for Path Computation and Function Placement in SDNs". In: *CoRR* abs/1603.09158 (2016). URL: `http://arxiv.org/abs/1603.09158` (visited on Sept. 19, 2019)
- M. Rost and S. Schmid. "Service Chain and Virtual Network Embeddings: Approximations using Randomized Rounding". In: *CoRR* abs/1604.02180 (2016). URL: `http://arxiv.org/abs/1604.02180` (visited on Sept. 19, 2019)
- ◦ V. Kotronis, R. Klöti, M. Rost, P. Georgopoulos, B. Ager, S. Schmid, and X. Dimitropoulos. *Investigating the Potential of the Inter-IXP Multigraph for the Provisioning of Guaranteed End-to-End Services*. Tech. rep. 360. ETH Zurich, Laboratory TIK, Feb. 2015. URL: `ftp://ftp.tik.ee.ethz.ch/pub/publications/TIK-Report-360.pdf` (visited on Sept. 19, 2019)

Under submission

Parts of this thesis are based on the following paper that is currently under submission.

Peer-reviewed Journals

- M. Rost and S. Schmid. "On the Hardness and Inapproximability of Virtual Network Embeddings". In: *IEEE/ACM Transactions on Networking (under submission)* (2019)

Contents

"Wir müssen wissen. Wir werden wissen."
– David Hilbert

1 Introduction

The advent of the Internet has profoundly impacted society as a whole. Today, we access information faster than ever before, watch movies instantaneously on our mobile devices, and stay in contact with far away friends via video calls. The success of the Internet and the ongoing societal transformations have been facilitated by several advances in the areas of computing and communication technologies. In particular, advances in *virtualization* technologies have been the main driver of innovation behind *cloud computing*: the (remote) offering of virtualized compute resources in data centers for customers, individuals and companies alike. The ever increasing reliance on such remote services necessitates the *reliable* operation and the safeguarding of quality guarantees for the rendered services. With the ever increasing growth of application traffic, sufficient bandwidth is required for optimal performance. Accordingly, means to provide *network virtualization* have attracted much attention over the last decade and lead – among others – to many realizations, most prominently in the form of the Software-Defined Networking (SDN) paradigm. Given the capability to flexibly allocate virtualized *compute and networking* resources, several algorithmic resource allocation challenges arose. Many of these resource allocation problems can be posed as a graph-theoretic Virtual Network Embedding Problem (VNEP). In the following, we first informally introduce the problem and then discuss its many applications.

A Glance at the Virtual Network Embedding Problem. The physical infrastructure is modeled as a network: nodes offer compute resources and the edges represent communication channels of limited bandwidth. Customers can request virtual networks that are analogously given as graphs, which are attributed with compute and bandwidth demands on the nodes and edges, respectively. The task of the infrastructure provider is to allocate resources appropriately within the physical network to *embed* the virtual network. Figure 1.1 depicts an example: a virtual network consisting of 4 *virtual* nodes **A**, **B**, **C**, **D** is embedded on a 5-node physical network, such that each virtual node is mapped to a physical node and adjacent virtual nodes are connected in the physical network via paths which are colored differently

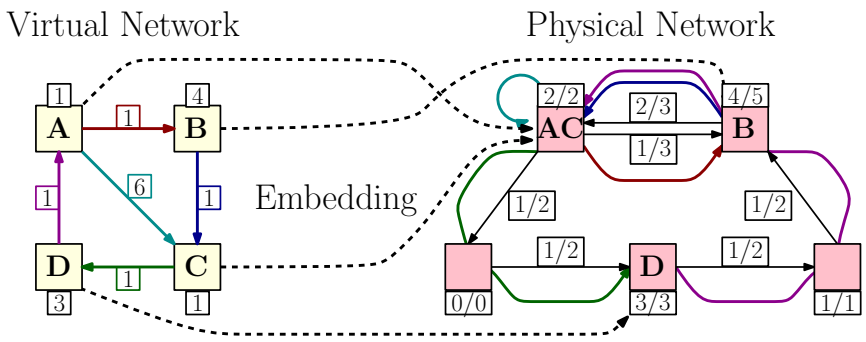
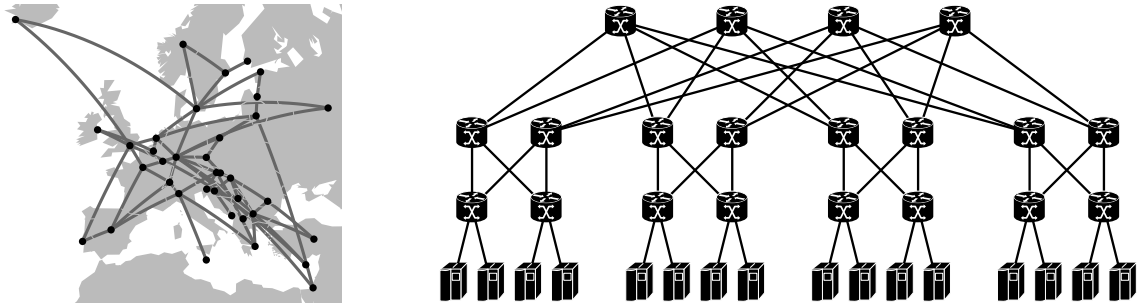


Figure 1.1: Exemplary embedding of a virtual network on a physical network. The numeric labels of the virtual network elements denote the resource demands, while for the substrate network the resource usage and the total capacity are given.



(a) GÉANT wide-area research network [Kni+11]

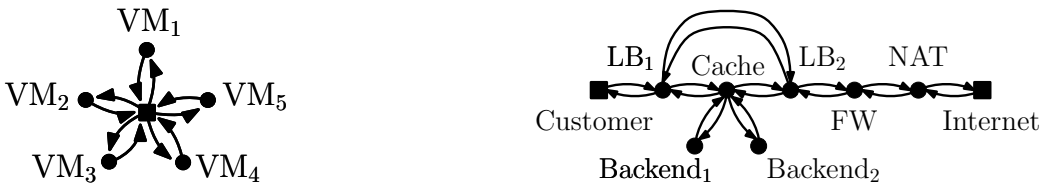
(b) 4-port fat-tree data center topology [ALV08]

Figure 1.2: Exemplary physical networks.

in Figure 1.1. Importantly, the embedding shown in the example is feasible, i.e., the physical network's capacities are not exceeded, as this would otherwise result in performance degradation. Besides enforcing feasibility of the embedding, additional constraints can be imposed, e.g., restricting the set of substrate nodes and substrate edges to which a specific virtual node or virtual edge can be mapped.

Origins of the VNEP and its Applications over Time. The Virtual Network Embedding Problem (VNEP) has received tremendous attention over the course of the last 15 years [Fis+13]. It originated first in the early 2000s in the context of provisioning value-added services for, among others, video conferencing [Cha+01; SIB03]. Additional interest in the VNEP arose in the networking community when needing to set up testbeds [RAL03]. Using such testbeds indeed was deemed to be of crucial importance to easily innovate the Internet using so called overlays [And+05]. By 2009, the VNEP was formulated as graph-theoretic optimization problem and several algorithms tackling it were published [Yu+08; CRB09] and by 2013 already more than 80 different algorithms for various flavors of the VNEP had been conceived [Fis+13].

While – by virtue of its use cases – algorithms for the VNEP were developed for applications in Wide-Area Networks (WANs, see Figure 1.2a for an example), the first specific virtual network abstraction for data centers was proposed in the form of Virtual (Oversubscribed) Clusters [Bal+11]: a virtual cluster topology is a simple star network with a logical switch connecting a set of VMs (see Figure 1.3a), while an oversubscribed cluster is a tree of depth 2 with VMs as leaves. Virtual clusters were specifically conceived for providing resource isolation for bandwidth-intensive MapReduce-like tasks and studied on specific data center topologies, namely VL2 [Gre+09] and fat-trees [ALV08] (see Figure 1.2b). While the underlying graph embedding problem did not change, the choice of hierarchical data center topologies lead to the design of specific dynamic programming algorithms to solve the embedding problem [Bal+11]. Research



(a) A virtual cluster with 5 Virtual Machines (VMs) connected to a single logical switch in the center [Bal+11].

(b) A Service Chain as envisioned in the context of mobile internet service providers [Nap+16]. The customer's traffic may be routed through a cache, which is connected to several backends, while all traffic is routed through a firewall and a network address translator.

Figure 1.3: Exemplary virtual network requests.

into the virtual cluster embedding problem is still ongoing, e.g., regarding temporal aspects of embeddings or heterogeneous resource demands [Xie+12; Dai+15; YLH17]. Beside the orchestration of virtual clusters, the VNEP was also studied in the form of the Virtual Data Center Embedding Problem [Amo+13; Rab+13; Zha+14]. Here, the focus lies on embedding virtualized data center requests, which span several data centers.

The most recent incarnation of the VNEP is based on the advent of Network Functions Virtualization (NFV) [Chi+12], which envisions the virtualization of network functions, which have been classically implemented via specific hardware appliances (middleboxes). In fact, in [She+12] it was found that the number of middleboxes comes close to the number of classic networking components like routers. The NFV approach proposes the replacement of these hardware appliances by virtualized network functions that can be hosted on commodity servers, thereby reducing capital expenditures. Additionally, the Internet Service Providers first proposing NFV also aimed for a distinct reduction in operational expenditures, due to simpler management [Chi+12]. Besides technological innovation to allow for NFV in conjunction with for example Software-Defined Networking [JP13; Soa+15; Mij+16], the virtual network abstraction of *service chains* emerged [QN15; Bha+16]. While service chains originally represented only a concatenation of functions (hence: chain) [MKK14; Sou+14], current use cases also include more general request topologies, which may contain cycles or are bidirected [HP15]. The notion of service chains and the principle of *network slicing* to provide Quality-of-Service guarantees are currently also specifically discussed in the context of mobile operators [Nap+16] and 5G [Beg+17; Fou+17]. Figure 1.3b depicts an example of such a use case.

Lastly, we note that NFV also has raised algorithmic challenges regarding the placement of (virtualized) network functions. Accordingly, the task of finding the best locations to place network functions, such that several service chains can be routed through these, has attracted significant attention over the course of the last years [MT14; Bou+15; Coh+15; LRS16].

Objectives and Restrictions. Due to the many applications of the VNEP, several objectives and mapping restrictions have been studied. In the following, a short overview is given.

The VNEP has been studied both in the online and offline setting. In the online setting the embedding of a single request is considered. Here, the objective is to find an embedding minimizing the resource allocation cost [CRB09; Fis+13] or minimizing the maximal load [CRB12; MKK14]. In the offline setting the embedding of several requests is considered and the most studied objective is to maximize the provider's profit by exerting admission control [CRB12; Eve+13], i.e., selecting a subset of requests to embed.

Besides enforcing that capacities are not exceeded, additional restrictions have emerged in various applications:

- Restrictions on the placement of virtual nodes first arose to enforce the closeness to locations of interest [CRB09], but were also used in the context of privacy policies to restrict mappings to certain countries [SSF12]. However, these restrictions are now also used in the context of Network Functions Virtualization and Service Chaining, as specific softwarized functions implemented as a virtual machine may be mapped only to commodity servers, while hardware firewall functions can only be mapped on respective hardware appliances [MKK14; HP15].
- Routing restrictions first arose in the context of expressing security policies, as for example some traffic may not be routed via insecure domains or physical links shall not be shared with specific other virtual networks [Bay+13; Fis+13].
- Restrictions on latencies were studied for the VNEP in [IR11] and have been recently studied intensively in the context of Service Function Chaining to achieve responsiveness and Quality-of-Service [MKK14; HP15].

Algorithmic Approaches. Solving the VNEP is the main resource allocation challenge in virtualized networks and is algorithmically challenging, as it is known to be $\mathcal{NP}$-hard [Ama+16]. Thus, for solving the VNEP mainly heuristic and exact, i.e., non polynomial-time, algorithms were proposed [Fis+13]. In particular, several dozens of algorithms were proposed to solve the VNEP and its siblings, including the Virtual Cluster Embedding Problem [Bal+11] and the Service Function Chain Embedding Problem [HB16]. Most approaches to solve the VNEP either rely on heuristics [CRB09] or metaheuristics [Fis+13]. Mixed Integer Programming is the most widely used exact approach to tackle the VNEP [IR11; Fis+13; MKK14].

Importantly, efficient approximation algorithms for the VNEP have not been studied. While hopes were expressed to obtain approximations using Linear Programming by Chowdhury et al. in 2009 [CRB09] and the survey [CB09] explicitly mentions approximations as desirable, no general approximation results are known. In 2013 and 2016, respectively, the surveys [Fis+13; Mij+16] call only for the development of heuristics, as the VNEP is acknowledged to be *computationally intractable*.

1.1 Problem Statement

The concept of (network) virtualization allows the shared usage of physical resources. A cornerstone of this paradigm is the efficient usage of the limited physical resources. This thesis provides algorithms to improve existing solutions either by providing *quality guarantees* (compared to heuristics) or to *provide solutions quicker* (compared to exact approaches).

Specifically, this thesis provides a theoretically well-founded basis for the development of *provably good algorithms*, i.e., approximations, for the VNEP. Such approximation algorithms provide guarantees both for the runtime as well as for *the solution quality* and may therefore bridge the gap between heuristic and exact algorithms. Specifically, compared to purely heuristical approaches, approximations may increase the solution quality or may facilitate finding valid solutions in the first place. Compared to exact algorithms, the bounded runtime may be beneficial to obtain *good* solutions more quickly or to, e.g., more quickly decide that no solution can exist. Besides the application of approximation algorithms to solve the VNEP, a deeper understanding of the challenges associated with obtaining approximations may eventually also shed light on how to obtain *better heuristics* or how to improve *exact algorithms* for the VNEP. Therefore, the study of approximations bears the potential to, e.g., improve the substrate provider's profit, reduce the costs of customers and improve their satisfaction by providing better solutions. Notably, this is independent of the approximation's applicability in practice. While this thesis focuses on provably good algorithms, the *efficient implementation* and *computational evaluation* of the respective algorithms is treated of equal importance and is used to identify the benefits and limitations of their applicability in practice.

Besides extensively studying the theoretical foundations of the VNEP, approaches to improve state-of-the-art solutions are also considered. Specifically, means to improve the resource utilization of virtual cluster embeddings in data centers and means to harness temporal flexibilities in scheduling virtual network requests are studied.

1.2 Contributions of this Thesis

This thesis provides a new theoretic framework for the VNEP, obtaining, among others, the first approximation algorithms:

- The computational complexity of the VNEP is thoroughly studied and a series of novel hardness results is obtained. In particular, the VNEP is shown to be $\mathcal{NP}$-complete and therefore inapproximable *under any objective*, unless $\mathcal{P}=\mathcal{NP}$ holds, and our results also hold when only approximate embeddings are studied. Even more, studying various restriction combinations, the $\mathcal{NP}$-completeness is proven even in

the absence of capacity constraints and all presented hardness results also pertain when request graphs are restricted to planar graphs.

- Given these $\mathcal{NP}$-hardness results, novel algorithmic relaxations of the VNEP are studied. On the one hand, the *Valid Mapping Problem* (VMP) is introduced, which relaxes the requirement that embeddings must obey the substrate's resource capacities. Based on the notion of valid mappings, the Fractional VNEP is introduced, which allows to return a convex combination of mappings instead of just a single one. While our hardness results pertain to the VMP and the Fractional VNEP as well, we constructively show that these problems become *fixed-parameter* tractable when taking the request graphs' structure into account. In particular, the VMP and the Fractional VNEP can be solved in polynomial-time for graphs of bounded treewidth. For all other graphs, the runtime is exponential in the treewidth of the request(s).

- We show that the ability to solve the Fractional VNEP directly translates to *the first* (parametrized) approximation algorithms for the offline VNEP for arbitrary request graphs. In particular, the considered approximations consider the setting in which *multiple* requests are given. We study two objectives and give approximations for both of them. Under the profit objective, admission control can be exerted to select a subset of requests maximizing the provider's revenue. Under the cost objective, all request must be embedded while minimizing resource costs. Based on these approximations, several heuristics are derived for the profit variant and their practical applicability and their potential to improve over existing heuristics is shown in extensive computational evaluations.

Furthermore, this thesis introduces solution approaches for two applications, the embedding of virtual clusters, and the temporal scheduling of virtual networks.

- Considering the specific virtual cluster request abstraction, which restricts request graphs to star networks with uniform demands, the optimal polynomial-time solvability of the respective online embedding problem is proven. Furthermore, a hose-based model adaptation is proposed, which can significantly reduce resource consumption while rendering the respective embedding problem hard again. A heuristic is derived and its performance is studied in an extensive computational evaluation to validate the approach.

- Turning towards temporal aspects of virtual network embeddings, we introduce and study the Temporal Virtual Network Embedding Problem (TVNEP), that asks not only to find embeddings for the requests but to also temporally *schedule* the respective requests. Interested in a qualitative analysis of the potential to harness temporal flexibilities, a series of Mixed-Integer Programs is proposed and analyzed. Given the ability to solve the TVNEP *exactly*, our computational evaluation shows that even minor temporal flexibilities may increase the provider's profit substantially.

1.3 Overview of this Thesis

At first, in Chapter 2 the VNEP variants studied in this thesis are formally introduced. In particular, besides introducing online and offline variants of the VNEP, a taxonomy of the various different VNEP restrictions is introduced.

In Chapter 3 the necessary algorithmic background for this thesis is given. Among others, the fundamentals of classic and parametrized computational complexity, Linear and Integer Programming, and (randomized) approximation algorithms are discussed, such that this thesis is mostly self-contained.

In Chapter 4 an overview on existing works on the VNEP and its related problems is given. Here, our presentation mainly focuses on known approximation and competitiveness results.

In Chapter 5, the computational complexity of the VNEP is studied and the $\mathcal{NP}$-completeness under several restriction combinations is shown. Furthermore, the hardness of computing approximate embeddings under request graph restrictions is proven.

Chapter 6 presents the first parametrized algorithms for solving the Fractional VNEP and the VMP. The algorithms are parametrized either by a novel graph number called extraction width or by the treewidth of the request graphs.

Chapter 7 constructively proves the existence of the first (parametrized) approximation algorithms for the offline VNEP, which generally come at the cost of exceeding resource capacities. Based on the approximation results several heuristics are derived and their practical applicability is evaluated using extensive computational evaluations.

In Chapter 8 algorithms for the embedding of virtual clusters as well as a new interpretation of the virtual cluster abstraction is proposed and evaluated.

In Chapter 9 the Temporal VNEP is introduced and the impact of harnessing temporal flexibilities is evaluated computationally.

This thesis is concluded in Chapter 10 by summarizing the results and discussing potential avenues for future research.

"All models are wrong, but some models are useful."
– George Edward Pelham Box

2 Formal Problem Statement

Within this section we formally introduce the various variants of the Virtual Network Embedding Problem (VNEP). Specifically, in Section 2.2 the input of the VNEP and some basic definitions relating to embeddings are given. In Section 2.3 several core problem formulations are introduced, while in Section 2.4 additional restrictions are discussed. Lastly, in Sections 2.5 and 2.6 the notion of approximate embeddings and a taxonomy of the VNEP variants are given.

2.1 General Notation

The following notation is used throughout this work. We use $[x]$ to denote the set $\{1, 2, \ldots, x\}$ for $x \in \mathbb{N}$. For a directed graph $G = (V, E)$, we denote by $\delta^+(u) = \{(u, v) \in E\}$ and $\delta^-(v) = \{(v, u) \in E\}$ the outgoing and incoming edges of node $u \in V$. Similarly, for an undirected graph $G = (V, E)$, $\delta(u) = \{\{u, v\} \in E\}$ denotes the edges incident to $u \in V$. We write $x \in G$ to denote any node or edge contained in G, i.e., $x \in V \cup E$. When considering functions on tuples, we often omit the parentheses of the tuple and simply write $f(a, b)$ instead of $f((a, b))$.

2.2 Input and Basic Definitions

Substrate Network. We refer to the physical network as substrate network and model it as a directed graph $G_S = (V_S, E_S)$. Capacities in the substrate are given by the function $d_S : G_S \to \mathbb{R}_{\geq 0} \cup \{\infty\}$. The capacity $d_S(u)$ of node $u \in V_S$ may represent for example the number of CPU cores while the capacity $d_S(u, v)$ of edge $(u, v) \in E_S$ represents the available bandwidth. By allowing to set substrate capacities to ∞, the capacity constraints on the respective substrate elements can be effectively disabled. Substrate elements $x \in G_S$ may be attributed with costs $c_S(x) \geq 0$ to indicate the *price* for using the resource x (per unit capacity). We denote by $\mathcal{P}_S$ the set of all simple paths in G_S.

Request Graphs. A request r is similarly modeled as a directed graph $G_r = (V_r, E_r)$ together with node and edge demands $d_r : G_r \to \mathbb{R}_{\geq 0}$. While at times only a single request is considered, a set of requests $\mathcal{R}$ may be given and each request $r \in \mathcal{R}$ may be attributed with a profit $b_r \geq 0$ that the provider obtains when embedding the request.

Valid Mappings and Feasible Embeddings. The general task is to find a *mapping* of a request graph G_r on the substrate network G_S, i.e., a mapping of request nodes to substrate nodes and a mapping of request edges to paths in the substrate. Virtual nodes and edges can only be mapped on substrate nodes and edges of sufficient capacity. Accordingly, we denote by $V_S^{r,i} = \{u \in V_S \mid d_S(u) \geq d_r(i)\}$ the set of substrate nodes supporting the mapping of node $i \in V_r$ and by $E_S^{r,i,j} = \{(u, v) \in E_S \mid d_S(u, v) \geq d_r(i, j)\}$ the substrate edges supporting the mapping of virtual edge $(i, j) \in E_r$.

Definition 2.1 (Basic Valid Mapping Definition)**.** A *valid* mapping of request r to the substrate G_S is a tuple $m_r = (m_V, m_E)$ of functions $m_V : V_r \to V_S$ and $m_E : E_r \to \mathcal{P}_S$, such that the following holds:

- The function m_V maps virtual to *suitable* substrate nodes: $m_V(i) \in V_S^{r,i}$ holds for $i \in V_r$.

- The function m_E maps each virtual edge $(i,j) \in E_r$ to a simple path in G_S connecting $m_V(i)$ to $m_V(j)$ while only using allowed edges, i.e., $m_E(i,j) \subseteq E_S^{r,i,j}$ holds for $(i,j) \in E_r$.

The set of all valid mappings of request r is denoted by $\mathcal{M}_r$. □

Considering the above definition, note the following. Firstly, the mapping $m_E(i,j)$ of the virtual edge $(i,j) \in E_r$ may be the empty path, if (and only if) i and j are mapped on the same substrate node. Secondly, the above definition of valid mapping only enforces that single resource allocations do not exceed the available capacity. To enforce that the cumulative allocations respect capacities, we introduce the following:

Definition 2.2 (Allocations)**.** We denote by $A(m_r, x) \in \mathbb{R}_{\geq 0}$ the resource allocations induced by valid mapping $m_r = (m_V, m_E) \in \mathcal{M}_r$ on resource $x \in G_S$. The following holds for node $u \in V_S$ and edge $(u,v) \in E_S$, respectively:

$$A(m_r, u) = \sum_{i \in V_r : m_V(i) = u} d_r(i)$$
$$A(m_r, u, v) = \sum_{(i,j) \in E_r : (u,v) \in m_E(i,j)} d_r(i,j)$$ □

We refer to one or several mappings as *feasible*, if the (cumulative) allocations do not exceed any resources' capacity:

Definition 2.3 (Feasible Embedding)**.** A set of mappings $\{m_r\}_{r \in \mathcal{R}}$ over a set of requests $\mathcal{R}$ is feasible, if and only if $\sum_{r \in \mathcal{R}} A(m_r, x) \leq d_S(x)$ holds for all resources $x \in G_S$. A single mapping m_r is feasible, if this holds for the singleton request set $\{m_r\}$. □

We introduce the notions of maximal demands and maximal allocations as these will help useful for bounding approximation guarantees later on.

Definition 2.4 (Maximal Demand, Maximal Allocations, Maximal Demand-to-Capacity Ratio)**.** Considering a request r, we denote by $d_{\max}(r,x)$ the maximal demand that a virtual element of request r may impose on substrate resource $x \in G_S$ and by $d_{\mathsf{max}}(x)$ the maximal demand on this resource over all requests. Furthermore, we denote by $A_{\max}(r,x)$ the maximal (cumulative) allocations that any *valid* mapping may induce on resource $x \in G_S$ and by $A_{\mathsf{max}}(x)$ the maximum of these allocations over all requests. Accordingly, the following holds:

$$d_{\max}(r,u) = \max\left(\{0\} \cup \{d_r(i) \mid i \in V_r : u \in V_S^{r,i}\}\right) \tag{2.1}$$
$$d_{\max}(r,u,v) = \max\left(\{0\} \cup \{d_r(i,j) \mid (i,j) \in E_r : (u,v) \in E_S^{r,i,j}\}\right) \tag{2.2}$$
$$d_{\mathsf{max}}(x) = \max_{r \in \mathcal{R}} d_{\max}(r,x) \tag{2.3}$$
$$A_{\max}(r,x) = \max\{A(m_r,x) \mid m_r \in \mathcal{M}_r\} \tag{2.4}$$
$$A_{\mathsf{max}}(x) = \max_{r \in \mathcal{R}} A_{\mathsf{max}}(r,x) \tag{2.5}$$

Furthermore, we introduce the maximal demand to capacity ratios with respect to nodes, edges, and all resources as follows:

$$\delta_V = \max_{u \in V_S} d_{\mathsf{max}}(u)/d_S(u) \tag{2.6}$$
$$\delta_E = \max_{(u,v) \in V_S} d_{\mathsf{max}}(u,v)/d_S(u,v) \tag{2.7}$$
$$\delta_{\max} = \max\{\delta_V, \delta_E\} \tag{2.8}$$

Notably, as virtual nodes and edges can only be mapped on respective substrate resources of sufficient capacity (cf. Definition 2.1), δ_V, δ_E and $\delta_{\max}$ are upper bounded by 1. □

2.3 Core Problem Definitions

Within this thesis, we mainly study the following offline optimization variants of the VNEP.

Definition 2.5 (Offline Virtual Network Embedding Problem, Profit Variant)**.** Given a set of requests $\mathcal{R}$, the task is to find a feasible embedding $\{m_r\}_{r\in\mathcal{R}'}$ of a subset of requests $\mathcal{R}' \subseteq \mathcal{R}$ maximizing the profit $\sum_{r\in\mathcal{R}'} b_r$. □

Definition 2.6 (Offline Virtual Network Embedding Problem, Cost Variant)**.** Given a set of requests $\mathcal{R}$, the task is to find a feasible embedding $\{m_r\}_{r\in\mathcal{R}}$ of all requests minimizing the overall embedding cost $\sum_{r\in\mathcal{R}} c_S(m_r) = \sum_{x\in G_S} c_S(x) \cdot \sum_{r\in\mathcal{R}} A(m_r, x)$. □

To obtain approximations for the offline VNEP, the following fractional VNEP variant will be of importance:

Definition 2.7 (Fractional Offline Virtual Network Embedding Problem, Profit and Cost)**.**
The Fractional Offline VNEP has the same input as the offline VNEP but allows for returning convex combinations of valid mappings (for each request) as a solution. Concretely, the Fractional VNEP asks for finding a set of convex combination of valid mappings $\mathcal{D}_r = \{(f_r^k, m_r^k) \in \mathbb{R}_{\geq 0} \times \mathcal{M}_r\}$ for each request $r \in \mathcal{R}$, such that:

(1) the sum of weights is bounded by 1, i.e., $\sum_{(f_r^k, m_r^k)\in\mathcal{D}_r} f_r^k \leq 1$ holds for request $r \in \mathcal{R}$, and

(2) the cumulative (fractional) allocations do not exceed the resource capacities, i.e., $\sum_{r\in\mathcal{R}} \sum_{(f_r^k, m_r^k)\in\mathcal{D}_r} f_r^k \cdot A(m_r^k, x) \leq d_S(x)$ holds for each resource $x \in G_S$.

Again, we consider the profit and the cost variants:

Profit: The task is to maximize the overall profit $\sum_{r\in\mathcal{R}} \sum_{(f_r^k, m_r^k)\in\mathcal{D}_r} f_r^k \cdot b_r$.

Cost: Requiring that $\sum_k f_r^k = 1$ holds for $r \in \mathcal{R}$, the task is to minimize the overall cost $\sum_{x\in G_S} c_S(x) \cdot \sum_{r\in\mathcal{R}} \sum_{(f_r^k, m_r^k)\in\mathcal{D}_r} f_r^k \cdot A(m_r^k, x)$. □

In the literature [Fis+13], the online variant considering only a single request is often considered and we introduce the following cost variant.

Definition 2.8 (Online Virtual Network Embedding Problem)**.** Given a single request r, the online VNEP asks for finding the *feasible* embedding m_r minimizing the embedding cost $c_S(m_r) = \sum_{x\in G_S} c_S(x) \cdot A(m_r, x)$. □

For computational complexity considerations, we also introduce the *decision* variant of the VNEP, asking whether there exists a feasible embedding.

Definition 2.9 (Virtual Network Embedding Problem, Decision Variant)**.** Given is a single request and the task is to decide whether a feasible embedding exists and to return one if one exists. □

We will study the following relaxation of the online VNEP asking to find the minimum cost valid mapping (which does not need to be feasible).

Definition 2.10 (Valid Mapping Problem (VMP))**.** Given a request r, the VMP asks for finding the valid mapping m_r minimizing the cost function $c_S(m_r) = \sum_{x\in G_S} c_S(x) \cdot A(m_r, x)$ or deciding that none exists. □

Importantly, we note that when request demands are small compared to the substrate capacities, specifically, if any valid mapping is also feasible, the online VNEP reduces to the VMP:

Observation 2.11. Given a request for which any valid mapping $m_r \in \mathcal{M}_r$ is *feasible*, i.e., $A_{\mathsf{max}}(r,x) \leq d_S(x)$ holds for all substrate resources $x \in G_S$, then the online VNEP reduces to the VMP: an optimal solution to the VMP yields also an optimal solutions to the VNEP.

In Chapter 9 the Temporal Virtual Network Embedding Problem (TVNEP) is studied, which extends the profit offline VNEP by incorporating scheduling aspects. Specifically, the TVNEP asks for (i) performing access control and (ii) additionally scheduling the respective requests under temporal flexibilities, such that the solution yields the highest profit while being feasible at any point in time.

Definition 2.12 (Temporal Virtual Network Embedding Problem)**.** Given is a time horizon $\mathcal{T} > 0$ and temporal characteristics $t_r^s \geq 0$, $t_r^e \leq \mathcal{T}$, and $t_r^d \in \mathbb{R}_{\geq 0}$ for each request $r \in \mathcal{R}$: t_r^s and t_r^e denotes the earliest and latest points in time at which r may be embedded, and t_r^d denotes the required embedding duration for the request. The task is to find valid mappings $\{m_r\}_{r\in\mathcal{R}'}$ for a subset of requests $\mathcal{R}' \subseteq \mathcal{R}$ together with points in time at which the request $r \in \mathcal{R}'$ starts $t_r^+ \in [0,\mathcal{T}]$ and ends $t_r^- \in [0,\mathcal{T}]$, such that:

(1) $t_r^- - t_r^+ = t_r^d$, $t_r^s \leq t_r^+$ and $t_r^- \leq t_r^e$ hold for $r \in \mathcal{R}'$, and

(2) resource allocations are feasible for all points in time $t \in [0,\mathcal{T}]$, i.e.,

$$d_S(x) \geq \sum_{\substack{r\in\mathcal{R}':\\ t\in(t_r^+,t_r^-)}} A(m_r,x)$$

holds for each resource $x \in G_S$, and

(3) the profit $\sum_{r\in\mathcal{R}'} b_r$ is maximized. □

In Chapter 8 the online embedding of Virtual Clusters is studied. Virtual Clusters are a specific abstraction that enforces that the (undirected) request graph is a star network, where the leaves and the edges have uniform node and edge capacity demands. Under this restriction and in the undirected graph model (cf. Section 2.4.1) this problem reduces to the online VNEP.

2.4 Variants of the VNEP

In the following, several adaptations and extensions of the VNEP are introduced. We introduce the VNEP on undirected graphs and with splittable edge mappings and then discuss additional common mapping restrictions, that restrict the validity of mappings further.

2.4.1 Undirected VNEP

Early work on the VNEP considered undirected graphs instead of directed graphs [Fis+13].

Definition 2.13 (VNEP on Undirected Graphs)**.** In the undirected VNEP both the substrate network G_S as well as the request graphs $\{G_r\}_{r\in\mathcal{R}}$ are *undirected* graphs. Accordingly, the definition of valid edge mappings (cf. Definition 2.1) is changed as follows. For a valid mapping $m_r = (m_V, m_E) \in \mathcal{M}_r$ and each undirected virtual request edge $\{i,j\} \in E_r$ the edge mapping $m_E(\{i,j\})$ returns an undirected path connecting $m_V(i)$ to $m_V(j)$. All other definitions remain unchanged and we may omit the curly brackets when referring to edge mappings and accordingly write $m_E(\{i,j\}) = m_E(i,j) = m_E(j,i)$. □

2.4.2 VNEP with Splittable Edge Mappings

Above, the unsplittable variant of the VNEP was introduced that maps each virtual edge on a *single* substrate path. Besides that, a splittable adaptation was conceived, such that fractions of the bandwidth can be transported using arbitrarily many paths [Yu+08; CRB12].

Definition 2.14 (VNEP with Splittable Edge Mappings)**.** For splittable edge mappings the notion of valid mappings is adapted as follows. For a valid mapping $m_r = (m_V, m_E) \in \mathcal{M}_r$ the edge mapping $m_E(i,j)$ returns a *convex combination* of paths connecting $m_V(i)$ to $m_V(j)$. Specifically, $m_E(i,j) \in \{(w^k_{i,j}, P^k_{i,j}) \mid w^k_{i,j} \geq 0, P^k_{i,j} \in \mathcal{P}_S\}$ must hold for each virtual edge $(i,j) \in E_r$, such that:

(1) the k-th path $P^k_{i,j}$ connects $m_V(i)$ to $m_V(j)$ for all $(\cdot, P^k_{i,j}) \in m_E(i,j)$, and

(2) the weights of the convex combination sum to 1, i.e., $\sum_{(w^k_{i,j}, P^k_{i,j}) \in m_E(i,j)} w^k_{i,j} = 1$ holds.

The bandwidth is distributed over the different paths according to their weights and the notion of edge mapping allocations is adapted as follows:

$$A(m_r, u, v) = \sum_{(i,j) \in E_r} \sum_{\substack{(w^k_{i,j}, P^k_{i,j}) \in m_E(i,j): \\ (u,v) \in P^k_{i,j}}} w^k_{i,j} \cdot d_r(i,j)\,. \qquad \square$$

Note that in the above definition directed substrate and request edges were assumed. However, splittable edge mappings can just as easily be incorporated into the undirected VNEP variant, i.e., when considering *undirected* virtual and substrate edges.

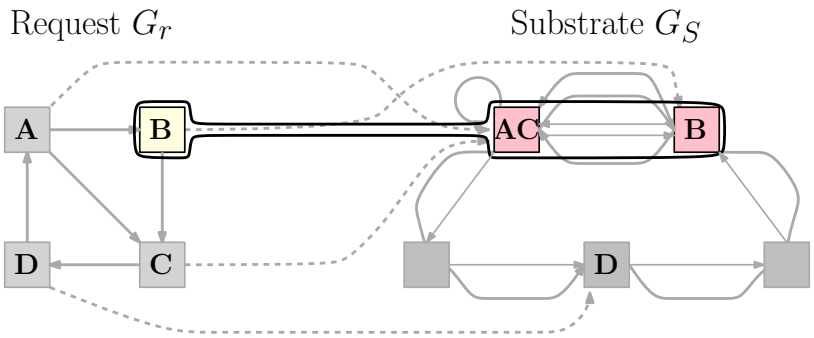
2.4.3 Additional Mapping Restrictions

As briefly discussed in Chapter 1, additional mapping requirements are enforced in many settings. Accordingly, we now formalize (i) node placement, (ii) edge routing, and (iii) latency restrictions. Node placement and edge routing restrictions effectively exclude potential mapping options for nodes and edges. Considering latency restrictions, we introduce latency bounds for each of the virtual edges. The following definitions all restrict the set of valid mappings and hence carry over to the respective optimization and decision variants. Figure 2.1 depicts examples for the respective restrictions.

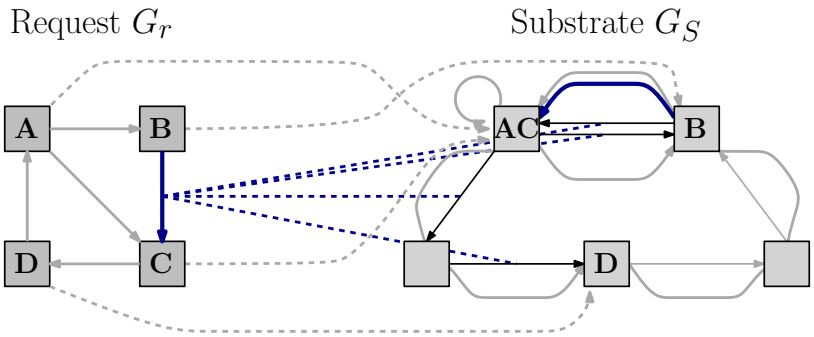
Definition 2.15 (Node Placement Restrictions)**.** For each virtual node $i \in V_r$ a set of forbidden substrate nodes $\overline{V}^{r,i}_S \subset V_S$ is provided. Accordingly, the set of allowed nodes $V^{r,i}_S$ is defined to be $\{u \in V_S \setminus \overline{V}^{r,i}_S \mid d_S(u) \geq d_r(i)\}$. $\square$

Definition 2.16 (Routing Restrictions)**.** For each virtual edge $(i,j) \in E_r$ a set of forbidden substrate edges $\overline{E}^{r,i,j}_S \subseteq E_S$ is provided. Accordingly, the set of allowed edges $E^{r,i,j}_S$ is set to be $\{(u,v) \in E_S \setminus \overline{E}^{r,i,j}_S \mid d_S(u,v) \geq d_r(i,j)\}$. $\square$

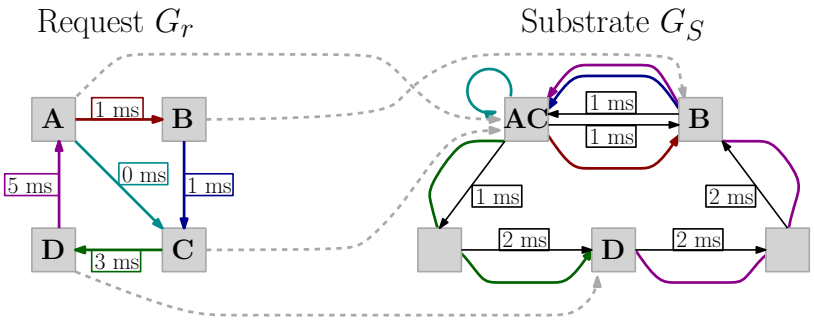
Definition 2.17 (Latency Restrictions)**.** For each substrate edge $e \in E_S$ the edge's latency is given via $l_S(e) \in \mathbb{R}_{\geq 0}$. Latency bounds for virtual edges are specified via the function $l_r : E_r \to \mathbb{R}_{\geq 0} \cup \{\infty\}$, and we will require that the latency along the substrate path $m_E(i,j)$, used to realize the edge $(i,j) \in E_r$, is less than $l_r(i,j)$. Formally, the Definition 2.1 of valid mappings is extended by including that $\sum_{e \in m_E(i,j)} l_S(e) \leq l_r(i,j)$ must hold for $(i,j) \in E_r$. $\square$



(a) Visualization of a node placement restriction: virtual node **B** may only be mapped on the top substrate nodes by excluding the other substrate nodes via the set $\overline{V}_S^{r,\mathbf{B}}$.



(b) Visualization of an edge placement restriction: virtual edge $(\mathbf{B}, \mathbf{C})$ may only be mapped on the depicted substrate edges by excluding the other substrate edges via the set $\overline{V}_S^{r,\mathbf{B},\mathbf{C}}$.



(c) Visualization of latency restrictions: each virtual edge is attributed with an upper bound such that the respective substrate paths may not exceed this latency.

Figure 2.1: Visualization of additional restrictions based on the example of Figure 1.1.

2.5 Approximate Embeddings

In the following we define the notion of approximate embeddings: instead of seeking a feasible embedding satisfying all capacity constraints for the various VNEP variants, one may consider solutions of bounded capacity violations. We refer to these as β-approximate (for node capacity violations) and γ-approximate (for edge capacity violations) embeddings. Again, we define the notion of approximate feasibility with respect to a set of mappings.

Definition 2.18 (β- / γ-Approximate Embeddings)**.**
A set of mappings $\{m_r\}_{r\in\mathcal{R}}$ is an approximate embedding, if the mappings are all valid but together violate substrate capacities within certain bounds. Specifically, we call an embedding β- and γ-approximate, when node and edge allocations are bounded by β and γ times the respective node or edge capacity. Formally, the following must hold for $\beta, \gamma \geq 1$:

$$\sum_{r\in\mathcal{R}} A(m_r, u) \leq \beta \cdot d_S(u) \qquad \forall u \in V_S$$
$$\sum_{r\in\mathcal{R}} A(m_r, u, v) \leq \gamma \cdot d_S(u, v) \quad \forall (u, v) \in E_S \qquad \square$$

One may also seek to relax the latency constraints, pertaining to the validity of mappings. Accordingly, we refer to a mapping which violates latency bounds by a factor δ as δ-approximate:

Definition 2.19 (δ-Approximate Mappings)**.**
A mapping m_r of a request r is a δ-approximate mapping, if latency constraints are obeyed within a factor δ, but is otherwise valid. Formally, the following must hold for $\delta \geq 1$:

$$\sum_{e\in m_E(i,j)} l_S(e) \leq \delta \cdot l_r(i, j) \qquad \forall (i, j) \in E_r \qquad \square$$

Besides these above relaxations on the feasibility and validity of embeddings, approximations for the VNEP are going to be studied and we use α to denote the approximation factors of the respective solutions. We naturally allow for the combination of these different notions of approximate solutions and write $(\alpha, \beta, \gamma, \delta)$-approximate to denote solutions which are within an α-factor of the optimum solution while being β-, γ-, and δ-approximate as defined above.

2.6 VNEP Taxonomy

Given the mapping restrictions above, we introduce the following taxonomy to concisely name the respective VNEP variants emerging from the different combinations of restrictions. Importantly, our taxonomy always refers to the directed variant with unsplittable edge mappings and whenever the undirected or splittable variant are considered, this will be explicitly stated.

Definition 2.20 (Taxonomy)**.** We use the notation $\langle\, \mathbf{C} \,|\, \mathbf{A} \,\rangle$ to indicate whether and which of the capacity constraints **C** and which of the additional constraints **A** are enforced.

C We denote by **V** node capacities, by **E** edge capacities, and by **-** that none are used. When node or edge capacities are *not* considered, we set the capacities of the respective substrate elements to ∞.

A For the additional restrictions **-**, **N**, **L**, and **R** stand for no restrictions, node placement, latency, and routing restrictions, respectively. $\square$

Hence, $\langle\, \mathbf{VE} \,|\, \mathbf{-} \,\rangle$ indicates the classic VNEP without additional constraints while obeying capacities and $\langle\, \mathbf{-} \,|\, \mathbf{NL} \,\rangle$ indicates the combination of node placement and latency restrictions without considering substrate capacities.

"Before there were computers, there were algorithms. But now that there are computers, there are even more algorithms, and algorithms lie at the heart of computing."

– Cormen et al. [Cor+09]

3 Algorithmic Background

Within this chapter the algorithmic foundations of this thesis are presented and some common notation is introduced. Specifically, in Section 3.1 the Big O notation is revisited while in Sections 3.2 and 3.3 fundamental notions of the classic and parametrized computational complexity theory are introduced. Then, in Sections 3.4 and 3.5 the foundations of Linear and Mixed-Integer Programming are discussed. Lastly, in Sections 3.6 and 3.7 the notions of approximation algorithms and competitive online algorithms are introduced.

3.1 Big O Notation

Let $F_{\mathbb{N}} = [\mathbb{N} \to \mathbb{N}]$ denote the space of all functions on natural numbers and let $f : \mathbb{N} \to \mathbb{N} \in F_{\mathbb{N}}$ be a specific function. We employ the Landau symbols $\mathcal{O}$ and o to denote the following classes of functions.

$$\mathcal{O}(f) = \left\{ g \in F_{\mathbb{N}} \mid \limsup_{n\to\infty} f(n)/g(n) < \infty \right\}$$
$$o(f) = \left\{ g \in F_{\mathbb{N}} \mid \limsup_{n\to\infty} f(n)/g(n) = 0 \right\}$$

While the above sets of functions specify (strict) upper bounds for the function f, we write $f \in \Omega(g)$, if $g \in \mathcal{O}(f)$ holds, and $f \in \omega(g)$, if $g \in o(f)$ holds, respectively. Additionally, if both $f \in \mathcal{O}(g)$ and $g \in \mathcal{O}(f)$ hold, we write $f \in \Theta(g)$ and $g \in \Theta(f)$ to denote that f and g are asymptotically equal. Furthermore, we use $\mathsf{poly}(f) \in f^{\mathcal{O}(1)}$ to denote any polynomial in the function f.

3.2 Classic Computational Complexity

In the following we introduce some computational complexity basics that are used in this thesis. For a more complete overview, we refer the reader to the excellent text books [AB09] for classic complexity theory and to [FG06; DF13] for an overview on parametrized complexity.

Letting Σ denote any finite and non-empty alphabet, the set of all words over Σ is denoted by $\Sigma^\star$. Given a boolean function $f : \Sigma^\star \to \{0, 1\}$, the language of all words *accepted* by f is denoted by $L_f = \{x \in \Sigma^\star \mid f(x) = 1\}$. Interpreting words as algorithmic inputs, comprising for example numbers, graphs, etc., deciding whether $x \in L_f$ holds for an input $x \in \Sigma^\star$ reduces to the *decision problem* of computing $f(x)$. The length of the word x is denoted by $|x|$.

Using Turing Machines as model of computation, a deterministic Turing Machine is said to decide L_f, iff. the Turing Machine computes the correct value $f(x)$ for all inputs $x \in \Sigma^\star$. A non-deterministic Turing Machine is said to decide L_f, if and only if, for an input $x \in L_f$ there exists some execution returning 1

while for $x \notin L_f$ it returns 0 along any execution path. The classes $\mathcal{P}$ and $\mathcal{NP}$ can accordingly be defined as follows.

Definition 3.1 (Complexity Classes $\mathcal{P}, \mathcal{NP}$)**.** Let $f \in F_{\mathbb{N}}$ denote a function. A language L is contained in $\textsc{Dtime}(f)$ (respectively $\textsc{Ntime}(f)$) if there exists a deterministic (respectively non-deterministic) Turing Machine deciding L that runs in time $\mathcal{O}(f(|x|))$. The complexity classes $\mathcal{P}$ and $\mathcal{NP}$ are the closure of $\textsc{Dtime}$ and $\textsc{Ntime}$ over all polynomial-time functions:

$$\mathcal{P} = \bigcup_{c \in \mathbb{N}} \textsc{Dtime}(n \mapsto n^c)$$
$$\mathcal{NP} = \bigcup_{c \in \mathbb{N}} \textsc{Ntime}(n \mapsto n^c) \qquad \square$$

To compare the complexity of decision problems, polynomial-time reductions are used:

Definition 3.2 (Polynomial-Time Reduction)**.** Given are two decision problems A and B together with their languages L_A and L_B. We say that A is reducible to B, if there exists a function $f : \Sigma^\star \to \Sigma^\star$ translating instances of A to instances of B, such that for all inputs $x \in \Sigma^\star$, $x \in L_a$ holds if and only if $f(x) \in L_B$ holds. If f can be computed in polynomial-time, then A is polynomial-time reducible to B and we write $A \preceq B$. $\square$

Accordingly, if $A \preceq B$ holds, then A can be decided by solving the decision problem for B and as A *reduces to* B, B can be considered at least as complex as A. As only polynomial-time reductions are considered in this thesis, the polynomiality of reductions is at times omitted and we simply use the term reduction. For problems contained in $\mathcal{NP}$, the following class of problems is of specific interest:

Definition 3.3 ($\mathcal{NP}$-Completeness)**.** A decision problem A is $\mathcal{NP}$-complete if and only if A is contained in $\mathcal{NP}$ and all other problems in $\mathcal{NP}$ reduce to A, i.e., $A \in \mathcal{NP}$ and $B \preceq A$ holds for any other problem $B \in \mathcal{NP}$. $\square$

Among other problems, the satisfiability problem of the propositional calculus (SAT) is known to be $\mathcal{NP}$-complete [Kar72]. To show the $\mathcal{NP}$-completeness of a decision problem A, it suffices to show that A is contained in $\mathcal{NP}$ and to show a polynomial-time reduction from any other $\mathcal{NP}$-complete problem B to A.

Several decision or optimization problems are computationally at least as hard as $\mathcal{NP}$-complete problems. This motivates the following notion of $\mathcal{NP}$-hardness.

Definition 3.4 ($\mathcal{NP}$-Hardness)**.** A Problem A is $\mathcal{NP}$-hard if an $\mathcal{NP}$-complete problem B can be solved in polynomial-time when given oracle access to solve instances of problem A. $\square$

Above, oracle access to A means that any instance of A can be solved in a single instruction while the time to construct the respective instances for A is counted towards the algorithm's runtime.

Lastly, the term *strongly* $\mathcal{NP}$-complete ($\mathcal{NP}$-hard) is used to indicate that the respective problem remains $\mathcal{NP}$-complete ($\mathcal{NP}$-hard) when the numeric input is given in its unary representation.

3.3 Parametrized Complexity and Tree Decompositions

As it is widely believed that $\mathcal{P} \neq \mathcal{NP}$ holds and for none of the $\mathcal{NP}$-complete problems polynomial-time algorithms are known, $\mathcal{NP}$-complete problems are generally believed to be *intractable*, i.e., not decidable in polynomial-time. However, over the course of the last decades, the notion of *parametrized*

complexity has been introduced to capture a more fine-grained *multivariate* notion of complexity [DF13]. A parametrization may concern, among others, the solution value or specific properties of the instances.

Formally, a parametrized language $L \subset \Sigma^\star \times \mathbb{N}$ consists of tuples $\langle x, k\rangle \in L$, where x represents the actual instance and k specifies the parametrization. For the respective parametrized decision problem, algorithms are studied which may use super-polynomial time in dependence of the parametrization.

Definition 3.5 (Parametrized Complexity Class $\mathcal{FPT}$ [DF13])**.** A parametrized language $L \subset \Sigma^\star \times \mathbb{N}$ lies in $\mathcal{FPT}$, if and only if there exists an algorithm and a constant $c \in \mathbb{N}$ together with a computable function $f \in F_{\mathbb{N}}$ such that for all $\langle x, k\rangle \in L$ the algorithm runs in time at most $f(k) \cdot |x|^c$ and decides $\langle x, k\rangle \in L$. □

Importantly, in the above definition the function f takes as parameter *only* the parameter k. Denoting by $L_k = \{\langle x, k\rangle \in L\}$ the k-th slice of L, the runtime $f(k) \cdot |x|^c$ is polynomial, with a parameter-depending constant factor $f(k)$.

Definition 3.6 (Parametrized Complexity Class $\mathcal{XP}$ [FG06])**.** A parametrized language $L \subset \Sigma^\star \times \mathbb{N}$ (uniformly) lies in $\mathcal{XP}$ if and only if there exists an algorithm together with a constant $c \in \mathbb{N}$ and a computable function $f \in F_{\mathbb{N}}$ such that for all $\langle x, k\rangle \in L$ the algorithm runs in time at most $|x|^{f(k)} + |x|^c$ and decides $\langle x, k\rangle \in L$. □

Note that for problems contained in $\mathcal{XP}$ each k-th slice lies in $\mathcal{P}$: $L_k \in \mathcal{P}$ holds for $k \in \mathbb{N}$. We refer to algorithms as being fixed-parameter tractable (FPT) or XP if their runtime is accordingly bounded by $f(k) \cdot |x|^c$ and $|x|^{f(k)} + |x|^c$ for constants $c \in \mathbb{N}$ and a computable function $f \in F_{\mathbb{N}}$.

A common parametrization for graph-theoretic problems is the treewidth of tree decompositions, which we revisit in the following [Bod97; Bod98; FG06]. In a nutshell, tree decompositions are used to represent arbitrary graphs as trees. The definition of tree decompositions ensures that (i) all nodes and edges of the original graph are *covered* while (ii) preserving crucial *structural information* of the (undirected) original graph.

Definition 3.7 (Tree Decomposition $\mathcal{T} = (T, \mathcal{B})$)**.** Given an undirected graph $G = (V, E)$, a tree decomposition of G is a pair $\mathcal{T} = (T, \mathcal{B})$ consisting of an undirected tree $T = (V_T, E_T)$ and a family $\mathcal{B} = \{B_t\}_{t \in V_T}$ of subsets $B_t \subseteq V$, also referred to as the node bags, for which the following conditions hold:

(1) For all nodes $u \in V$, the set $V_T^{-1}(u) = \{t \in V_T \mid u \in B_t\}$ of tree nodes containing node u is connected in T.

(2) Each node and each edge is contained in at least one of the bags, i.e., for all nodes $u \in V$ there exists a tree node $t \in V_T$, such that $u \in B_t$ holds, and for all edges $\{u, v\} \in E$ there exists a tree node $t \in V_T$, such that $\{u, v\} \subseteq B_t$ holds. □

The treewidth is then defined as follows (cf. [FG06]).

Definition 3.8 (Width of a Tree Decomposition and Treewidth)**.** The width $\mathrm{tw}(\mathcal{T}) \in \mathbb{N}$ of a specific tree decomposition $\mathcal{T}$ equals the maximal bag size minus one, i.e., $\mathrm{tw}(\mathcal{T}) = \max_{t \in V_T} |B_t| - 1$. The treewidth of an undirected graph equals the minimal width among all decompositions. □

Finding tree decompositions of minimal width is itself a challenging optimization problem and is known to be $\mathcal{NP}$-hard [FG06]. However, if the treewidth of a graph G is known to be $k \in \mathbb{N}$, the problem of finding a tree decomposition is fixed-parameter tractable [Bod96]. Several graph classes are known to have bounded treewidths (cf. Table 3.1). While all mentioned graph classes of Table 3.1 are planar, it is a well-known result that planar graphs in general do not exhibit bounded treewidth. In particular, a $k \times k$ grid has a treewidth of k [FG06].

Graph Class	Treewidth	Description
trees	1	Connected graph without cycles.
cacti	2	Two cycles intersect at most in a single node.
series-parallel	2	Source-terminal graphs; generated only using parallel and serial composition.
(1-)outerplanar	2	Planar graph, whose nodes all lie on the outer face.
k-outerplanar	$k+1$	Planar graph; removal of nodes on outer face yields $(k-1)$-outplanar graph.

Table 3.1: Graph Classes of Bounded Treewidth [Bod98]

3.4 Linear Programming

In the following we summarize important results on Linear Programming. Our presentation is mostly based on the text book by Matousek and Gärtner [MG07].

A Linear Program (LP) is a convex optimization problem over continuous variables, linear constraints, and a linear objective. Specifically, an LP of $n \in \mathbb{N}$ variables and $m \in \mathbb{N}$ constraints in *standard form* can be stated as follows:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{n} c_j \cdot x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} \cdot x_j \leq b_i \qquad \forall i \in \{1, \dots, m\} \\ & x_j \geq 0 \qquad \forall j \in \{1, \dots, n\} \end{aligned}$$

The same LP can be represented in matrix notation as follows.

$$\text{maximize} \;\; \mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} \;\; \text{subject to} \;\; A \cdot \mathbf{x} \leq \mathbf{b} \text{ and } \mathbf{x} \geq 0 \tag{\textbf{P}}$$

Here, the matrix $A = (a_{ij} \in \mathbb{Q}^{m \times n})$ specifies the coefficients of the LP while the vectors $b \in \mathbb{Q}^m$ and $c \in \mathbb{Q}^n$ specify the left-hand side of the constraints and the objective respectively.

We denote by $\mathcal{F}_{\mathrm{P}} = \{x \in \mathbb{Q}^n \mid A \cdot \mathbf{x} \leq \mathbf{b}\}$ the set of feasible solutions of this LP. If no feasible solution exists, i.e., if $\mathcal{F}_{\mathrm{P}} = \emptyset$ holds, then the LP is referred to as *infeasible*. A feasible LP is referred to as *unbounded* if no optimal solution exists, i.e., if $\sup\{\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} \mid \mathbf{x} \in \mathcal{F}_{\mathrm{P}}\}$ does not exist.

An important concept in Linear Programming is *duality*, which allows to associate with each maximization LP an equivalent minimization LP and vice versa. Specifically, the *dual* of the above introduced *primal* LP (**P**) over the variables $y \in \mathbb{Q}^m$ is given by:

$$\text{minimize} \;\; \mathbf{b}^{\mathrm{T}} \cdot \mathbf{y} \;\; \text{subject to} \;\; A \cdot \mathbf{y} \geq \mathbf{c} \text{ and } \mathbf{y} \geq 0 \tag{\textbf{D}}$$

Notably, the number of constraints and variables of the dual equals the number of variables and constraints of the primal, respectively. Denoting the set of feasible solutions of (D) by $\mathcal{F}_{\mathrm{D}}$, the following theorems are folklore:

Theorem 3.9 (Weak Duality [MG07])**.** Considering any feasible primal and any feasible dual solution, the objective value of the dual solution is an upper bound of the objective value of the primal solutions. Formally, for any $x \in \mathcal{F}_{\mathrm{P}}$ and any $y \in \mathcal{F}_{\mathrm{D}}$ the following holds: $\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} \leq \mathbf{b}^{\mathrm{T}} \cdot \mathbf{y}$.

Theorem 3.10 (Strong Duality [MG07])**.** For any primal LP (**P**) and any corresponding dual LP (**D**), exactly one of the following statements hold:

1. Both (**P**) and (**D**) are infeasible.
2. (**P**) is unbounded and (**D**) is infeasible.
3. (**D**) is unbounded and (**P**) is infeasible.
4. Both (**P**) and (**D**) are feasible and for any optimal primal solution $\hat{\mathbf{x}} \in \mathcal{F}_{\mathrm{P}}$ and any optimal dual solution $\hat{\mathbf{y}} \in \mathcal{F}_{\mathrm{D}}$ the objective values are equal, i.e., $\mathbf{c}^{\mathrm{T}} \cdot \hat{\mathbf{x}} = \mathbf{b}^{\mathrm{T}} \cdot \hat{\mathbf{y}}$.

An algorithm that solves an LP either detects infeasibility or unboundedness or returns an optimal solution. Several algorithms to solve LPs exist. The Simplex algorithm was first developed by George Dantzig in 1947: it iterates over basic solutions of the LP that correspond to extreme points of the corresponding polyhedron. The order in which these extreme points are iterated over is specified by the *pivot* rule. While the Simplex algorithm performs well for many instances, no pivot rule is known for which the Simplex attains a polynomial runtime in the *worst-case*. Current results observed that the classic Simplex algorithm under Dantzig's pivot rule is $\mathcal{NP}$-mighty, i.e., it can be used to (implicitly during the execution) solve any problem contained in $\mathcal{NP}$ [DS18].

The first polynomial-time algorithm for Linear Programming was derived by Leonid Khachyian in 1979 and was coined the Ellipsoid algorithm, as it is based on the ellipsoid method invented in 1970 by Shor, Judin, and Nemirovski. While the Ellipsoid algorithm was not of significance for practical applications, other polynomial-time algorithms known as 'interior point' methods have had more practical impact and we state the following theorem:

Theorem 3.11 (Polynomial-Time Algorithms for LPs [Ans99])**.** Considering an LP with n variables and letting L denote the total bit length for describing the matrix A and the vectors $\mathbf{b}, \mathbf{c}$, the LP can be solved (to optimality) in time $\mathcal{O}(n^3 \cdot L / \ln n)$.

Despite the Ellipsoid algorithm's practical insignificance, it was of tremendous importance. Among other results, Grötschel, Lovazs and Schrijver proved the following central result based on the Ellipsoid algorithm:

Theorem 3.12 (Optimization via Separation [GLS88])**.** Consider a Linear Program over n variables where the constraints $\{(a_i, b_i) \in \mathbb{Q}^n \times \mathbb{Q}\}_{i \in \mathcal{I}}$ are given via an index set $\mathcal{I}$, such that $\mathbf{a}_i^{\mathrm{T}} \cdot \mathbf{x} \leq b_i$ must hold for each $i \in \mathcal{I}$. Given access to a polynomial-time *separation* oracle that determines if any of the constraints is violated (and returns one if it is violated), an optimal solution can be computed using an extension of the Ellipsoid algorithm in *polynomial-time*. This also holds true when the number of constraints, i.e., the size of the index set, is *exponential*.

The above result also has repercussions beyond the solvability of LPs with an exponential number of constraints. In fact, the above method also enables so called *column generation* approaches, which are employed when the number of variables is exponential while the number of constraints is polynomial [DL05; MG07]. Specifically, given such an LP, its dual will have an exponential number of constraints and only a polynomial number of variables. Being able to separate the constraints in polynomial-time, an optimal dual solution can be computed. Even more, the generated dual constraints correspond to variables (i.e., columns) in the primal LP. Thus, in practice the following scheme is predominant which ensures primal feasibility throughout its execution: given an initial feasible primal solution, the dual values can be computed and the separation task is performed, yielding new columns then introduced in the primal LP [MG07]. Recomputing the primal LP solution and iterating this process as long as violated constraints exist, the primal LP can be solved to optimality. Note that the separation process is also referred to as *pricing*, as newly generated variables correspond to columns of negative reduced cost which might then enter the (simplex) basis.

For solving Linear Programs several industrial as well as academic optimization suites are available and we mention just a few. Gurobi [Gur19], IBM CPLEX [IBM19], and Fico Xpress [FIC19] provide commercial LP solvers which may be used for free for academic purposes. Open source LP solvers are provided as part of the SCIP Optimization Suite [Gle+18] and as part of the Google OR-Tools [Goo19]. Within this thesis, all LP computations have been performed using Gurobi. Gurobi offers both primal and dual Simplex implementations as well as an implementation of the Barrier interior-point algorithm.

3.5 (Mixed-)Integer Programming

An important extension of Linear Programming is Integer Programming. In Integer Programs (IPs) the variables are not continuous but constrained to *integer* numbers. Mathematical formulations employing both integer and continuous variables are referred to as Mixed-Integer Programs (MIPs), while Integer Programs whose variables are restricted to $\{0, 1\}$ are referred to as Binary Programs (BPs). In the following, we simply use the term Integer Programming. However, all discussed results are applicable to any form of Integer Programs.

In contrast to Linear Programs which can be solved in polynomial-time, solving IPs lies in $\mathcal{NP}$ [Pap81] and finding solutions to Binary Programs was among the first 21 problems to be proven $\mathcal{NP}$-complete by Karp in 1972 [Kar72]. Given an Integer Program, a corresponding Linear Programming relaxation is obtained by relaxing the integrality constraints of the variables. Denoting by $\mathcal{F}_{\text{IP}}$ the set of feasible integral solutions and by $\mathcal{F}_{\text{LP}}$ the set of feasible continuous solutions, we note that $\mathcal{F}_{\text{IP}} \subseteq \mathcal{F}_{\text{LP}}$ holds. Accordingly, the optimal LP relaxation solution is always a bound for the best integer solution.

To solve Integer Programs several solution techniques exist. The most common one is branch-and-bound [MG07], which employs a search tree (branching) together with the usage of Linear Programming relaxations to compute (local) bounds on the objective value. These LP relaxations are not only used to guide the search for the best solutions, but are also used to cut off parts of the search tree. In particular, considering a search tree node, whose local objective value is x, then the node itself (and all potential children) can be discarded if a solution of value greater or equal to x is known (for maximization problems). This intuitively holds true as no feasible integer solution of higher objective value can exist. The branching process creates children by partitioning the space of values that the integer variables may attain.

The above mentioned software packages for solving LPs also allow for solving IPs and feature various (proprietary) heuristics to perform branching and heuristics to compute feasible solutions. Tobias Achterberg's work on the open-source solver SCIP [Ach09] exemplifies the tremendous complexity of state-of-the-art IP solvers. In particular, SCIP v6.0 has more than 2,400 parameters and contains several dozen different heuristics for performing the branching, deciding which node to branch on, etc. [Gle+18].

While solving IPs to optimality may be very time consuming, IP solvers report at any point in time during the execution on the current *objective gap* that is computed based on the currently best known upper bound and the best incumbent solution. Referring to the upper bound as *dual bound* D and to the best current solution as *primal bound* P, the objective gap is computed via $|P - D|/|P|$. Accordingly, solutions computed by IP solvers always come with a relative quality guarantee (unless $|P|$ equals zero) and in practice often times the solution process is terminated when a specified objective gap is reached.

To strengthen the LP relaxations, IP solvers often generate additional constraints which are valid based on the integrality of the variables. To give a minimal example, considering two binary variables $x, y \in \{0, 1\}$ and the constraint $x + y \leq 1.5$, this constraint can be tightened to $x + y \leq 1$ and we refer the interested reader to [Cor08] for an in-depth discussion of cut generation techniques. While such cuts may be derived automatically by the solver, it is at times beneficial to add custom constraints to the model to improve the LP relaxations. These constraints, which do not cut off any integer solution while tightening the convex hull of the respective LP relaxations are called *valid inequalities* or *user cuts*. Whenever cuts are added in

conjunction with using the branch-and-bound method to solve the IP, the resulting algorithm is referred to as branch-and-cut.

Lastly, IPs can also be solved by combining column generation techniques with branch-and-bound methods. Such methods are referred to as branch-and-price.

3.6 (Randomized) Approximation Algorithms

In the following theoretic notions pertaining to approximation algorithms in general and to randomized approximation algorithms in particular are introduced. The following definitions are taken from the text book [WS11].

Definition 3.13 (Approximation Algorithm [WS11])**.** An α-approximation algorithm for an optimization problem is a polynomial-time algorithm that for all instances of the problem produces a solution whose value is within a factor of α of the value of an optimal solution. □

Accordingly, for maximization problems the approximation factor α lies (in general) below 1 while for minimization problems approximation factors greater than 1 are considered. For some problems approximation schemes exist, allowing the computation of approximate solutions which are arbitrarily close to the optimum.

Definition 3.14 (Polynomial-Time Approximation Scheme (PTAS) [WS11])**.** A polynomial-time approximation scheme is a family of algorithms $\{A_\epsilon\}$, where there is an algorithm for each $\epsilon > 0$, such that A_ϵ is a $(1+\epsilon)$-approximation algorithm (for minimization problems) or a $(1-\epsilon)$-approximation (for maximization problems). □

Definition 3.15 (Fully Polynomial-Time Approximation Scheme (FPTAS) [WS11])**.** A fully polynomial-time approximation scheme is an approximation scheme such that the running time of A_ϵ is bounded by a polynomial in $1/\epsilon$ and the instance's input size. □

While the polynomial runtime of approximations is a crucial part of the above definitions, within the framework of parametrized complexity also FPT- and XP-approximations are considered, such that the approximation algorithm's runtime is polynomial for a fixed parameter.

Linear Programming is often used for designing approximation algorithms [MR95; WS11]. Here, the notion of the integrality gap plays an important role, as it measures the (worst-case) ratio of the LP objective to the objective value of an optimal solution [WS11]. Approximations can often be obtained either by interpreting the LP values as probabilities and applying randomized rounding [RT85] or via deterministic rounding by employing, among others, greedy techniques [WS11]. Randomized algorithms often only yield approximate solutions with some probability. However, by repeatedly executing the algorithm in a Monte Carlo fashion, the probability to obtain an approximate solution then approaches 1. Specifically, whenever a randomized algorithm can be used to obtain an approximate solution with probability $1 - 1/n$ for any value n in polynomial-time in $n \in \mathbb{N}$, we say that the approximation yields a solution *with high probability*.

To bound tail probabilities of 'bad' events, the following measure of concentration bounds, which are proven in Appendix A, are useful. While similar results are contained in any textbook on randomized algorithms [MR95; DP09; WS11; AS16; MU17], the bounds proven in the related work are less general (see discussion in Appendix A).

Theorem A.1. Let X be the sum of N random variables $X_1, \ldots, X_n$ with $X_i \in [0,1]$ for $i \in [N]$. Denoting by $\hat{\mu}_i \geq \mu_i = \mathsf{E}[X_i]$ upper bounds on the expected value of random variable X_i, $i \in [N]$, the following holds for any $\delta > 0$ and $\hat{\mu} = \sum_{i \in [N]} \hat{\mu}_i$:

$$\mathsf{Pr}[X \geq (1+\delta) \cdot \hat{\mu}] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}} \right)^{\hat{\mu}} \leq e^{-\delta^2 \cdot \hat{\mu}/3} \,. \tag{A.1}$$

Theorem A.2. Let X be the sum of N random variables $X_1, \ldots, X_n$ with $X_i \in [0,1]$ for $i \in [N]$. Denoting by $\tilde{\mu}_i \leq \mu_i = \mathsf{E}[X_i]$ lower bounds on the expected value of random variable X_i, $i \in [N]$, the following holds for any $\delta \in (0,1)$ and $\tilde{\mu} = \sum_{i \in [N]} \tilde{\mu}_i$:

$$\mathsf{Pr}[X \leq (1-\delta) \cdot \tilde{\mu}] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}} \right)^{\tilde{\mu}} \leq e^{-\delta^2 \cdot \tilde{\mu}/2} \,. \tag{A.2}$$

Theorem A.3. Let $X = \sum_{i \in [N]} X_i$ be the sum of $N \in \mathbb{N}$ *independent* random variables with $X_i \in [0,1]$ and let $\mu_i = \mathsf{E}[X_i]$ denote their respective expected value for $i \in [N]$. Let $\hat{\mu}_i \geq \mu_i$ be an upper bound on the expected value of $X_i \in [0,1]$, $i \in [N]$, and let $\hat{\mu} = \sum_{i \in [N]} \hat{\mu}_i$ such that,

$$\hat{\mu} = \sum_{i \in [N]} \hat{\mu}_i \geq 1 \,. \tag{A.3}$$

Denote by c, n, λ, ρ, ξ constants, such that,

$$\begin{aligned} c \geq& 2 && \text{(A.4)} \\ n \geq& 3 && \text{(A.5)} \\ \rho \geq& 2 && \text{(A.6)} \\ \rho \cdot c \geq& e && \text{(A.7)} \\ \xi \geq& \ln \ln n / \ln n && \text{(A.8)} \\ \lambda =& \rho \cdot c \cdot \xi \cdot \ln n / \ln \ln n \,. && \text{(A.9)} \end{aligned}$$

The following holds:

$$\mathsf{Pr}[X \geq \lambda \cdot \hat{\mu}] \leq 1/n^{c \cdot \hat{\mu} \cdot \xi} \,. \tag{A.10}$$

3.7 Competitive Online Algorithms

Lastly, we briefly revisit the notion of *competitive* online algorithms [BE05; BN09]. An online algorithm is given inputs sequentially and only in parts. For each part of the input a decision of how to process the input has to be made, while previously made decisions can in general not be revoked [BN09]. Online algorithms are generally required to exhibit polynomial runtimes. To make this rather abstract outline tangible, in the following online algorithms are discussed in the context of the VNEP.

For the online VNEP setting, it is assumed that requests arrive over time and upon the arrival the request either has to be embedded or to be rejected [Eve+13]. Importantly, the sequence of (future) requests is not known to the algorithm invoked for performing the processing. The objective of the online algorithm may be to minimize congestion [Ban+15], or to maximize the provider's profit as in the profit offline VNEP (cf. Definition 2.5) [Eve+13].

The notion of approximations can be extended in the following way to the online setting. For the definition, we assume a maximization objective, as, e.g., to maximize the provider's profit. An algorithm is called α-competitive, if the algorithm always achieves at least an α-fraction of the profit of an optimal offline

algorithm, which is given (i) unlimited computation time as well as (ii) the knowledge of the whole (possibly infinite) request sequence. Accordingly, an α-competitive algorithm is robust and yields performance guarantees independent of the sequence or structure of the requests arriving in the future.

To develop competitive algorithms and prove their *competitiveness*, several methods have been studied, e.g., the potential function method described in [BE05]. Also, the primal-dual approach has been successfully applied to prove competitiveness for various problems [BN09]. In the following, the basics of the primal-dual method are put forth and we refer the reader to [BN09] for an in-depth discussion.

The key idea of the primal-dual method is to consider a (primal) Linear Programming relaxation of the studied problem together with its dual. An algorithm based on the primal-dual method will at any point in time store a (feasible) solution to the primal and the dual LP formulations containing the inputs received thus far. Upon the arrival of a new input, both the primal and the dual solution have to be adapted in a way to preserve (at least dual) feasibility. The key idea for the analysis is then to prove that upon each new input, the primal and dual objectives diverge only by some bounded factor α. Then, as the (feasible) dual solution is a bound for the *optimal offline* solution, the α-competitiveness is proven.

"People think that computer science is the art of geniuses but the actual reality is the opposite, just many people doing things that build on each other, like a wall of mini stones."

– Donald E. Knuth

4 Related Work

In this chapter related works pertaining to the VNEP are discussed. As a brief overview on the applications of the VNEP and the different model restrictions was given in Chapter 1, we restrict the discussion to algorithmic approaches. Furthermore, we exclude related work pertaining to the applications presented in Chapters 8 and 9.

In Section 4.1 known results on the complexity of the VNEP are reviewed, while in Sections 4.2 and 4.3 known results pertaining to online and offline algorithms for the VNEP are revisited.

4.1 Computational Complexity of the VNEP

Despite the relevance of the VNEP and the large body of literature on it [Fis+13], the complexity of the VNEP has not received much attention. While it can be easily seen that the Virtual Network Embedding Problem encompasses several $\mathcal{NP}$-hard problems as, e.g., the k-disjoint paths and unsplittable flow problems [KS97; Chu+07], the minimum linear arrangement problem [DPS02], or the subgraph isomorphism problem [Epp02], many works on the VNEP, e.g., [CRB12; Fis+13], cite a $\mathcal{NP}$-hardness result contained in a technical report from 2002 by Andersen [And02]. However, the proof is a sketch at best and the authors of [Ama+16] note that the technical report lacks sufficient detail to verify its correctness.

Accordingly, only recently in 2016, the first formal $\mathcal{NP}$-hardness proofs were given by Amaldi et al. [Ama+16] for the profit offline VNEP on undirected graphs:

Theorem 4.1 ($\mathcal{NP}$-hardness results presented in [Ama+16])**.**
The following holds for the profit offline VNEP on undirected graphs.

(1) VNEP $\langle$ **VE** | **-** $\rangle$ is strongly $\mathcal{NP}$-hard.

(2) VNEP $\langle$ **VE** | **N** $\rangle$ is inapproximable within a factor $|V_S|^{1/2-\epsilon}$ for any $\epsilon > 0$, unless $\mathcal{P} = \mathcal{NP}$.

(3) VNEP $\langle$ **VE** | **-** $\rangle$ is strongly $\mathcal{NP}$-hard even when considering only a single request.

(4) VNEP $\langle$ **VE** | **-** $\rangle$ is strongly $\mathcal{NP}$-hard even when all requests consist only of a single node.

Our analysis of the computational complexity of the VNEP (cf. Chapter 5) extends these results significantly, showing, among others, the $\mathcal{NP}$-completeness of the (decision) VNEP in the setting $\langle$ **-** | **NR** $\rangle$, i.e., when only considering node placement and routing restrictions in the absence of capacities.

4.2 Online Algorithms

In the following prior algorithms for the online variant of the VNEP are presented. Here, the requests are assumed to arrive over time and admission control must be exerted instantaneously: the request is either rejected or it is accepted and a feasible embedding (with respect to the residual capacities) must be found. We group the works according to their guarantees: heuristics do not provide any optimality guarantees, while approximations do for a single request, and competitive online algorithms provide guarantees for any sequence of requests.

4.2.1 Heuristics

Most published works consider heuristics [Yu+08; CRB09; LK09; Che+11]. In the respective works, the main task is to minimize the resources used for embedding each individual request to obtain a high acceptance rate of requests and accordingly increase the substrate provider's profit. Besides minimizing the resource footprint, also load-balancing [CRB09] and the avoidance of resource-fragmentation [Che+11] are considered. Most of the early works considered undirected request graphs. Furthermore, early on the idea of *splittable* embeddings has been proposed [Yu+08; CRB09]. Importantly, all the heuristics for the online VNEP can be considered to be greedy in the following sense: requests are always accepted, if the respective embedding heuristic has found a feasible solution. As embeddings of requests may be long lasting and the successive heuristic embedding process might lead to resource fragmentation and sub-optimal solutions in general, migrations of virtual nodes and edges have been studied in [Yu+08; SSF12; Bie+14].

ViNE Heuristics. In the following, we discuss the ViNE heuristics [CRB09; CRB12], which are one of the most cited families of heuristics for the online VNEP. We discuss these, as they are based on Linear Programming and will be used as a baseline in our evaluations. In particular, the authors of [CRB09; CRB12] propose the usage of the Multi-Commodity Flow (MCF) LP formulation (discussed in depth in Section 6.1) to guide the embedding process of single requests. In essence, the MCF formulation uses node assignment variables $y_i^u \in [0,1]$ for each virtual node $i \in V_r$ and each substrate node $u \in V_S$, subject to the natural constraint that for each virtual node these node assignment variables sum to 1. Similarly, for edge mappings, edge assignment variables $z_{i,j}^{u,v} \in [0,1]$ are used to indicate whether the substrate edge $(u,v) \in E_S$ lies on the path of the virtual edge $(i,j) \in E_r$. By linking node and edge mapping variables in the LP and considering capacity constraints, the first heuristic approach *coordinating node and edge mappings* was obtained which outperformed existing heuristics significantly. Specifically, the authors of [CRB09; CRB12] proposed the following heuristics to find feasible embeddings given the LP solution. Firstly, the node mappings are computed by either deterministically choosing the substrate node u maximizing y_i^u for each virtual node or the node mapping variables are interpreted as *probabilities* and the node mappings are accordingly probabilistically rounded. After having found feasible mappings for all nodes, the splittable edge mappings are either computed optimally using the MCF formulation (under the given node mappings), or when considering the unsplittable VNEP, the edges are realized by employing (capacity observing) shortest paths. Furthermore, the authors of [CRB09; CRB12] proposed two different objectives for the LP formulation: one simply minimizing resource usage and the second one performing load balancing.

4.2.2 Approximation Algorithms

The author of this thesis is only aware of a single approximation result in the online algorithm realm. In particular, Bansal et al. have studied, among others, the embedding of single d-depth trees on arbitrary substrate graphs [Ban+15]. Here, a d-depth tree is defined as any (rooted) tree for which any leaf is reachable from the root with less than d hops. The following XP-time result was obtained for undirected graphs:

Theorem 4.2 (Online Embedding Approximation for d-depth Trees [Ban+15])**.** There is a randomized (α, β, γ)-approximation for the online embedding of d-depth trees, which runs in time $|V_S|^{\mathcal{O}(d)}$, with $\alpha = 2$ and $\beta = \gamma = \mathcal{O}(d^2 \cdot \log(|V_S| \cdot d))$.

The result is based on strong Linear Programming relaxations which can be considered to be inspired by the Sherali-Adams hierarchy [SA90]. In particular, Bansal et al. introduce LP variables for (any prefix of) each path from the root to any of the leaves and enumerate *all* mapping possibilities for the respective

nodes. These variables are then linked together in such a way that the LP variables model *conditional* probabilities: considering a specific root-leaf path (prefix) of length k, the weight of the $|V_S|$ many path extensions of length $k+1$ sum to the weight of the path (prefix) of length k. The crucial idea to obtain the approximation result is to introduce valid cuts, referred to as *conditional congestion* constraints, which help in bounding the resource violations.

Considering the runtime bound stated in Theorem 4.2, we note the following. It is easy to see, that the runtime for the algorithm by Bansal et al. is bounded by $\mathsf{poly}(|V_S|^d \cdot |V_r|)$, where the $\mathsf{poly}()$ factor arises, among others, from solving the LP and the $|V_r|$ factor has to be considered as the number of paths is polynomial in the number of leaves. However, under the assumption that $|V_r| \in \mathcal{O}(|V_S|^d)$ holds, the runtime can be equivalently bounded by $|V_S|^{\mathcal{O}(d)}$ as done by Bansal et al., where all the polynomial factors are encapsulated in the exponent.

4.2.3 Competitive Online Algorithms.

While the study of online algorithms is mostly restricted to heuristics, also some competitive online algorithms have been studied in [Eve+13; Bie+14; EMP16]. Specifically, in [Eve+13] competitive online algorithms for the VNEP under fixed node placements was studied and the following result has been shown for the profit maximization objective.

Theorem 4.3 (General VNet Online Packing (GVOP) Algorithm [Eve+13])**.** Assuming that feasible virtual network embeddings can be optimally computed, then the GVOP algorithm is $(1/2, \gamma)$-competitive, i.e., it achieves at least half the profit of an optimal offline solution, while violating edge capacities by a factor of at most $\gamma = \mathcal{O}\left(\log(|E_S|) \cdot (\max_{e \in E_S} d_S(e)) \cdot b_{\max}\right)$, where $b_{\max}$ denotes the maximum profit of any request.

Note that the restriction of fixed node mappings, has been due to the fact that at the time of publication no polynomial-time algorithms were known for the (near-)optimal embedding of requests. However, the result can be extended to obtain competitive algorithms under flexible node mappings. Specifically, based on results obtained in this thesis on the polynomial-time solvability of the Virtual Cluster Embedding Problem, Fattohi applied the GVOP algorithm in his thesis [Fat18] to obtain competitive online algorithms for the embedding of virtual clusters.

Besides the result of [Fat18], an $\Omega(1/\log(|V_S| \cdot |E_{\max}|))$-competitive online algorithm was proposed in [EMP16] for *service chains*, where $|E_{\max}|$ denotes the maximal number of edges of any of the requests.

Bansal et al. have also studied competitive online algorithms in their work [Ban+15]. However, they study the *congestion minimization* objective: each request has to be embedded but the overall goal is to minimize the maximal congestion at any point in time. Given an (α, β, γ)-approximation for the online embedding of a single request, the algorithm can be used as oracle to obtain a $\mathcal{O}((\alpha + \max\{\beta, \gamma\}) \cdot \log(|V_S| \cdot D))$-competitive online algorithm, where D is the ratio of maximal to minimal durations. Accordingly, based on their result for d-depth trees (cf. Theorem 4.2), an $\mathcal{O}((2 + d^2 \cdot \log(|V_S| \cdot d)) \cdot \log(|V_S| \cdot D))$-competitive online algorithm is obtained. However, the online algorithm's runtime is again parametrized by the maximal depth of any considered tree and the algorithm accordingly only runs in polynomial-time when d is upper bounded by some constant.

Lastly, Bansal et al. have also studied the congestion minimization for graphs which are complete and have uniform demands [Ban+15]. Their result for complete graphs is based on Räcke tree decompositions [Räc08]: the substrate graph is represented as a tree, in which the substrate nodes are leaves, such that the capacity between any pair of leaf nodes is a logarithmic approximation of the actual bottleneck cut-capacity in the original graph. Bansal et al. use these tree decompositions mainly to be able to embed the (uniform and complete) request by performing dynamic programming. Their algorithm is shown to be $\mathcal{O}(\log^2 |V_S| \cdot \log\log |V_S| \cdot \log(|V_S| \cdot \mathcal{T}))$-competitive, where $\mathcal{T}$ denotes the overall considered time horizon of the online algorithm execution.

4.3 Offline Algorithms

Offline algorithms have first been developed in the realm of the VNEP to improve the performance of online heuristics by considering batches of requests [CRB12; JK15]. Intuitively, by collecting a set of requests and collectively performing admission control, the chance to select the most profitable subset of requests is increased while also potentially reducing the induced (overall) resource allocations.

4.3.1 Heuristics

Heuristics for the offline VNEP are given for example in [HLZ08; CRB12; JK15; Ném+18]. Specifically, the authors of the ViNE heuristics proposed the WiNE (windows-based Virtual network Embedding) greedy heuristic [CRB12]: requests are ordered descendingly according to their profits and are embedded sequentially using a heuristic. The authors studied the performance of WiNE using their ViNE heuristics and observed that the acceptance ratio could be increased [CRB12] by batching requests into groups.

An interesting work close to this thesis is the work of Jarray and Karmouch [JK15], in which a more involved heuristic for the offline profit VNEP was proposed. Using column generation methods, an adaptation of the fractional offline profit VNEP (cf. Definition 2.7) is solved and accordingly for each request a set of weighted mappings $\mathcal{D}_r = \{(f_r^k, m_r^k) \mid f_r^k > 0, m_r^k \in \mathcal{M}_r\}$ is returned. In fact, by combining heuristic and exact methods, their column generation approach *only* returns mappings which are (in the absence of other mappings) *feasible*. This is achieved by performing the separation procedure heuristically in a first step, and, if no feasible solution was found by the heuristic, a Mixed-Integer Program is used to perform the separation procedure *optimally*. After having computed the optimal LP solution according to their model that only considers feasible valid mappings, an integral solution is computed either by transforming the LP into an Integer Program and solving it using branch-and-bound techniques or computing a heuristic solution by iteratively and deterministically selecting mappings that maximize the profit.

4.3.2 Approximations

Bansal et al. also gave offline approximations in their seminal work [Ban+15]. Notably again, the authors study the congestion minimization objective and *no admission control* is performed, i.e., all requests must be embedded. For this setting, two approximation results were presented. Their first result is a very general one. Specifically, given an (α, β, γ)-approximation algorithm for the online embedding of a single request, an $\mathcal{O}((\alpha + \max\{\beta, \gamma\}) \cdot \log |V_S| / \log\log |V_S|)$ offline algorithm is obtained. In particular, the offline algorithm employs the approximation algorithm as an approximate separation oracle to compute an *approximate* Linear Programming solution which is then turned into an integral solution using randomized rounding. Given the XP-time approximation for d-depth trees, a respective offline result was shown for this specific type of request.

Furthermore, Bansal et al. also give an $\mathcal{O}(\log^3 |V_S|)$ offline approximation when request graphs have uniform demands and the request graphs are complete. Again, this result employs Räcke tree decompositions [Räc08].

The first (polynomial-time) approximation algorithm in the offline setting was given by Even, Schmid, and the author of this thesis in [ERS16b]. Specifically, the profit variant of the VNEP $\langle \mathbf{E} \,|\, \mathbf{NR} \rangle$ is considered for *service chains*, i.e., request realizations are linear chains. By considering Linear Programming solutions and rounding them as proposed by Raghavan and Thompson [RT85], the following results are obtained.

Theorem 4.4 (Approximation of Offline Profit VNEP $\langle \mathbf{E} \,|\, \mathbf{NR} \rangle$ for Service Chains [ERS16a])**.** The following results hold under the assumption that request demands are small compared to the capacities, i.e., when $\dfrac{\min_{(u,v)\in E_S} d_S(u,v)}{\max_{r\in\mathcal{R}} |E_r| \cdot \max_{r\in\mathcal{R},(i,j)\in E_r} d_r(i,j)} \geq \dfrac{(4.2+\epsilon)\cdot(1+\epsilon)\cdot \ln |E_S|}{\epsilon^2}$ holds for some $\epsilon > 0$:

(1) The probability to exceed any edge capacity is upper bounded by $1/|E_S|$ and the probability to achieve less than a $(1-\epsilon)/(1+\epsilon)$ fraction of the optimal profit b_{opt} is upper bounded by $-\exp\big(((1-\epsilon)\cdot\ln(1-\epsilon)-\epsilon)\cdot b_{\text{opt}}/(\max_{r\in\mathcal{R}} b_r \cdot \max_{r\in\mathcal{R},(i,j)\in E_r} d_r(i,j))\big)$.

(2) Under the additional assumptions that all requests have the same profit and that $b_{\text{opt}} > \min_{(u,v)\in E_S} d_S(u,v)$ holds, a $(1-\mathcal{O}(\epsilon))$-approximation is obtained *with high probability* that does not violate edge capacities.

Notably, the above approximation result (2) only holds under the stated assumptions on the demand to capacity ratio and uniform benefits, and are only applicable for linear chains. Compared to this prior result, the approximations given within this thesis hold for general request graphs and do neither require assumptions on the demand to capacity ratio nor on the benefits.

4.3.3 Exact Methods

Besides heuristics, also exact approaches based on Mixed-Integer Programming were studied [MKK14; Sah+15; Bas+17; DKM18]. The ability to compute optimal solutions for a set of requests is within these works often used to analyze the performance of different objectives or the sensitivity of solutions under varying constraints of the studied model. The respective Mixed-Integer Programming formulations are all based on adaptations of the Multi-Commodity Flow formulation, whose integer variant is discussed in Section 5.1.

"Try a hard problem. You may not solve it, but you will prove something else."
– John E. Littlewood

5

Computational Complexity of the Virtual Network Embedding Problem

In this chapter, we initiate the systematic study of the computational complexity of the Virtual Network Embedding Problem under all combinations of node and edge mapping restrictions studied in this work. Specifically, we consider node mapping restrictions and node capacities, as well as routing policies, latency restrictions, and edge capacities (cf. Sections 2.2 and 2.4.3). We present a powerful 3-SAT reduction framework in Section 5.2, which is the base for nearly all hardness results presented henceforth. Specifically, we generally show the $\mathcal{NP}$-completeness of the decision variant of the VNEP, which in turn also implies the $\mathcal{NP}$-hardness of any optimization variant. In particular, the following is shown:

- We show the $\mathcal{NP}$-completeness of *six* different VNEP variants in Section 5.3. For example, we consider the variant only enforcing capacity constraints, but also one in which only node placement and latency restrictions must be obeyed *in the absence of capacity constraints.*
- We extend these results in Section 5.4 and show that the considered variants remain $\mathcal{NP}$-complete even when computing *approximate* embeddings, which may exceed latency or capacity constraints by certain factors.
- In Section 5.5 it is shown that the respective VNEP variants remain $\mathcal{NP}$-complete even when restricting substrate graphs to directed acyclic graphs (DAGs) and request graphs to planar, degree-bounded DAGs.
- In Section 5.6 we lastly discuss implications for the hardness of the VMP and accordingly for the fractional offline VNEP.

As we are proving $\mathcal{NP}$-completeness, the implications of our results are severe. Given the $\mathcal{NP}$-completeness of finding *any* feasible solution, finding an optimal solution *subject to any objective* is at least $\mathcal{NP}$-hard. Furthermore, unless $\mathcal{P} = \mathcal{NP}$ holds, the respective variants cannot be approximated to within *any* factor.

Table 5.1 summarizes our results and is to be read as follows. Any of the six rightmost columns represents a specific VNEP variant. The checkmark (✓) symbol indicates restrictions that are enforced, while the $\star$ symbol indicates restrictions which are not considered. Accordingly, considering a specific variant, the respective column should be read from top to bottom. For example, for $\langle$ **VE** | **-** $\rangle$, its $\mathcal{NP}$-completeness is shown in Theorem 5.10 while its inapproximability when relaxing edge capacity constraints is shown in Theorem 5.21. Lastly, all results also hold under the graph restrictions mentioned in the two bottom rows. Importantly, it is easy to see that enabling additional restrictions (marked as $\star$ in Table 5.1), does not change the results:

Lemma 5.1. A VNEP variant $\langle$ **A** | **C** $\rangle$ that encompasses all restrictions of $\langle$ **A'** | **C'** $\rangle$ is at least as hard as $\langle$ **A'** | **C'** $\rangle$.

Proof. The capacity constraints as well as the additional requirements were all formulated in such a fashion that any one of these can be disabled. Considering capacities and latencies, one may set the respective substrate capacities to ∞ and the latencies of edges to 0, respectively. For node placement and edge restrictions one may set the forbidden node and edge sets to the empty set. Hence, there exists a trivial reduction from $\langle \mathbf{A} \,|\, \mathbf{C} \rangle$ to $\langle \mathbf{A'} \,|\, \mathbf{C'} \rangle$ and the result follows. ■

VNEP variants	Identifier according to Def. 2.20	$\langle \mathbf{VE} \,|\, \mathbf{-} \rangle$	$\langle \mathbf{E} \,|\, \mathbf{N} \rangle$	$\langle \mathbf{V} \,|\, \mathbf{R} \rangle$	$\langle \mathbf{-} \,|\, \mathbf{NR} \rangle$	$\langle \mathbf{-} \,|\, \mathbf{NL} \rangle$	$\langle \mathbf{V} \,|\, \mathbf{L} \rangle$
	Node Capacities	✓	⋆	✓	⋆	⋆	✓
	Edge Capacities	✓	✓	⋆	⋆	⋆	⋆
	Node Placement Restrictions	⋆	✓	⋆	✓	✓	⋆
	Edge Routing Restrictions	⋆	⋆	✓	✓	⋆	⋆
	Latency Restrictions	⋆	⋆	⋆	⋆	✓	✓

Results							
	$\mathcal{NP}$-completeness and inapproximability under any objective	Thm. 5.10	Thm. 5.11	Thm. 5.12	Thm. 5.13	Thm. 5.13	Thm. 5.14
	$\mathcal{NP}$-completeness and inapproximability when increasing node capacities by a factor $\beta < 2$	Thm. 5.15	-	Thm. 5.15	-	-	Thm. 5.15
	Inapproximability when increasing edge capacities by a factor $\gamma \in \Theta(\log n / \log\log n)$ (unless $\mathcal{NP} \subseteq \mathcal{BP}\text{-}TIME(\bigcup_{d\geq 1} n^{d \log\log n})$)	Thm. 5.21	Thm. 5.19	-	-	-	-
	$\mathcal{NP}$-completeness and inapproximability when loosening latency bounds by a factor $\delta < 2$	-	-	-	-	Thm. 5.16	Thm. 5.16
	Results are preserved for acyclic substrates (except for Theorems 5.19 and 5.21)	Obs. 5.22					
	Results are preserved for acyclic, planar, degree-bounded requests	Thm. 5.25					

Table 5.1: Overview on obtained computational complexity results for the VNEP.

5.1 Integer Programming Formulation

We first give an Integer Programming (IP) formulation, which can be used to solve any of the considered decision VNEP variants. A similar formulation was proposed in [IR11]. Given the hardness results presented in this chapter and given that solving IPs lies in $\mathcal{NP}$ [Pap81], the IP may serve as an attractive approach to solve the respective variants in *exponential* time. Besides the practical application, the existence of our formulation (constructively) shows that the VNEP variants considered here are also all contained in $\mathcal{NP}$.

Our formulation naturally encompasses node placement and routing restrictions, while for latencies an additional constraint is introduced. The decision variable $x \in \{0, 1\}$ is used to indicate, whether the request graph G_r is embedded or not. By maximizing x, the IP decides whether a feasible embedding exists ($x = 1$) or whether no such embedding exists ($x = 0$). The mapping of virtual nodes is modeled

Integer Program 5.1: VNEP Decision Variant

$$\max x \tag{5.1}$$

$$\sum_{u \in V_S} y_i^u = x \qquad \forall i \in V_r \tag{5.2}$$

$$\sum_{u \in V_S \setminus V_S^{r,i}} y_i^u = 0 \qquad \forall i \in V_r \tag{5.3}$$

$$\sum_{(u,v) \in \delta^+(u)} z_{i,j}^{u,v} - \sum_{(v,u) \in \delta^-(u)} z_{i,j}^{v,u} = y_i^u - y_j^u \qquad \forall (i,j) \in E_r, u \in E_S \tag{5.4}$$

$$\sum_{(u,v) \in E_S \setminus E_S^{r,i,j}} z_{i,j}^{u,v} = 0 \qquad \forall (i,j) \in E_r \tag{5.5}$$

$$\sum_{i \in V_r} d_r(i) \cdot y_i^u \leq d_S(u) \qquad \forall u \in V_S \tag{5.6}$$

$$\sum_{(i,j) \in E_r} d_r(i,j) \cdot z_{i,j}^{u,v} \leq d_S(u,v) \qquad \forall (u,v) \in E_S \tag{5.7}$$

$$\sum_{(u,v) \in E_S} l_S(u,v) \cdot z_{i,j}^{u,v} \leq l_r(i,j) \qquad \forall (i,j) \in E_r \qquad ^\star \tag{5.8}$$

$$x \in \{0,1\} \tag{5.9}$$

$$y_i^u \in \{0,1\} \qquad \forall i \in V_r, u \in V_S \tag{5.10}$$

$$z_{i,j}^{u,v} \in \{0,1\} \qquad \forall (i,j) \in E_r, (u,v) \in E_S \tag{5.11}$$

using decision variables $y_i^u \in \{0,1\}$ for $i \in V_r$ and $u \in V_S$. If $y_i^u = 1$ holds, then the virtual node $i \in V_r$ is mapped on substrate node $u \in V_S$. Constraint 5.2 enforces that each virtual node is mapped to one substrate node, *if* the request is embedded ($x = 1$), while Constraint 5.3 excludes unsuitable substrate nodes.

For computing edge mappings the decision variables $z_{i,j}^{u,v} \in \{0,1\}$ for $(i,j) \in E_r$ and $(u,v) \in E_S$ are employed. If $z_{i,j}^{u,v} = 1$ holds, then the substrate edge (u,v) lies on the path $m_E(i,j)$. Constraints 5.4 and 5.5 embed virtual links as paths in the substrate, if the request is embedded. In particular, Constraint 5.4 constructs a unit flow for virtual edge $(i,j) \in E_r$ from the location $u \in V_S$ onto which i was mapped ($y_i^u = 1$) to the location $v \in V_S$ onto which j was mapped ($y_j^v = 1$), while Constraint 5.5 excludes unsuitable edges. Constraints 5.6 and 5.7 enforce that substrate capacities are obeyed. Lastly, Constraint 5.8 is only used when latencies are considered: it enforces that the sum of latencies along the embedding path of a virtual edge is smaller than the respective latency bound.

5.2 Reduction Framework

This section presents the main insight and contribution of this chapter, namely a generic reduction framework that allows to derive hardness results by slightly tailoring the proof for the individual problem variants. Our reduction framework relies on 3-SAT and we first introduce some notation. Afterwards we continue by constructing a (partial) VNEP instance, whose solution will indicate whether the 3-SAT formula is satisfiable.

5.2.1 3-SAT: Notation and Problem Statement

We denote by $\mathcal{L}_\phi = \{x_k\}_{k \in [N]}$ a set of $N \in \mathbb{N}$ literals and by $\mathcal{C}_\phi = \{\mathcal{C}_i\}_{i \in [M]}$ a set of $M \in \mathbb{N}$ clauses, in which literals may occur either positively or negated. The formula $\phi = \bigwedge_{\mathcal{C}_i \in \mathcal{C}_\phi} \mathcal{C}_i$ is a 3-SAT formula, iff. each clause $\mathcal{C}_i$ is the disjunction of at most 3 literals of $\mathcal{L}_\phi$. Denoting the truth values by F and T, 3-SAT

asks to determine whether an assignment $\alpha : \mathcal{L}_\phi \to \{\mathrm{F}, \mathrm{T}\}$ exists, such that ϕ is satisfied. 3-SAT is one of Karp's 21 $\mathcal{NP}$-complete problems:

Theorem 5.2 (Karp [Kar72])**.** Deciding 3-SAT is $\mathcal{NP}$-complete.

For reducing 3-SAT to VNEP, it is important that the clauses be ordered and we define the following:

Definition 5.3 (First Occurence of Literals)**.** We denote by $\mathcal{C} : \mathcal{L}_\phi \to [M]$ the function yielding the index of the clause in which a literal first occurs. Hence, if $\mathcal{C}(x_k) = i$ holds, then x_k is contained in $\mathcal{C}_i$ while not contained in $\mathcal{C}_{i'}$ for $i' \in [i-1]$. □

As we are interested in the satisfiability of a 3-SAT formula ϕ, we define the set of satisfying assignments *per clause*:

Definition 5.4 (Satisfying Assignments)**.** By $\mathcal{A}_i = \{a_{i,m} : \mathcal{L}_i \to \{\mathrm{F}, \mathrm{T}\} \mid a_{i,m} \text{ satisfies } \mathcal{C}_i\}$ the set of all possible assignments of truth values to the literals $\mathcal{L}_i$ of $\mathcal{C}_i$ satisfying $\mathcal{C}_i$ is denoted. Note that all elements of $\mathcal{A}_i$ are functions. □

Lastly, to abbreviate notation, we employ $\mathcal{L}_{i,j} = \mathcal{L}_i \cap \mathcal{L}_j$ to denote the intersection of the literal sets of $\mathcal{C}_i$ and $\mathcal{C}_j$.

5.2.2 General VNEP Instance Construction

For a given 3-SAT formula ϕ, we now construct a VNEP instance consisting of a substrate graph $G_{S(\phi)}$ and a request graph $G_{r(\phi)}$. The question whether the formula ϕ is satisfiable will eventually reduce to the question whether a feasible embedding of $G_{r(\phi)}$ on $G_{S(\phi)}$ exists. Figure 5.1 illustrates the construction described in the following.

Definition 5.5 (Request Graph $G_{r(\phi)}$)**.** For a given 3-SAT formula ϕ we define the request graph $G_{r(\phi)} = (V_{r(\phi)}, E_{r(\phi)})$ as follows. For each clause $\mathcal{C}_i \in \mathcal{C}_\phi$ a node v_i is introduced, i.e., $V_{r(\phi)} = \{v_i \,|\, \mathcal{C}_i \in \mathcal{C}_\phi\}$. An edge (v_i, v_j) is introduced if either the i-th clause $\mathcal{C}_i$ introduces a literal used in the j-th clause $\mathcal{C}_j$, or if $j = i+1$ holds. Accordingly, we set $E_{r(\phi)} = E^{\mathcal{L}}_{r(\phi)} \sqcup E^{\mathcal{N}}_{r(\phi)}$ with:

$$E^{\mathcal{L}}_{r(\phi)} = \{(v_i, v_j) \,|\, \exists x_k \in \mathcal{L}_{i,j} : \ \mathcal{C}(x_k) = i\}$$
$$E^{\mathcal{N}}_{r(\phi)} = \{(v_i, v_{i+1}) \,|\, i < M \wedge (v_i, v_{i+1}) \notin E^{\mathcal{L}}_{r(\phi)}\}$$

Note that edges pertaining to *literals* $E^{\mathcal{L}}_{r(\phi)}$ take precedence over edges $E^{\mathcal{N}}_{r(\phi)}$ created for *neighboring* request nodes. □

Note that the above definition only defined the request topology and did not specify demands or other restrictions, as these will be set in the respective reductions. Matching the general construction of the request graph, the substrate graph is analogously defined, albeit introducing up to 7 substrate nodes per clause: the respective substrate nodes will correspond to the *satisfying assignments* of the respective clause.

Definition 5.6 (Substrate Graph $G_{S(\phi)}$)**.** For a given 3-SAT formula ϕ the substrate graph $G_{S(\phi)} = (V_{S(\phi)}, E_{S(\phi)})$ is defined as follows. For each clause $\mathcal{C}_i \in \mathcal{C}_\phi$ and each potential assignment $a_{i,m} \in \mathcal{A}_i$ of truth values satisfying $\mathcal{C}_i$ a substrate node is used, i.e., $V_{S(\phi)} = \bigcup_{\mathcal{C}_i \in \mathcal{C}_\phi} \mathcal{A}_i$. Two substrate nodes $a_{i,m} \in V_{S(\phi)}$ and $a_{j,n} \in V_{S(\phi)}$ are connected in either of the following cases:

1. if $(v_i, v_j) \in E^{\mathcal{L}}_{r(\phi)}$ holds then the edge $(a_{i,m}, a_{j,n})$ is only introduced if the assignments $a_{i,m}$ and $a_{j,n}$ agree on the literals $\mathcal{L}_{i,j}$ contained in both clauses and
2. if $(v_i, v_j) \in E^{\mathcal{N}}_{r(\phi)}$ holds, then any edge $(a_{i,m}, a_{j,n})$ with $a_{i,m} \in \mathcal{A}_i$ and $a_{j,n} \in \mathcal{A}_j$ is introduced.

Accordingly, we set $E_{S(\phi)} = E_S^{\mathcal{L}} \sqcup E_S^{\mathcal{N}}$, with:

$$E_S^{\mathcal{L}} = \left\{ (a_{i,m}, a_{j,n}) \;\middle|\; (v_i, v_j) \in E_{r(\phi)}^{\mathcal{L}} \text{ and } a_{i,m}(x_l) = a_{j,n}(x_l) \text{ for } x_l \in \mathcal{L}_{i,j} \right\}$$
$$E_S^{\mathcal{N}} = \left\{ (a_{i,m}, a_{j,n}) \;\middle|\; (v_i, v_j) \in E_r^{\mathcal{N}} \right\}$$ □

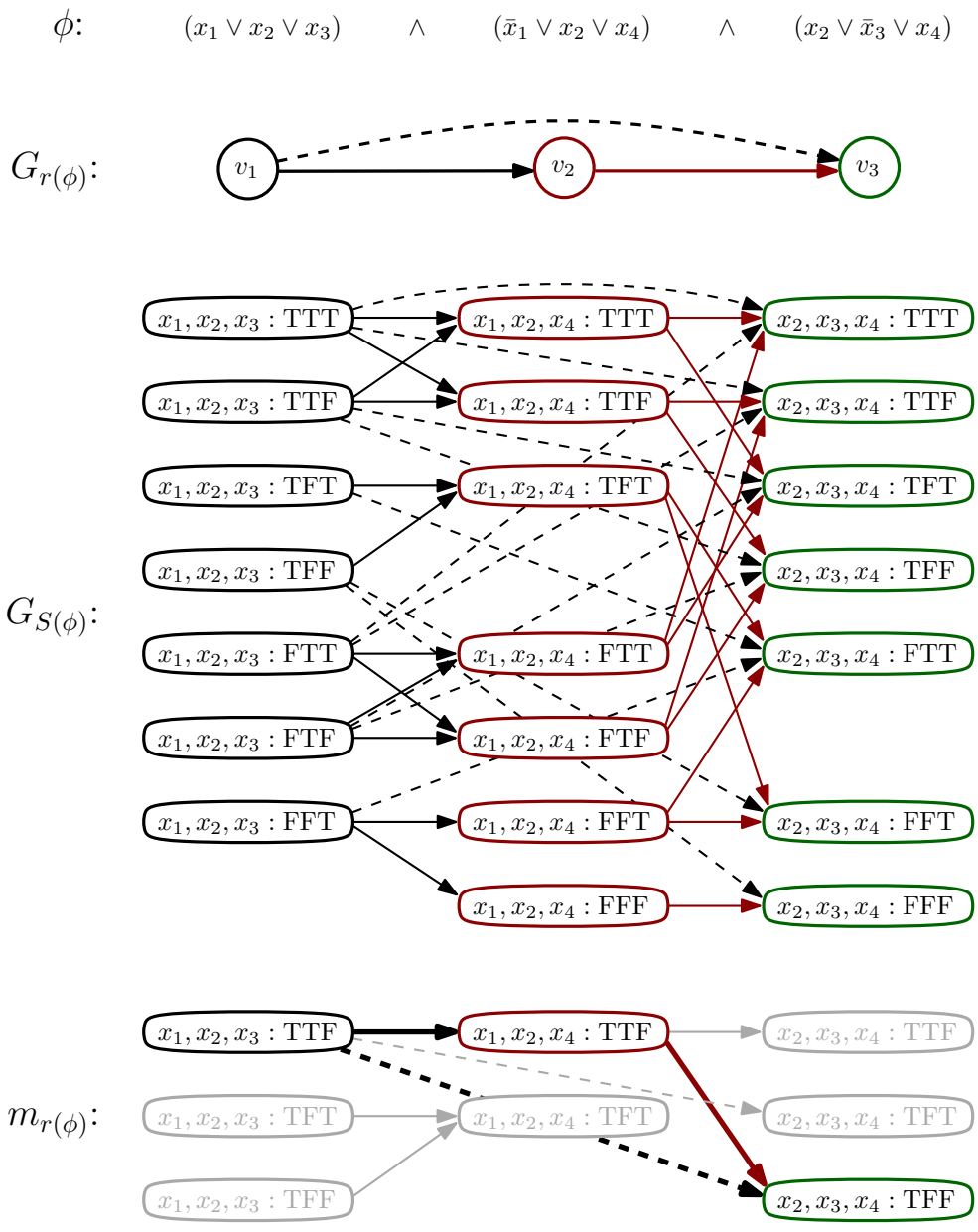


Figure 5.1: Visualization of the construction of request and substrate graphs for the 3-SAT formula ϕ (cf. Definitions 5.5 and 5.6). Additionally, a mapping $m_{r(\phi)}$ satisfying the conditions of Lemma 5.8 is shown. Accordingly, the formula ϕ is satisfied. Concretely, the mapping represents the assignment of truth values $x_1 = \mathrm{T}$, $x_2 = \mathrm{T}$, $x_3 = \mathrm{F}$, $x_4 = \mathrm{F}$.

5.2.3 The Base Lemma

In the following we give the base lemma, on which nearly all of our results are based. It shows the connection between the satisfiability of 3-SAT formulas and the existence of specific valid mappings introduced below.

Definition 5.7 (Satisfiable Valid Mappings $\mathcal{M}^{\text{SAT}}_{r(\phi)}$)**.** We denote by $\mathcal{M}^{\text{SAT}}_{r(\phi)}$ the set of valid mappings of $G_{r(\phi)}$ on $G_{S(\phi)}$, such that each virtual node v_i pertaining to the i-th clause is mapped on substrate nodes $\mathcal{A}_i$ and that each virtual edge is embedded using a single substrate edge, i.e.:

$$\mathcal{M}^{\text{SAT}}_{r(\phi)} = \left\{ m \in \mathcal{M}_r \;\middle|\; m_V(v_i) \in \mathcal{A}_i \text{ for } v_i \in V_{r(\phi)} \text{ and } |m_E(v_i, v_j)| = 1 \text{ for } (v_i, v_j) \in E_{r(\phi)} \right\} \quad \square$$

Lemma 5.8. A 3-SAT formula ϕ is satisfiable iff. $\mathcal{M}^{\text{SAT}}_{r(\phi)} \neq \emptyset$.

Proof. We first show that if ϕ is satisfiable, then a mapping $m_{r(\phi)} \in \mathcal{M}^{\text{SAT}}_{r(\phi)}$ must exist. Afterwards, we show that if such a mapping $m_{r(\phi)} \in \mathcal{M}^{\text{SAT}}_{r(\phi)}$ exists, then ϕ must be satisfiable.

Assume that ϕ is satisfiable and let $\alpha : \mathcal{L}_\phi \to \{\text{F}, \text{T}\}$ denote an assignment of truth values, such that α satisfies ϕ. We construct a mapping $m_{r(\phi)} = (m_V, m_E)$ for request r as follows. The virtual node $v_i \in V_{r(\phi)}$ corresponding to clause $\mathcal{C}_i$ is mapped onto the substrate node $a_{i,m} \in \mathcal{A}_i \subseteq V_{S(\phi)}$, iff. $a_{i,m}$ agrees with α on the assignment of truth values to the contained literals, i.e., $a_{i,m}(x_k) = \alpha(x_k)$ for $x_k \in \mathcal{C}_i$. As α satisfies ϕ, it satisfies each clause and hence $m_V(v_i) \in V_{S(\phi)}$ holds for all $\mathcal{C}_i \in \mathcal{C}_\phi$. The virtual edge $(v_i, v_j) \in E_{r(\phi)}$ is mapped via the direct edge between $m_V(v_i)$ and $m_V(v_j)$: if $(v_i, v_j) \in E^{\mathcal{L}}_{r(\phi)}$ holds, this edge $(m_V(v_i), m_V(v_j))$ must exist in $E^{\mathcal{L}}_S$, as $m_V(v_i) = a_{i,m}$ and $m_V(v_j) = a_{j,n}$ must agree by construction on the assignment of truth values for all literals. Secondly, if $(v_i, v_j) \in E^{\mathcal{N}}_{r(\phi)}$ holds, then the edge $(m_V(v_i), m_V(v_j))$ is clearly contained in $E^{\mathcal{N}}_S$. Hence, the constructed mapping $m_{r(\phi)}$ satisfies the conditions specified for $\mathcal{M}^{\text{SAT}}_{r(\phi)}$, such that $m_{r(\phi)} \in \mathcal{M}^{\text{SAT}}_{r(\phi)}$ holds, hence completing the first half of the proof.

We now show that if there exists a mapping $m_{r(\phi)} \in \mathcal{M}^{\text{SAT}}_{r(\phi)}$, then the formula ϕ is indeed satisfiable. We constructively recover an assignment of truth values $\alpha : \mathcal{L}_\phi \to \{\text{F}, \text{T}\}$ from the mapping $m_{r(\phi)}$ by iteratively extending the initially empty assignment. Concretely, we iterate over the mappings of the virtual nodes corresponding to the clauses of $\mathcal{C}_\phi$ one by one according to the precedence relation of the indices. By our assumption on the node mapping, $m_V(v_i) \in \mathcal{A}_i$ holds. Accordingly, as the substrate node $m_V(v_i)$ represents an assignment of truth values to the literals of clause $\mathcal{C}_i$, we extend α by setting $\alpha(x_k) \triangleq \big[m_V(v_i)\big](x_k)$ for all literals x_k contained in $\mathcal{C}_i$.

We first show that this extension is always valid in the sense that previously assigned truth values are never changed. To this end, assume that the clauses $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{i-1}$ were handled without any such violations. Hence the literals $\bigcup_{j<i} \mathcal{L}_j$ have been assigned truth values in the first $i-1$ iterations not contradicting previous assignments. When extending α by the mapping of $m_V(v_i)$ in the i-th iteration, there are two cases to consider. First, if none of the literals $\mathcal{L}_i$ were previously assigned a truth value, i.e., $\mathcal{L}_i \cap \bigcup_{j<i} \mathcal{L}_j = \emptyset$ holds, then the extension of α as described above cannot lead to a contradiction. Otherwise, if $\mathcal{L}_{i,\text{pre}} = \mathcal{L}_i \cap \bigcup_{j<i} \mathcal{L}_j \neq \emptyset$ holds, we show that extending α by $m_V(v_i) = a_{i,m}$ does not change the truth value of any literal x_k contained in $\mathcal{L}_{i,\text{pre}}$.

For the sake of contradiction, assume that $x_k \in \mathcal{C}_i$ is a literal, for which $\alpha(x_k)$ does not equal $\big[m_V(v_i)\big](x_k)$. As x_k was previously assigned a value, there must exist a clause $\mathcal{C}_j$ in which x_k was first used, such that $j < i$ holds. Let $m_V(v_i) = a_{i,m} \in \mathcal{A}_i$ and $m_V(v_j) = a_{j,n} \in \mathcal{A}_j$. By our assumption all edges are mapped using a single substrate edge, accordingly $m_E(v_i, v_j) = \langle (a_{j,n}, a_{i,m}) \rangle$ must hold. Hence, the substrate edge $(a_{j,n}, a_{i,m})$ must exist and must be contained in $E^{\mathcal{L}}_{r(\phi)}$ as $\mathcal{L}_{i,\text{pre}} \neq \emptyset$ holds. Since $(v_j, v_i) \in E^{\mathcal{L}}_{r(\phi)}$ holds, the respective substrate edge must be contained in $E^{\mathcal{L}}_S$ by definition. As $E^{\mathcal{L}}_S$ contains only edges *if* assignments agree with each other, $\big[m_V(v_j)\big](x_k) = a_{j,n}(x_k) = a_{i,m}(x_k) = \big[m_V(v_i)\big](x_k)$ is obtained. This contradicts our assumption that $\alpha(x_k) \neq \big[m_V(v_i)\big](x_k)$ holds. Hence, the extension of α is always valid.

By construction of the substrate graph $G_{S(\phi)}$, the node set $\mathcal{A}_i \subseteq V_{S(\phi)}$ contains only the assignments of truth values for the literals $\mathcal{L}_i$ of clause $\mathcal{C}_i \in \mathcal{C}_\phi$ that satisfy the respective clause. Hence, the constructed assignment α satisfies each clause and thus the formula ϕ, hence concluding the proof. ■

The above base lemma is the heart of our reduction framework for obtaining our $\mathcal{NP}$-completeness and $\mathcal{NP}$-hardness results. The following formalizes this observation.

Lemma 5.9. If the restrictions $\langle \mathbf{X} \,|\, \mathbf{Y} \rangle$ are sufficiently expressive to constrain feasible mappings of $G_{r(\phi)}$ to $G_{S(\phi)}$ to exactly the mappings of $\mathcal{M}^{\text{SAT}}_{r(\phi)}$ for any 3-SAT formula ϕ, then:

1. the decision VNEP $\langle \mathbf{X} \,|\, \mathbf{Y} \rangle$ is $\mathcal{NP}$-complete,
2. any optimization VNEP $\langle \mathbf{X} \,|\, \mathbf{Y} \rangle$ is $\mathcal{NP}$-hard and inapproximable (within any factor), unless $\mathcal{P} = \mathcal{NP}$ holds.

Proof. We outline the polynomial-time reduction from 3-SAT to the respective decision VNEP under constraints $\langle \mathbf{X} \,|\, \mathbf{Y} \rangle$. Considering any 3-SAT formula ϕ, the respective request and substrate graphs $G_{r(\phi)}$ and $G_{S(\phi)}$ are constructed. As the number of request nodes is bounded by the number of clauses and the number of substrate nodes is bounded by 7 times the number of clauses, the construction is polynomial. Under the assumption that the VNEP variant is sufficiently expressive to constrain the set of feasible embeddings to exactly $\mathcal{M}^{\text{SAT}}_{r(\phi)}$ (cf. Definition 5.7), the question of whether the formula ϕ is satisfiable reduces to the question of whether a feasible embedding of $G_{r(\phi)}$ on $G_{S(\phi)}$ exists (cf. Lemma 5.8). This reduction yields the $\mathcal{NP}$-hardness of the respective VNEP variant under restrictions $\langle \mathbf{X} \,|\, \mathbf{Y} \rangle$ as 3-SAT is $\mathcal{NP}$-complete. As the Integer Program presented in Section 5.1 can be used to decide VNEP under any restrictions $\langle \mathbf{X} \,|\, \mathbf{Y} \rangle$ and solving Integer Programs lies in $\mathcal{NP}$ [Pap81], the respective VNEP lies in $\mathcal{NP}$, hence showing the $\mathcal{NP}$-completeness of the respective VNEP variant. This completes the proof of the first statement.

The second statement holds as optimizing over the set of feasible solutions requires deciding whether a feasible solution exists in the first place. Accordingly, any optimization variant is also $\mathcal{NP}$-hard. Furthermore, any optimization variant over the same set of restrictions is also inapproximable to within any factor, unless $\mathcal{P} = \mathcal{NP}$. ■

5.3 Hardness of the VNEP

We employ our framework outlined in the previous section to derive a series of hardness results for the VNEP. In particular, we first show the $\mathcal{NP}$-completeness of the original VNEP variant $\langle \mathbf{VE} \,|\, \text{-} \rangle$ in the absence of additional restrictions. Given this result, we investigate the remaining other five combinations of node and edge mapping restrictions and show, among others, that also deciding $\langle \text{-} \,|\, \mathbf{LN} \rangle$ is $\mathcal{NP}$-complete. Hence, even when the physical network does not impose any resource constraints (i.e., nodes and links have infinite capacities), finding an embedding satisfying latency and node placement restrictions is $\mathcal{NP}$-complete. Again, it must be noted that adding further restrictions only renders the VNEP harder (cf. Lemma 5.1).

We first consider the most basic VNEP variant $\langle \mathbf{VE} \,|\, \text{-} \rangle$.

Theorem 5.10. VNEP $\langle \mathbf{VE} \,|\, \text{-} \rangle$ is $\mathcal{NP}$-complete and cannot be approximated under any objective (unless $\mathcal{P} = \mathcal{NP}$).

Proof. We show the statement via a polynomial-time reduction from 3-SAT according to Lemma 5.9. Specifically, we show how to constrain the set of feasible embeddings of $G_{r(\phi)}$ to $G_{S(\phi)}$ to exactly $\mathcal{M}^{\text{SAT}}_{r(\phi)}$ (cf. Definition 5.7), such that each virtual node corresponding to the i-th clause is mapped on a substrate node corresponding to the i-th clause, albeit embedding all virtual edges using a single substrate edge.

To enforce the node mapping property of $\mathcal{M}^{\text{SAT}}_{r(\phi)}$, namely that each virtual node $v_i \in V_{r(\phi)}$ must be mapped on nodes in $\mathcal{A}_i$, we set unit substrate node capacities and demands:

$$\begin{aligned} d_{S(\phi)}(a_{i,m}) &= 1 \quad \forall\, \mathcal{C}_i \in \mathcal{C}_\phi, a_{i,m} \in \mathcal{A}_i \\ d_{r(\phi)}(v_i) &= 1 \quad \forall\, v_i \in V_{r(\phi)} \end{aligned}$$

Given these demands and capacities, we note the following. Firstly, as the substrate nodes have a capacity of 1 and the virtual nodes have a demand of 1, at most one virtual node may be placed on any substrate node. Secondly, the request graph $G_{r(\phi)}$ contains edges (v_i, v_{i+1}) for $i \in [N-1]$, while the substrate is acyclic with edges always being oriented towards substrate nodes pertaining to clauses with a higher index.

Hence, if a virtual node $v_i \in V_{r(\phi)}$ is mapped on a substrate node $a_{k,o} \in V_{S(\phi)}$ corresponding to the k-th clause, then the virtual node v_{i+1} corresponding to the next clause must be mapped on a substrate node $a_{k',o'} \in V_{S(\phi)}$ with $k' \geq k+1$. If $v_i \in V_{r(\phi)}$ was to be mapped on a substrate node $a_{k,o} \in V_{S(\phi)}$ with $k > i$, then at least the last node v_M of the chain $\langle v_i, v_{i+1}, \ldots, v_M \rangle$ cannot be suitably mapped. By the same argument, $v_i \in V_{r(\phi)}$ cannot be mapped on a substrate node $a_{k,o}$ with $k < i$, as then at least the first node v_1 could not be mapped feasibly on any substrate node. Therefore, any feasible embedding must map the node $v_i \in V_{r(\phi)}$ on a substrate node $a_{i,m} \in V_{S(\phi)}$, therefore restricting the node mappings exactly as in the definition of the set $\mathcal{M}^{\text{SAT}}_{r(\phi)}$.

To enforce the edge mapping restrictions specified for $\mathcal{M}^{\text{SAT}}_{r(\phi)}$, namely that each virtual edge is embedded to exactly one substrate edge, the following non-unit edge capacities and demands are set for some λ with $0 < \lambda < 1/|\mathcal{C}_\phi|$:

$$\begin{aligned} d_{S(\phi)}(e) &= 1 + \lambda \cdot j \quad \forall\, \mathcal{C}_j \in \mathcal{C}_\phi, e \in \delta^-(\mathcal{A}_j) \\ d_{r(\phi)}(e) &= 1 + \lambda \cdot j \quad \forall\, v_j \in V_{r(\phi)}, e \in \delta^-(v_j) \end{aligned}$$

Accordingly, the capacity of a substrate edge and the demand of a virtual edge is determined by the index of the clause its head is representing: the higher the clause-index of the edge's head, the higher the capacity. Given these capacities and demands, we now show that any virtual edge must be mapped on exactly one substrate edge. To this end, assume for the sake of contradiction that the virtual edge $(v_i, v_j) \in E_{r(\phi)}$ is not mapped on a single substrate edge. As v_i must be mapped on some node $a_{i,m} \in \mathcal{A}_i$ and v_j must be mapped on some node $a_{j,n} \in \mathcal{A}_j$, and as both the request and the substrate are directed acyclic graphs, the mapping of edge (v_i, v_j) must route through at least one intermediate node. Denote by $a_{k,l} \in \mathcal{A}_k$ for $i < k < j$ the first intermediate node lying on the path along which the edge (v_i, v_j) is routed. By construction, the capacity of the substrate edge $(a_{i,m}, a_{k,l})$ is $1 + \lambda \cdot k$. However, as $k < j$ holds and the edge (v_i, v_j) has a demand of $1 + \lambda \cdot j$, the edge (v_i, v_j) cannot be routed via $a_{k,l}$. Thus, the only feasible edges for embedding the respective virtual edges are the direct connections between any two substrate nodes.

Therefore, according to the above capacities and demands, any feasible embedding m satisfies the conditions of $\mathcal{M}^{\text{SAT}}_{r(\phi)}$ and is hence contained in it. On the other hand, the above imposed capacities do not constrain the set of feasible embeddings any further: each mapping $m \in \mathcal{M}^{\text{SAT}}_{r(\phi)}$ is a feasible embedding according to the above capacities. Thus, as the set of feasible embeddings equals $\mathcal{M}^{\text{SAT}}_{r(\phi)}$, the result follows by Lemma 5.9. ■

In the following the above proof is adapted to other settings. All the proofs purely rely on constraining the set of feasible embeddings to the set $\mathcal{M}^{\text{SAT}}_{r(\phi)}$ and the application of Lemma 5.9. Hence, in the following we only prove that the set of feasible embeddings equals $\mathcal{M}^{\text{SAT}}_{r(\phi)}$.

Theorem 5.11. VNEP $\langle \mathbf{E} \,|\, \mathbf{N} \rangle$ is $\mathcal{NP}$-complete and cannot be approximated under any objective (unless $\mathcal{P} = \mathcal{NP}$).

Proof. In this setting node placement restrictions and substrate edge capacities are enforced. Employing the node placement restrictions, we can force the mapping of virtual node $v_i \in V_{r(\phi)}$ onto substrate nodes $\mathcal{A}_i$ by setting $\overline{V}_{S(\phi)}^{r(\phi),v_i} = V_{S(\phi)} \setminus \mathcal{A}_i$ for all $v_i \in V_{r(\phi)}$. Utilizing the same edge capacities and demands as in the proof of Theorem 5.10, virtual edges have to be mapped using a single edge, as intermediate nodes do not support the respective demand. Hence, the result follows. ■

Theorem 5.12. VNEP $\langle \mathbf{V} \,|\, \mathbf{R} \rangle$ is $\mathcal{NP}$-complete and cannot be approximated under any objective (unless $\mathcal{P} = \mathcal{NP}$).

Proof. In this setting node capacities and routing restrictions must be obeyed. We employ the same node capacities as in the proof of Theorem 5.10, such that a virtual node $v_i \in V_{r(\phi)}$ may only be mapped on a substrate node contained in $\mathcal{A}_i$. Furthermore, routing restrictions are set to only allow direct edges. Specifically, for the virtual edge $(v_i, v_j) \in E_{r(\phi)}$ the set of forbidden edges $\overline{E}_{S(\phi)}^{r(\phi),v_i,v_j}$ is set to $E_{S(\phi)} \setminus (\mathcal{A}_i \times \mathcal{A}_j)$. As the node demands and capacities enforce the node mapping restrictions of $\mathcal{M}_{r(\phi)}^{\text{SAT}}$ and the routing restrictions enforce that each virtual edge must be mapped on exactly a single substrate edge, the result follows. ■

Theorem 5.13. The VNEP is $\mathcal{NP}$-complete under restrictions $\langle \text{-} \,|\, \mathbf{NR} \rangle$ and $\langle \text{-} \,|\, \mathbf{NL} \rangle$ and cannot be approximated under any objective (unless $\mathcal{P} = \mathcal{NP}$).

Proof. Both VNEP variants do not consider capacities. Allowing for node placement restrictions, the node mapping restrictions of $\mathcal{M}_{r(\phi)}^{\text{SAT}}$ (cf. Definition 5.7) are easily safeguarded (cf. proof of Theorem 5.11). By employing the same routing restrictions as in the proof of Theorem 5.12 the result follows directly for the case $\langle \text{-} \,|\, \mathbf{NR} \rangle$.

For $\langle \text{-} \,|\, \mathbf{NL} \rangle$, latency restrictions can be employed to enforce that virtual edges span at most a single substrate edge. Concretely, we set unit substrate edge latencies and unit virtual edge latency bounds:

$$\begin{aligned} l_{S(\phi)}(e) &= 1 \quad \forall e \in E_{S(\phi)} \\ l_{r(\phi)}(e) &= 1 \quad \forall e \in E_{r(\phi)} \end{aligned}$$

Accordingly, each virtual edge can only be realized by using at most a single substrate edge. Furthermore, given the node mapping restrictions, the virtual nodes cannot be mapped onto the same substrate node. Hence, the set of feasible embeddings equals exactly $\mathcal{M}_{r(\phi)}^{\text{SAT}}$ and the result also follows for $\langle \text{-} \,|\, \mathbf{NL} \rangle$. ■

Theorem 5.14. VNEP $\langle \mathbf{V} \,|\, \mathbf{L} \rangle$ is $\mathcal{NP}$-complete and cannot be approximated under any objective (unless $\mathcal{P} = \mathcal{NP}$).

Proof. This variant enforces node capacities while also obeying latency restrictions. Again, utilizing unit node capacities and demands as in the proof of Theorem 5.10, the node mapping restrictions of $\mathcal{M}_{r(\phi)}^{\text{SAT}}$ are safeguarded. By employing unit substrate latencies and unit latency restrictions for each virtual edge as in the proof of Theorem 5.13, also the edge mapping restrictions of $\mathcal{M}_{r(\phi)}^{\text{SAT}}$ are enforced, thereby yielding the result. ■

5.4 Hardness of Computing Approximate Embeddings

Given the hardness results presented in Section 5.3, the question arises to which extent the hardness can be overcome when only computing approximate embeddings (cf. Definitions 2.18 and 2.19), i.e., embeddings that may violate capacity or latency constraints by certain factors. Based on the proofs presented in Section 5.3, we first derive hardness results for computing β-approximate embeddings (allowing node capacity

violations) and δ-approximate embeddings (allowing latency violations). For γ-approximate embeddings, a reduction from a variant of the edge-disjoint paths problem will be given.

Theorem 5.15. For $\langle \mathbf{VE} \,|\, \text{-} \,\rangle$, $\langle \mathbf{V} \,|\, \mathbf{R} \,\rangle$, and $\langle \mathbf{V} \,|\, \mathbf{L} \,\rangle$ finding a β-approximate embedding is $\mathcal{NP}$-complete as well as inapproximable under any objective (unless $\mathcal{P} = \mathcal{NP}$) for any $\beta < 2$.

Proof. The $\mathcal{NP}$-completeness proofs under the restrictions $\langle \mathbf{VE} \,|\, \text{-} \,\rangle$, $\langle \mathbf{V} \,|\, \mathbf{R} \,\rangle$, and $\langle \mathbf{V} \,|\, \mathbf{L} \,\rangle$ relied all on the same argument to show that the virtual node $v_i \in V_{r(\phi)}$ must be mapped on any substrate node contained in $\mathcal{A}_i \subseteq V_{S(\phi)}$ (cf. Theorems 5.10, 5.12, and 5.14): due to the unit substrate node capacities and the unit node demands only a single virtual node can mapped on a substrate node and accordingly the chain of virtual nodes $v_1, v_2, \ldots, v_M$ must be embedded linearly using substrate node sets $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_M$. Notably, the remaining parts of the proofs were only using that $v_i \in V_{r(\phi)}$ must be mapped on any node in $\mathcal{A}_i \subseteq V_{S(\phi)}$.

Clearly, in the above proofs one may increase the substrate node capacities by any factor $1 \leq \beta < 2$ without any changes in the respective proofs: even if substrate nodes were to have a capacity of β, only a single virtual node can be hosted by any of the substrate nodes. As asking for a β-approximate embeddings is equivalent to finding a non-approximate embedding while increasing all node capacities by the factor β, any algorithm for the VNEP returning β-approximate embeddings for $\beta < 2$ could still be used to decide 3-SAT and therefore even deciding whether a β-approximate embedding exists remains $\mathcal{NP}$-complete and inapproximable for any $\beta < 2$. ■

Proving the $\mathcal{NP}$-completeness of δ-approximate embeddings goes along the same lines:

Theorem 5.16. For $\langle \text{-} \,|\, \mathbf{NL} \,\rangle$ and $\langle \mathbf{V} \,|\, \mathbf{L} \,\rangle$ finding an δ-approximate embedding is $\mathcal{NP}$-complete as well as inapproximable under any objective (unless $\mathcal{P} = \mathcal{NP}$) for any $\delta < 2$.

Proof. The proofs of Theorems 5.13 and 5.14 relied on the fact that due to the latency constraints each virtual edge must be mapped on a single substrate edge. As the latencies of substrate edges are uniformly set to 1 and all latency bounds are 1 as well, computing a δ-approximate embedding for $\delta < 2$ implies that each virtual edge can still only be mapped on a single substrate edge. Analogously to the proof of Theorem 5.15, the respective VNEP variants remain $\mathcal{NP}$-complete and inapproximable even when only asking to decide whether a δ-approximate embedding exists. ■

For γ-approximate embeddings we employ an inapproximability result of a variant of the edge-disjoint paths problem:

Definition 5.17 (DirEDPwC [Chu+07])**.** The Directed Edge-Disjoint Paths Problem with Congestion (DirEDPwC) is defined as follows. Given is a directed graph $G = (V, E)$ together with a set of $l \in \mathbb{N}$ source-sink pairs (commodities) $\{(s_k, t_k)\}_{k \in [l]}$, $s_k, t_k \in V$, and a constant $c \in \mathbb{N}$. The task is to find a path P_k connecting s_k to t_k for each $k \in [l]$, such that at most c many paths are routed via any edge $e \in E$. □

It is well-known that the edge-disjoint paths problem on directed graphs is hard to approximate:

Theorem 5.18 (Chuzhoy et al. [Chu+07])**.** Let $n = |V|$ denote the number of nodes. Given an instance of the DirEDPwC, it is impossible to distinguish between the following two cases in polynomial-time, unless $\mathcal{NP} \subseteq \mathcal{BP}\text{-}TIME(\bigcup_{d \geq 1} n^{d \log \log n})$:

1. A solution with congestion $c = 1$ exists .
2. No solution with congestion $c \in \Theta(\log n / \log \log n)$ exists.

Above, $\mathcal{BP}$-*TIME*$(f(n))$ denotes the class of problems solvable by probabilistic Turing machines in time $f(n)$ with bounded error-probability [AB09]. Given the approximation-preserving reductions presented above, the inapproximability of DirEDPwC carries over to the respective VNEP variants.

To apply this result for the DirEDPwC in the context of γ-approximate embeddings, reductions from DirEDPwC to the VNEP variants $\langle \mathbf{E} \,|\, \mathbf{N} \rangle$ and $\langle \mathbf{VE} \,|\, \text{-} \rangle$ are used. Importantly, these reductions are preserved when relaxing edge capacities, such that γ-approximate embeddings translate to solutions of the DirEDPwC with a congestion increase by a factor γ.

Theorem 5.19. Solving the γ-approximate decision VNEP under restrictions $\langle \mathbf{E} \,|\, \mathbf{N} \rangle$ is not possible in polynomial-time for $\gamma \in \Theta(\log n / \log\log n)$ with $n = |V_S|$, unless $\mathcal{NP} \subseteq \mathcal{BP}$-*TIME*$(\bigcup_{d\geq 1} n^{d \log\log n})$ holds. Again, this result naturally also extends to optimization variants in the setting $\langle \mathbf{E} \,|\, \mathbf{N} \rangle$.

Proof. We first give a reduction from DirEDPwC to the decision VNEP under restrictions $\langle \mathbf{E} \,|\, \mathbf{N} \rangle$. Given a DirEDPwC instance on the graph $G = (V, E)$ with commodities $(s_k, t_k)_{k\in[l]}$ and congestion c, an equivalent VNEP instance consisting of the substrate graph $G_{S(\text{edp})} = (V_{S(\text{edp})}, E_{S(\text{edp})})$ and the request graph $G_{r(\text{edp})} = (V_{r(\text{edp})}, E_{r(\text{edp})})$ is constructed. Firstly, the substrate is set to equal the original graph, i.e., $G_{S(\text{edp})} = G$, and the request graph $G_{r(\text{edp})} = (V_{r(\text{edp})}, E_{r(\text{edp})})$ is defined as follows. $V_{r(\text{edp})}$ consists of two virtual nodes per commodity, $V_{r(\text{edp})} = \{i_k, j_k | k \in [l]\}$, and we set $E_{r(\text{edp})} = \{(i_k, j_k) | k \in [l]\}$. Let $\sigma : V_{r(\text{edp})} \to V_{S(\text{edp})}$ denote the function indicating the original substrate node locations of the respective commodities. Specifically, $\sigma(i_k) = s_k$ and $\sigma(j_k) = t_k$ holds for all $k \in [l]$. We employ node mapping requirements to force the mapping of virtual nodes i_k and j_k to the locations of the respective commodities s_k and t_k by setting $\overline{V}_{S(\text{edp})}^{r(\text{edp}),i} = V \setminus \{\sigma(i)\}$ for $i \in V_{r(\text{edp})}$. Setting edge capacities in the substrate to c (the congestion value) and virtual edge demands to 1, deciding the respective VNEP problem is equivalent to deciding DirEDPwC: any embedding $m_{r(\text{edp})} = (m_V, m_E)$ of the VNEP instance induces a solution to the DirEDPwC instance by setting $P_k = m_E(i_k, j_k)$ for $k \in [l]$ and vice versa. Importantly, note that when considering γ-approximate solutions for the above VNEP instance, a respective DirEDPwC solution of congestion $\gamma \cdot c$ can be obtained in exactly the same way.

Given that γ-approximate embeddings yield an increase in the congestion of the DirEDPwC solution by the same factor, we can now prove the impossibility to decide whether γ-approximate embeddings exist for $\gamma \in \Theta(\log|V_S| / \log\log|V_S|)$. For the sake of contradiction assume that there exists a polynomial-time algorithm for the γ-approximate VNEP for some $\gamma \in o(\log|V_S| / \log\log|V_S|)$ and that $\mathcal{NP} \not\subseteq \mathcal{BP}$-*TIME*$(\bigcup_{d\geq 1} n^{d \log\log n})$ holds. Now, consider a DirEDPwC instance with $c = 1$ and assume that a feasible solution of congestion 1 exists. Clearly, the DirEDPwC solution of congestion $c = 1$ induces a feasible embedding (without exceeding edge capacities) of the respective VNEP instance. Accordingly, the γ-approximate VNEP algorithm must return a γ-approximate embedding in polynomial-time. The γ-approximate solution can then be used to recover a solution to the original DirEDPwC instance having congestion $c = \gamma \in o(\log|V_S| / \log\log|V_S|) = o(\log n / \log\log n)$, where $n = |V| = |V_S|$ denotes the number of nodes of the original DirEDPwC instance. However, under the assumption that $\mathcal{NP} \not\subseteq \mathcal{BP}$-*TIME*$(\bigcup_{d\geq 1} n^{d \log\log n})$ holds, the construction of this DirEDPwC instance of congestion $c = \gamma$ contradicts Theorem 5.18, which states that such an approximate solution cannot (always) be found in polynomial-time. Hence, finding γ-approximate in polynomial-time is in general impossible for some $\gamma \in \Theta(\log n / \log\log n)$. ∎

We will now derive a similar result for the impossibility of solving the γ-approximate VNEP variant under capacity restrictions $\langle \mathbf{VE} \,|\, \text{-} \rangle$. In contrast to the variant $\langle \mathbf{E} \,|\, \mathbf{N} \rangle$, the reduction from DirEDPwC is slightly more involved as the endpoints of the commodities have to be fixed using node capacities only. To concisely state the result, we introduce the notion of core nodes. Specifically, considering a graph $G = (V, E)$ we use $V^c = \{u \in V \,|\, |\delta^+(u) \cup \delta^-(u)| \geq 2\}$ to denote the *core* nodes having more than a single incoming or outgoing edge. We may assume that any DirEDPwC instance does only consist of core nodes, as the following lemma shows.

Lemma 5.20. Any DirEDPwC instance on the graph $G = (V, E)$ can be reduced to an equivalent instance on a graph $G_p = (V_p, E_p)$, such that V_p contains only core nodes, i.e., $V_p^c = V_p$ holds.

Proof. Given the initial graph $G = (V, E)$, the idea is that any non-core node $u \in V \setminus V^c$ does not offer any routing decisions and can therefore be removed from the instance. Specifically, consider a node $u \in V$ only having a single incoming edge e^- or only one outgoing edge e^+: the edge e^- will only be used by commodities whose target was mapped on u while the edge e^+ will only be used by commodities whose source was mapped on u. Furthermore, these commodities have to use the respective edge. When the number of commodities having to use such an edge lies above the congestion c, then clearly no solution can exist, while otherwise the respective node u can be removed, while reassigning the source or the target of the respective commodities from u to the node incident to u. By iterating this process, equivalent DirEDPwC instances are obtained until no non-core nodes exist anymore and the graph $G_p = (V_p, E_p)$ is obtained with $V_p = V_p^c$. ■

Theorem 5.21. Solving the γ-approximate decision VNEP under restrictions $\langle \mathbf{VE} \,|\, \text{-} \rangle$ is not possible in polynomial-time for $\gamma \in \Theta(\log n / \log\log n)$ with $n = |V_S^c|$, unless $\mathcal{NP} \subseteq \mathcal{BP}\text{-}TIME(\bigcup_{d \geq 1} n^{d \log\log n})$ holds. Again, this result naturally also extends to optimization variants in the setting $\langle \mathbf{VE} \,|\, \text{-} \rangle$.

Proof. We essentially use the same argumentation as in the proof of Theorem 5.19, but employ a different reduction from DirEDPwC to VNEP to fix the source and target mappings of commodities. Our reduction again takes as input a DirEDPwC instance on the graph $G = (V, E)$ with l commodities $(s_k, t_k)_{k \in [l]}$ and outputs an equivalent VNEP instance consisting of a substrate $G_{S(\text{edp})}$ and a request $G_{r(\text{edp})}$.

To construct the VNEP instance, we first introduce some additional notation. We denote by $O^+ : V \to \mathbb{N}$ and $O^- : V \to \mathbb{N}$ the function that counts the number of times a node $v \in V$ occurs as source or as sink in the commodities: $O^+(v) = |\{k \in [l] \mid s_k = v\}|$ and $O^-(v) = |\{k \in [l] \mid t_k = v\}|$. To construct the substrate graph $G_{S(\text{edp})}$ the original graph G is extended as follows. For each node $v \in V$, we add $O^+(v)$ many copies $V_{S(\text{edp})}^{+,v} = \{v_1^+, v_2^+, \ldots, v_{O^+(v)}^+\}$ and $O^-(v)$ many copies $V_{S(\text{edp})}^{-,v} = \{v_1^-, v_2^-, \ldots, v_{O^-(v)}^-\}$. For each copy v_k^+ an edge (v_k^+, v) is added to $E_{S(\text{edp})}$ while for any sink node v_k^- the edge (v, v_k^-) is introduced. Additionally using the function $U : V \to [|V|]$ assigning each vertex a unique numeric identifier, we define substrate node capacities according to the following rule: all original nodes, $v \in V_{S(\text{edp})} \cap V$, are assigned a capacity of 0, while setting $d_S(v_k^+) = U(v)$ and $d_S(v_l^-) = U(v)$ for $v \in V$ and $k \in [O^+(v)]$ and $l \in [O^-(v)]$.

The request graph $G_{r(\text{edp})} = (V_{r(\text{edp})}, E_{r(\text{edp})})$ is constructed as in the proof of Theorem 5.19: $V_{r(\text{edp})}$ consists of two virtual nodes per commodity, $V_{r(\text{edp})} = \{i_k, j_k | k \in [l]\}$, and we set $E_{r(\text{edp})} = \{(i_k, j_k) | k \in [l]\}$. Using again $\sigma : V_{r(\text{edp})} \to V$ to denote the function indicating the original substrate node location of the respective virtual nodes, the demand of virtual nodes is set to match the capacity of the respective endpoints they are to be mapped on: $d_{r(\text{edp})}(i) = U(\sigma(i))$ is set for $i \in V_{r(\text{edp})}$. Given these capacities, we now prove that for any feasible embedding $m_{r(\text{edp})} = (m_V, m_E)$ any virtual node $i \in V_{r(\text{edp})}$ must indeed be mapped on a copy corresponding to $\sigma(i)$. Specifically, we show that for $i_k \in V_{r(\text{edp})}$ and $j_k \in V_{r(\text{edp})}$, corresponding to the source and the target of commodity k, $m_V(i_k) \in V_{S(\text{edp})}^{+,\sigma(i_k)}$ and $m_V(j_k) \in V_{S(\text{edp})}^{+,\sigma(j_k)}$ must hold.

We show the above statement by using an inductive argument and start off by first considering only the mappings of virtual nodes which shall be mapped on the (unique) substrate node u of highest value $U(u)$. Clearly, any virtual node $i_k \in V_{r(\text{edp})}$ with $\sigma(i_k) = u$ or any virtual node j_k with $\sigma(j_k) = u$ can only be mapped on substrate nodes of capacity $U(u)$. As only the substrate nodes contained in $V_{S(\text{edp})}^{+,u} \cup V_{S(\text{edp})}^{-,u}$ offer this capacity, $m_V(i_k) \in V_{S(\text{edp})}^{+,u} \cup V_{S(\text{edp})}^{-,u}$ and $m_V(j_k) \in V_{S(\text{edp})}^{+,u} \cup V_{S(\text{edp})}^{-,u}$ must hold. Furthermore, as the virtual node i_k induces a flow towards j_k and not both virtual nodes can be placed on the same copy of u, the virtual node must be mapped on a node contained in $V_{S(\text{edp})}^{+,u}$ as only these have outgoing edges. Analogously, the virtual node j_k must be mapped on a node contained in $V_{S(\text{edp})}^{-,u}$, as only these nodes have an incoming edge. As the demanded capacities and the substrate capacities of node u are equal, all

virtual nodes can be mapped, while having to use all available capacity on the copies of node u. The above proof scheme can now be iteratively applied for substrate nodes offering the second most capacity etc., thus concluding our proof that $m_V(i_k) \in V_{S(\text{edp})}^{+,\sigma(i_k)}$ and $m_V(j_k) \in V_{S(\text{edp})}^{+,\sigma(j_k)}$ holds for any $i_k, j_k \in V_{r(\text{edp})}$.

We now argue shortly that by the above construction a solution to the DirEDPwC instance exists if and only if a solution to the respective VNEP instance exists. To this end, we set the edge capacities to the congestion value c for original edges contained in E and to 1 to any newly introduced edges towards copies of substrate nodes. Clearly, any embedding of the virtual edge $(i_k, j_k) \in E_r$ must start at $m_V(i_k) \in V_{S(\text{edp})}^{+,\sigma(i_k)}$ and therefore must traverse $\sigma(i_k) = s_k$ as the second node. Analogously, the virtual edge must end in $m_V(j_k) \in V_{S(\text{edp})}^{-,\sigma(j_k)}$ and therefore must traverse $\sigma(j_k) = t_k$ as second last node. By the same argument any solution to the DirEDPwC instance yields a solution to the VNEP: considering commodity $k \in [l]$ first suitable (unused) source and target nodes $v^+ \in V_{S(\text{edp})}^{+,s_k}$ and $v^- \in V_{S(\text{edp})}^{-,t_k}$ are chosen and afterwards the embedding is extended by $m_V(i_k) = v^+$, $m_V(j_k) = v^-$, and $m_E(i_k, j_k) = \langle (v^+, s_k), P_k, (t_k, v^-) \rangle$.

Similar to the proof of Theorem 5.19 it remains to show that the above equivalence is preserved when considering γ-approximate embeddings, which may exceed edge capacities by the factor γ. This indeed remains true, as the above argument that the endpoints of each virtual edge $(i_k, j_k) \in E_{r(\text{edp})}$ must correctly be mapped on any of the respective source nodes $V_{S(\text{edp})}^{+,s_k}$ and sink nodes $V_{S(\text{edp})}^{-,t_k}$ only relied on (i) the chosen *node* capacities and demands and (ii) the orientation of the respective incident edges. Thus, a γ-approximate embedding increases the congestion of the corresponding DirEDPwC solution by a factor of exactly γ.

Lastly, using the same argument as in the proof of Theorem 5.19, assuming the existence of a DirEDPwC instance having a solution with congestion $c = 1$, it is impossible for any algorithm to find a γ-approximate embedding with $\gamma \in o(\log n / \log\log n)$ with $n = |V|$, unless $\mathcal{NP} \subseteq \mathcal{BP}\text{-}TIME(\bigcup_{d \geq 1} n^{d \log\log n})$ holds. As we may assume by Lemma 5.20 that the original graph G only consisted of core nodes, we have $n = |V| = |V_S^c|$, yielding the result. ■

5.5 Hardness under Graph Restrictions

All of our above hardness results are based on a reduction from 3-SAT (except for Theorems 5.19 and 5.21), yielding a specific directed acyclic substrate graph $G_{S(\phi)}$ and a specific directed acyclic request graph $G_{r(\phi)}$ and we note the following:

Observation 5.22. Theorems 5.10 - 5.16 still hold when restricting the request and the substrate to acyclic graphs.

Given the hardness of the VNEP and as for example Virtual Clusters (an undirected star network) can be optimally embedded in polynomial time (cf. Chapter 8), one might ask whether the hardness is preserved when restricting request graphs further.

In this section, we derive the result that the VNEP variants considered remain $\mathcal{NP}$-complete when request graphs are planar and degree-bounded. To obtain that the decision VNEP is $\mathcal{NP}$-complete for planar and degree-bounded request graphs, we consider reductions from a *planar* variant of 3-SAT, namely Clause-Linked Planar 3-Bounded 3-SAT (CP3B-3-SAT):

Theorem 5.23 (CP3B-3-SAT is $\mathcal{NP}$-complete [Fel+95])**.**
Deciding the satisfiability of a 3-SAT formula remains $\mathcal{NP}$-complete under the following additional restrictions.

1. The graph $G_\phi = (V_\phi, E_\phi)$ is planar, where

$$V_\phi = \{v_i \mid C_i \in \mathcal{C}_\phi\} \cup \{u_k \mid x_k \in \mathcal{L}_\phi\}$$
$$E_\phi = \{\{v_i, u_k\} \mid \mathcal{C}_i \in \mathcal{C}_\phi, x_k \in x_k : x_k \in \mathcal{C}_i\} \cup \{v_i, v_{i+1} \mid i \in [M-1]\} \cup \{\{v_M, v_1\}\}.$$

2. Each clause $\mathcal{C}_i \in \mathcal{C}_\phi$ contains at most three literals.

3. Each variable $x_k \in \mathcal{L}_\phi$ occurs in exactly three clauses.

An example of a graph G_ϕ pertaining to a formula ϕ is depicted in Figure 5.2.

The following lemma connects the CP3B-3-SAT formula ϕ with the corresponding request graph $G_{r(\phi)}$.

Lemma 5.24. Given a CP3B-3-SAT formula ϕ, the following holds for the request graph $G_{r(\phi)}$ (cf. Definition 5.5):

1. The request graph $G_{r(\phi)}$ is planar.

2. The node-degree of $G_{r(\phi)}$ is bounded by 8.

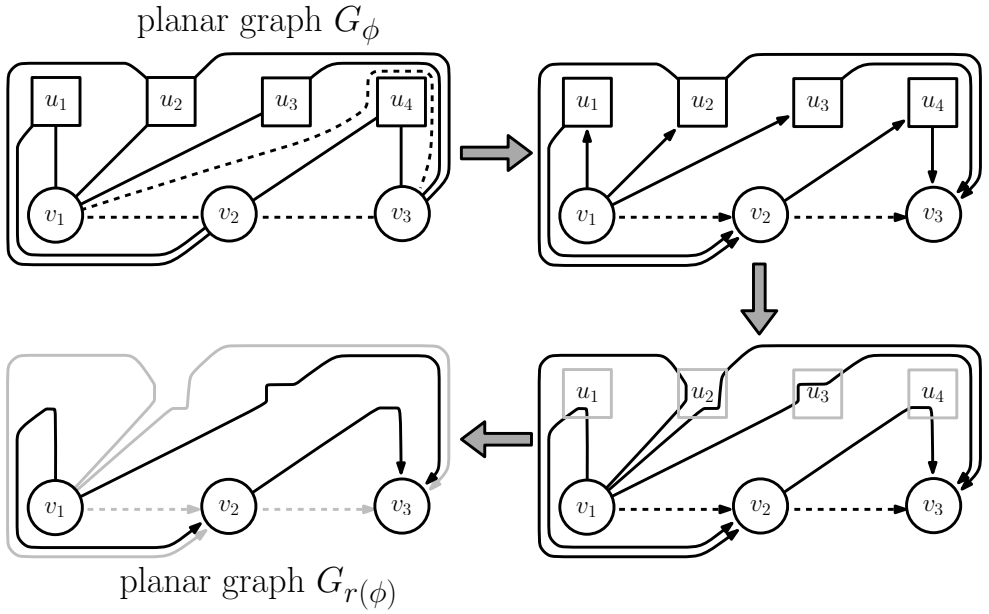


Figure 5.2: Depicted is the transformation process of a planar graph G_ϕ corresponding to a CP3B-3-SAT formula ϕ (cf. Theorem 5.23) to the planar graph $G_{r(\phi)}$. Concretely, the example formula of Figure 5.1 is revisited, i.e., $\phi = C_1 \wedge C_2 \wedge C_3$, with $C_1 = x_1 \vee x_2 \vee x_3$, $C_2 = \bar{x}_1 \vee x_2 \vee x_4$, and $C_3 = x_2 \vee \bar{x}_3 \vee x_4$. The solid edges connect clause nodes and literal nodes, while the dashed edges link the clause nodes.
In the first step the edge $\{v_3, v_1\}$ is removed and all remaining edges are directed: edges between clause nodes and literal nodes are oriented towards literal nodes iff. the literal occurs in the respective clause for the first time (according to the ordering of clause nodes) and edges between clause nodes are oriented towards the clause with the higher index. In the second step, each outgoing edge of a literal node is joined with the single incoming edge (duplicating it when necessary), hence allowing to remove the literal nodes. In the last step, duplicate edges are removed, yielding the request graph $G_{r(\phi)}$. Each step of this transformation process safeguards the graph's planarity.

Proof. We consider an arbitrary CP3B-3-SAT formula ϕ to which the conditions of Theorem 5.23 apply. We first show that the corresponding request graph $G_{r(\phi)}$ is planar by detailing a transformation process leading from the planar graph G_ϕ to $G_{r(\phi)}$ while preserving planarity (see Figure 5.2).

Starting with the undirected graph G_ϕ, first the edge $\{v_M, v_1\}$ is removed and all edges are oriented: an edge between a clause node and a variable node is oriented from a clause node to a literal node iff. the literal occurs in the respective clause for the first time according to the clauses' ordering. The edges between clause nodes are always oriented towards the clause with the higher index.

Given this directed graph, the literal nodes are now removed by joining the single incoming edge of the literal nodes with *each* outgoing edge of the corresponding literal node. In particular, considering the literal node u_2 of Figure 5.2, the single incoming edge (v_1, u_2) is joined with the outgoing edges (u_2, v_2) and (u_2, v_3) to obtain the edges (v_1, v_2) and (v_1, v_3), respectively. As the duplication of the single incoming edge cannot refute planarity and all incoming and outgoing edges connect to the same node, the planarity of the graph is preserved in this step. Lastly, duplicate edges are removed to obtain the graph $G_{r(\phi)}$, which is, in turn, planar.

It remains to show, that the request graph $G_{r(\phi)}$ corresponding to ϕ exhibits a bounded node-degree of 8 (in the undirected interpretation of the graph $G_{r(\phi)}$). To see this, we note the following. Based on the second and third conditions of CP3B-3-SAT (cf. Theorem 5.23) each clause node is connected with at most two different other clauses via each literal node it is connected to, yielding at most 6 neighbors. Furthermore, any clause is directly connected to at most two further clause nodes via the edges between clause nodes. ■

Given the above, we easily derive the following theorem:

Theorem 5.25. Theorems 5.10 - 5.16 hold when restricting the request graphs to be planar and degree 8-bounded. Theorems 5.19 and 5.21 hold for planar and degree 1-bounded graphs.

Proof. Our $\mathcal{NP}$-completeness proofs in Section 5.3 and Section 5.4 (except for Theorems 5.19 and 5.21) relied solely on the reduction from 3-SAT to the decision VNEP using the base Lemma 5.8. As formulas of CP3B-3-SAT are a strict subset of the 3-SAT formulas, the base Lemma 5.8 is still applicable for CP3B-3-SAT formulas. However, due to the structure of CP3B-3-SAT formulas, the corresponding requests in the reductions are planar and exhibit a node-degree bound of 8 by Lemma 5.24. Hence, solving the VNEP is $\mathcal{NP}$-complete, even when restricting the requests to planar and / or degree-bounded ones. Lastly, we note that Theorems 5.19 and 5.21 hold for planar and degree 1-bounded (unconnected) request graphs, as in the reduction only such requests were considered. ■

5.6 Hardness of the VMP and the Fractional Offline VNEP

Before concluding this chapter, based on the above results for the various VNEP variants we derive some insights into the hardness of the Valid Mapping Problem (VMP), which does only ask for finding valid mappings (cf. Definition 2.10). As the VMP differs from the VNEP solely by not respecting cumulative node and edge allocations (cf. Observation 2.11), solving the decision variant of the VMP is equivalent to solving the decision variant of the VNEP in the settings $\langle$ **-** | **NL** $\rangle$ and $\langle$ **-** | **NR** $\rangle$. Accordingly, the results obtained for these VNEP variants carry over to the VMP and we note the following:

Corollary 5.26 (Hardness of the VMP)**.** When disregarding node and edge capacities, the VNEP reduces to the VMP. Accordingly, considering the decision variant of the VMP, i.e., the task to decide whether there exists any valid mapping, all results obtained for the VNEP settings $\langle$ **-** | **NL** $\rangle$ and $\langle$ **-** | **NR** $\rangle$ carry over to the VMP in these respective settings. In particular, the following holds:

(1) The decision variant of the VMP is $\mathcal{NP}$-complete and all optimization variants, specifically also the one presented in Definition 2.10, are inapproximable (unless $\mathcal{P}=\mathcal{NP}$) when considering the settings $\langle$ **-** | **NL** $\rangle$ and $\langle$ **-** | **NR** $\rangle$ (by Theorem 5.13).

(2) For the variant $\langle$ **-** | **NL** $\rangle$ the above still holds when loosening latency bounds by a factor $\delta < 2$ (by Theorem 5.16).

(3) The above mentioned results are pertained even for acyclic substrate graphs and for acyclic, planar, degree-bounded requests (by Observation 5.22 and Theorem 5.25).

As the fractional offline VNEP (cf. Definition 2.7) asks for returning a convex combination of *valid mappings*, the same result also applies for the fractional offline VNEP and we only state the $\mathcal{NP}$-hardness:

Corollary 5.27. The following holds for the profit and cost variants of the fractional offline VNEP.

(1) The fractional offline VNEP is $\mathcal{NP}$-hard and inapproximable (unless $\mathcal{P}=\mathcal{NP}$) when considering the settings $\langle$ **-** | **NL** $\rangle$ and $\langle$ **-** | **NR** $\rangle$.

(2) For $\langle$ **-** | **NL** $\rangle$ the above still holds when loosening latency bounds by a factor $\delta < 2$.

(3) The above mentioned results are pertained even for acyclic substrate graphs and for acyclic, planar, degree-bounded requests (by Observation 5.22 and Theorem 5.25).

5.7 Summary and Novelty of Contributions

Above, a comprehensive set of hardness results for the various settings of the VNEP was given. Our results are negative in nature: we show that the problem variants are $\mathcal{NP}$-complete and hence inapproximable (unless $\mathcal{P}=\mathcal{NP}$) and that this holds true even for restricted classes of request graphs.

Our results complement and extend the known $\mathcal{NP}$-hardness results obtained by Amaldi et al. in [Ama+16] (see Theorem 4.1). In particular, we have proven $\mathcal{NP}$-completeness and do not only consider the profit variant. Furthermore, we study the latency and edge routing restrictions which were not considered by Amaldi et al. and show that each combination of node and edge restriction renders the VNEP hard.

We believe that our results are of great importance for future work on embedding virtual networks. For example, our results on the variant enforcing node placement and latency restrictions are of specific interest for Service Function Chaining. Surprisingly, the respective problem is hard even when not considering any capacity constraints. Furthermore, the structural result that the VNEP and the VMP are hard even for planar request graphs is of importance when studying approximations, as this proves that no polynomial-time approximations can exist for this class of request graphs (unless $\mathcal{P}=\mathcal{NP}$ holds), hence, motivating the study of parametrized algorithms for general request graphs in Chapters 6 and 7.

"Its running time is polynomial; also, it is very insensitive to the number of constraints in the following sense: we do not need to list the faces in advance, but only need a subroutine which recognizes feasibility of a vector and if it is infeasible then computes a hyperplane separating it from P. Searching for such a hyperplane is another combinatorial optimization problem which is often much easier to solve. So this method, the Ellipsoid Method, reduces one combinatorial optimization problem to a second one. "

– Grötschel, Lovász, Schrijver [GLS81]

6

XP-Algorithms for the Fractional Offline VNEP and the VMP

In this chapter we present different algorithms based on Linear Programming to solve the fractional offline VNEP (cf. Definition 2.7). As will be shown in Chapter 7, the ability to solve the fractional offline VNEP will allow us to obtain the first approximation algorithms for both the profit and the cost variant of the offline VNEP. However, as shown in Section 5.6, computing solutions to the fractional offline VNEP is $\mathcal{NP}$-hard and inapproximable for the VNEP settings $\langle$ **-** $|$ **NL** $\rangle$ and $\langle$ **-** $|$ **NR** $\rangle$. Thus, even though the fractional variant poses a major simplification of the offline VNEP, as the respective algorithms do not have to decide on *a single* valid mapping anymore, solving the fractional VNEP is itself a challenging task.

In this chapter LP formulations for the restrictions $\langle$ **VE** $|$ **NR** $\rangle$ and $\langle$ **VE** $|$ **NRL** $\rangle$ are given. Notably, these LP formulations can of course also be used in settings with fewer restrictions, solving the fractional offline VNEP also in the other potential settings. To motivate our approach, we shortly discuss the connection to approximations and then give a detailed overview of the contents of this chapter.

Enumerative LP Formulations and Overview of Randomized Rounding. To motivate the Linear Programming approaches provided in this chapter, we shortly discuss two general LP formulations for the fractional offline VNEP and discuss how randomized rounding will then be applied in general to obtain approximate solutions for the VNEP.

The Formulations 6.1 and 6.2 for the profit and the cost variant are based on the explicit enumeration of all valid mappings for each request. In particular, a 'decision' variable $f_r^k \geq 0$ is used for each mapping $m_r^k \in \mathcal{M}_r$ of each request $r \in \mathcal{R}$, to indicate the *extent* to which mapping m_r^k is used to embed $r \in \mathcal{R}$. The Constraints 6.3 and 6.7 safeguard the (fractional) feasibility of cumulative allocations (cf. Definition 2.7). Constraint 6.2 of the profit formulation ensures that the cumulative mapping weights of each request do not exceed 1, while the Constraint 6.6 of the cost formulation ensures that each request is *fully* (fractionally) embedded. The respective objectives naturally follow the definition of the

LP Formulation 6.1: Enumerative Formulation for the Profit Fractional Offline VNEP

$$\max \sum_{r \in \mathcal{R}, m_r^k \in \mathcal{M}_r} f_r^k \cdot b_r \tag{6.1}$$

$$\sum_{m_r^k \in \mathcal{M}_r} f_r^k \leq 1 \qquad \forall r \in \mathcal{R} \tag{6.2}$$

$$\sum_{r \in \mathcal{R}, m_r^k \in \mathcal{M}_r} f_r^k \cdot A(m_r^k, x) \leq d_S(x) \qquad \forall x \in G_S \tag{6.3}$$

$$f_r^k \geq 0 \qquad r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \tag{6.4}$$

LP Formulation 6.2: Enumerative Formulation for the Cost Fractional Offline VNEP

$$\min \sum_{x \in G_S} c_S(x) \cdot \sum_{r \in \mathcal{R}, m_r^k \in \mathcal{M}_r} f_r^k \cdot A(m_r, x) \tag{6.5}$$

$$\sum_{m_r^k \in \mathcal{M}_r} f_r^k = 1 \qquad \forall r \in \mathcal{R} \tag{6.6}$$

$$\sum_{r \in \mathcal{R}, m_r^k \in \mathcal{M}_r} f_r^k \cdot A(m_r^k, x) \leq d_S(x) \qquad \forall x \in G_S \tag{6.7}$$

$$f_r^k \geq 0 \qquad r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \tag{6.8}$$

respective fractional offline VNEP variants. Disregarding that the set of all valid mappings $\mathcal{M}_r$ per request can in general not be constructed in polynomial-time, solutions to the respective LPs yield the set $\mathcal{D}_r = \{(f_r^k, m_r^k) \mid m_r^k \in \mathcal{M}_r, f_r^k \geq 0\}$ of convex combinations[1] of mappings for each request $r \in \mathcal{R}$. The *randomized rounding* of a solution then generally works as follows. For each request $r \in \mathcal{R}$, the mapping m_r^k is selected with probability f_r^k (allowing for rejecting r with probability $1 - \sum_k f_r^k$ in the profit case). As shown in Chapter 7 rounding the solutions in this way, the first approximations for the different VNEP variants are obtained.

Overview. Solving the enumerative Formulations 6.1 and 6.2 is (at first) prohibitive due to the exponential number of valid mappings and in this chapter we present several solution approaches to solve the fractional offline VNEP. Common to all of these formulations is the observation that the *structure of the request graphs* greatly influences the ability to solve the fractional VNEP in polynomial-time. Concretely, we show in Section 6.1 that the classic Multi-Commodity Flow (MCF) formulation is sufficient to solve the fractional VNEP in the setting $\langle \mathbf{VE} \,|\, \mathbf{NR} \rangle$, *when the request graphs are trees*, but fails as soon as the request graphs contain cycles.

Analyzing the root causes for the MCF's failure to capture the structural properties of request graphs with cycles, a first extended LP formulation is derived, which is applicable only for cactus request graphs and increases in size by a factor $\mathcal{O}(|V_S|)$ compared to the MCF formulation. Later on, in Section 6.3, the idea of the LP formulation for cactus graphs is extended to arbitrary request graphs. However, this comes at a distinct increase in formulation size: the size grows exponentially in a novel graph number, namely the *extraction width*. While this yields the first XP-algorithms (parametrized in the *extraction width*), we show that computing the optimal extraction width is $\mathcal{NP}$-hard. Accordingly, in Section 6.4 we turn towards the well-studied notion of tree decompositions and derive XP-algorithms for the fractional VNEP which are parametrized in the *treewidth* of the request graphs. Compared to the extraction width approach, the treewidth based approach has several distinct advantages:

1. Since tree decompositions have been extensively studied over the course of the last decade, efficient algorithms are readily available to compute the (optimal) tree decompositions for small graphs.

2. Using the notion of tree decompositions, we present a dynamic programming approach to directly solve the VMP. This algorithm can then be readily applied as separation oracle to solve the LP Formulations 6.1 and 6.2 via *column generation*. Here, the advantage lies in the observation that, from a practical point of view, the column generation LP can speed up LP computations by magnitudes in practice (validated in the evaluation presented in Section 7.6.3).

3. Lastly, while enforcing latency constraints using classic LP formulation is arguably not possible, it is facilitated by an extension of the presented dynamic programming algorithm to solve the VMP.

[1] While for the cost variant the set $\mathcal{D}_r$ will indeed be a convex combination of mappings, such that $\sum_k f_r^k = 1$ holds, this is generally not true for the profit variant. However, we still use the term convex combination in this case, as the respective combination could be extended by the 'empty mapping' having weight $1 - \sum_k f_r^k$.

Therefore, only using the tree decomposition based dynamic programming algorithm, the fractional offline VNEP can be solved for the setting $\langle \mathbf{VE} \,|\, \mathbf{NRL} \rangle$.

As the (fractional) offline VNEP is considered throughout this section, we will omit the term offline from now on. Furthermore, our LP formulations will always incorporate node placement and edge routing restrictions as discussed in Section 2.4.

6.1 The Classic Multi-Commodity Formulation and Its Limits

In this section, we revisit the Multi-Commodity Flow (MCF) formulation for solving the VNEP (see Formulation 6.3), which is widely used [CRB09; MKK14]. We first show the positive result that the formulation is sufficiently strong to compute solutions to the fractional VNEP when requests are *trees*. Subsequently, we show that the formulation fails to allow for the decomposition of *cyclic* request graphs into convex combinations of valid mappings.

6.1.1 The Multi-Commodity Formulation

We explain the formulation by considering its integer variant. Notably, the IP Formulation 5.1 is generalized here as several requests are now considered. Otherwise, we employ the same variables and notation. The variable $x_r \in \{0,1\}$ indicates whether request $r \in \mathcal{R}$ is embedded or not. The variable $y^u_{r,i} \in \{0,1\}$ indicates whether virtual node $i \in V_r$ is mapped on substrate node u. Similarly, the flow variable $z^{u,v}_{r,i,j} \in \{0,1\}$ indicates whether the substrate edge $(u,v) \in E_S$ is part of the path of the virtual edge $(i,j) \in E_r$. The variable $a^x_r \geq 0$ denotes the cumulative allocations that the embedding of request r induces on resource $x \in G_S$.

By Constraints 6.9 and Constraint 6.10, virtual nodes are only mapped on suitable substrate nodes when $x_r = 1$ holds. Constraint 6.11 induces an unsplittable unit flow for each virtual edge $(i,j) \in E_r$ from the substrate location onto which i was mapped to the substrate location onto which j was mapped. By Constraint 6.12 the mapping of virtual edges may only consist of *allowed* substrate edges. Constraints 6.13 and 6.14 compute the cumulative allocations while Constraint 6.15 enforces that resource capacities are respected. Applying the objective $\max \sum_{r \in \mathcal{R}} b_r \cdot x_r$ the profit variant is obtained. Setting $\min \sum_{x \in G_S, r \in \mathcal{R}} c_S(x) \cdot a^x_r$ and enforcing $x_r = 1$ for all requests $r \in \mathcal{R}$ the cost variant is obtained.

Formulation 6.3: Multi-Commodity Flow Base Formulation for the VNEP

$$\sum_{u \in V_S^{r,i}} y^u_{r,i} = x_r \qquad \forall r \in \mathcal{R}, i \in V_r \tag{6.9}$$

$$y^u_{r,i} = 0 \qquad \forall r \in \mathcal{R}, i \in V_r, u \in V_S \setminus V_S^{r,i} \tag{6.10}$$

$$\sum_{(u,v) \in \delta^+(u)} z^{u,v}_{r,i,j} - \sum_{(v,u) \in \delta^-(u)} z^{v,u}_{r,i,j} = y^u_{r,i} - y^u_{r,j} \qquad \forall r \in \mathcal{R}, (i,j) \in E_r, u \in V_S \tag{6.11}$$

$$z^{u,v}_{r,i,j} = 0 \qquad \forall r \in \mathcal{R}, (i,j) \in E_r, (u,v) \in E_S \setminus E_S^{r,i,j} \tag{6.12}$$

$$\sum_{(i,j) \in E_r} d_r(i,j) \cdot z^{u,v}_{r,i,j} = a^{u,v}_r \qquad \forall r \in \mathcal{R}, (u,v) \in E_S \tag{6.13}$$

$$\sum_{i \in V_r} d_r(i) \cdot y^u_{r,i} = a^u_r \qquad \forall r \in \mathcal{R}, u \in V_S \tag{6.14}$$

$$\sum_{r \in \mathcal{R}} a^x_r \leq d_S(x) \qquad \forall x \in G_S \tag{6.15}$$

The LP formulation is obtained by relaxing the domain of the above introduced binary variables to $[0, 1]$. The following lemma states that whenever a virtual node $i \in V_r$ is (fractionally) mapped on a certain substrate node, suitable mappings for all incident edges and their endpoints can be found.

Lemma 6.1 (Local Connectivity Property of the MCF Formulation)**.** Consider a fractional solution $(x_r, \vec{y}_r, \vec{z}_r, \vec{a}_r)$ to the LP Formulation 6.3 for request $r \in \mathcal{R}$. If $y^u_{r,i} > 0$ holds for $i \in V_r$ and $u \in V_S^{r,i}$, then for incoming edges $(k, i) \in E_r$ and outgoing edges $(i, j) \in E_r$ there exist substrate paths $P^{v,u}_{r,k,i}$ and $P^{u,w}_{r,i,j}$, such that:

(1) $P^{v,u}_{r,k,i}$ is a path from v to u, such that $y^v_{r,k} > 0$ and $z^e_{r,k,i} > 0$ holds for $e \in P^{v,u}_{r,k,i}$.

(2) $P^{u,w}_{r,i,j}$ is a path from u to w, such that $y^w_{r,j} > 0$ and $z^e_{r,i,j} > 0$ holds for $e \in P^{u,w}_{r,i,j}$.

The respective paths $P^{v,u}_{r,k,i}$ and $P^{u,w}_{r,i,j}$ can be found in time $\mathcal{O}(|E_S|)$ by a simple graph search.

Proof. Fix any substrate node $u \in V_S$ for which $y^u_{r,i} > 0$ holds. We first consider the outgoing edge $(i, j) \in E_r$. By Constraint 6.9, $\sum_{u \in V_S^{r,i}} y^u_{r,i} = \sum_{v \in V_S^{r,j}} y^v_{r,j}$ holds. Hence, the virtual node $j \in V_r$ must be mapped also at least with value $y^u_{r,i}$. If j is also partially mapped on u, i.e., if $y^u_{r,j} > 0$ holds, then the result follows directly, as u connects to u using (and allowing) the empty path $P^{u,u}_{r,i,j} = \langle\rangle$. If, on the other hand, $y^u_{r,j} = 0$ holds, then Constraint 6.11 induces a flow of value $y^u_{r,i}$ at substrate node u with respect to the commodity $z_{r,i,j}$. As the right hand side of Constraint 6.11 may only attain negative values at nodes $w \in V_S^{r,j}$ for which $y^w_{r,j} > 0$ holds, the flow (of commodity $z_{r,i,j}$) emitted at node u must eventually reach a node $w \in V_S^{r,j}$ with $y^w_{r,j} > 0$ and hence the result follows for any outgoing edge $(i, j) \in E_r$. Note that the corresponding path $P^{u,w}_{r,i,j}$ can be constructed in time $\mathcal{O}(|E_S|)$ by a simple breadth-first search, which only considers edges $(u', v') \in E_S$ for which $z^{u',v'}_{r,i,j} > 0$ holds.

The argument for incoming edges $(k, i) \in E_r$ is the same and the respective paths $P^{v,u}_{r,k,i}$ can be recovered by breadth-first searches traversing substrate edges $(u, v) \in E_S$ in their opposite direction when $z^{u,v}_{r,k,i} > 0$ holds. ■

6.1.2 Decomposing Solutions to the MCF Formulation

Given the connectivity property of Lemma 6.1, we argue how solutions to the LP relaxation of the MCF formulation can be decomposed into convex combinations $\mathcal{D}_r = \{(f^k_r, m^k_r) \mid m^k_r \in \mathcal{M}_r, f^k_r \geq 0\}$ *as long as the request graphs are trees.* The ideas presented henceforth will also apply for the decomposition of our novel LP formulations presented in Section 6.2 and 6.3.

We naturally apply the idea of Ford and Fulkerson [FF10] for decomposing $s - t$ flows into paths to our setting. Given a LP solution $(x_r, \vec{y}_r, \vec{z}_r, \vec{a}_r)$ for request $r \in \mathcal{R}$, we need to find a valid mapping $m_r = (m_V, m_E) \in \mathcal{M}_r$ which is *covered* by the embedding variables. Concretely, letting $\mathcal{V}(m_r) = \{y^{m_V(i)}_{r,i} \mid i \in V_r\} \cup \{z^{u,v}_{r,i,j} \mid (i, j) \in E_r, (u, v) \in m_E(i, j)\}$ denote all the LP variables involved under mapping m_r, we say that the mapping m_r is covered by the LP solution iff. $f_r = \min \mathcal{V} > 0$ holds. Accordingly, the mapping m_r of weight f_r can be *extracted* by reducing the variables in $\mathcal{V}$ by f_r while adding (f_r, m_r) to the set of convex combinations $\mathcal{D}_r$. Importantly, after the extraction, the now adapted LP solution is still feasible and hence the extraction process can be repeated. To find a mapping in the first place, the mapping of nodes and edges has to be done in some order. We refer to this order as the extraction order:

Definition 6.2 (Extraction Order $G^{\mathcal{X}}_r$)**.** Given a virtual network $G_r = (V_r, E_r)$, we refer to any rooted graph $G^{\mathcal{X}}_r = (V_r, E^{\mathcal{X}}_r, s_r)$ as an *extraction order*, if the following holds:

(1) $G^{\mathcal{X}}_r$ is a directed acyclic graph, s.t. each node is reachable from the root $s_r \in V_r$, and

(2) $E^{\mathcal{X}}_r$ is obtained from E_r by (potentially) reversing the orientation of some edges.

Algorithm 6.4: Decomposition algorithm of MCF solutions for Tree Requests

Input : Tree request $r \in \mathcal{R}$ together with a solution $(x_r, \vec{y}_r, \vec{z}_r, \vec{a}_r)$ for Formulation 6.3
Extraction order $G_r^{\mathcal{X}} = (V_r, E_r^{\mathcal{X}}, s_r)$
Output: Convex combination $\mathcal{D}_r = \{D_r^k = (f_r^k, m_r^k)\}_k$ of valid mappings

set $\mathcal{D}_r \leftarrow \emptyset$ **and** $k \leftarrow 1$
while $x_r > 0$ **do**
  set $m_r^k \leftarrow (m_V, m_E) \leftarrow (\emptyset, \emptyset)$
  set $\mathcal{Q} \leftarrow \{s_r\}$
  choose $u \in V_S^{r,s_r}$ **with** $y_{r,s_r}^u > 0$ **and set** $m_V(s_r) \leftarrow u$
  while $|\mathcal{Q}| > 0$ **do**
    choose $i \in \mathcal{Q}$ **and set** $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{i\}$
    foreach $(i,j) \in \delta^+_{E_r^{\mathcal{X}}}(i)$ **do**
      if $(i,j) = \overrightarrow{E}_r(i,j)$ **then**
        compute path $P_{r,i,j}^{u,v}$ from $m_V(i) = u$ to $v \in V_S^{r,j}$ according to Lemma 6.1
        such that $y_{r,j}^v > 0$ and $z_{r,i,j}^{u',v'} > 0$ hold for $(u',v') \in P_{r,i,j}^{u,v}$
        set $m_V(j) \leftarrow v$ **and** $m_E(i,j) \leftarrow P_{r,i,j}^{u,v}$
      else
        compute path $P_{r,j,i}^{v,u}$ from $v \in V_S^{r,j}$ to $m_V(i) = u$ according to Lemma 6.1
        such that $y_{r,j}^v > 0$ and $z_{r,j,i}^{u',v'} > 0$ hold for $(u',v') \in P_{r,j,i}^{v,u}$
        set $m_V(j) \leftarrow v$ **and** $m_E(\overrightarrow{E}_r(i,j)) \leftarrow P_{r,j,i}^{u,v}$
      set $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{j\}$
  set $\mathcal{V}_k \leftarrow \{x_r\} \cup \{y_{r,i}^{m_V(i)} | i \in V_r\} \cup \{z_{r,i,j}^{u,v} | (i,j) \in E_r, (u,v) \in m_E(i,j)\}$
  set $f_r^k \leftarrow \min \mathcal{V}_k$
  set $v \leftarrow v - f_r^k$ **for all** $v \in \mathcal{V}_k$ **and set** $a_r^x \leftarrow a_r^x - f_r^k \cdot A(m_r^k, x)$ **for all** $x \in G_S$
  add $D_r^k = (f_r^k, m_r^k)$ to $\mathcal{D}_r$ **and set** $k \leftarrow k+1$
return $\mathcal{D}_r$

We denote by $\overrightarrow{E}_r : E_r^{\mathcal{X}} \to E_r$ the function yielding the edge's original orientation and by $\overrightarrow{E}_r^{\mathcal{X}} : E_r \to E_r^{\mathcal{X}}$ its inverse. We write $\delta^+_{\mathcal{X}}(i) = \{(i,j) \in E_r^{\mathcal{X}}\}$ and $\delta^-_{\mathcal{X}}(i) = \{(j,i) \in E_r^{\mathcal{X}}\}$ to denote the outgoing and incoming edges with respect to the edge set $E_r^{\mathcal{X}}$. □

Given the extraction order $G_r^{\mathcal{X}}$, the extraction process works by first choosing a suitable mapping location for the root s_r. Given this location, Lemma 6.1 is applied to obtain mappings for all outgoing edges of s_r (according to $E_r^{\mathcal{X}}$) *together* with their heads. Continuing to apply Lemma 6.1 for each of the newly mapped nodes, a complete mapping in which all virtual nodes and edges are mapped on suitable substrate nodes and edges is constructed.

Algorithm 6.4 formalizes the decomposition scheme to extract convex combinations of valid mappings from solutions to Formulation 6.3. The algorithm extracts mappings m_r^k of value f_r^k iteratively, as long as $x_r > 0$ holds. Initially, in the k-th iteration, none of the virtual nodes and edges are mapped. As $x_r > 0$ holds, the root node s_r must be mapped accordingly by Constraint 6.9, i.e., there must exist a node $u \in V_S^{r,s_r}$ with $y_{r,s_r}^u > 0$ and the algorithm sets $m_V(s_r) = u$. Given this initial fixing, the algorithm iteratively extracts nodes from the queue $\mathcal{Q}$ which have already been mapped and considers all outgoing virtual edges $(i,j) \in E_r^{\mathcal{X}}$. If the orientation of edge (i,j) was not changed, i.e., if $(i,j) = \overrightarrow{E}_r(i,j)$ holds, then Lemma 6.1 is applied to obtain a mapping of the edge (i,j) together with its head j. If the edge's orientation was reversed, i.e., iff. $(i,j) \neq \overrightarrow{E}_r(i,j)$ holds, Lemma 6.1 can be applied again, only now a path

from the mapping of *the head* i (according to the edge's original orientation) to some mapping of the tail j is obtained. Lastly, the minimum mapping value f_r^k is computed and the variables of the LP (including the allocation variables) are decreased accordingly. The formal correctness of the algorithm is proven in Lemma 6.3.

Lemma 6.3. Given a virtual network request $r \in \mathcal{R}$, whose underlying undirected graph is a tree, and a solution $(x_r, \vec{y}_r, \vec{z}_r, \vec{a}_r)$ to the LP Formulation 6.3, the solution can be decomposed into convex combinations of valid mappings $\mathcal{D}_r = \{(f_r^k, m_r^k) \mid m_r^k \in \mathcal{M}_r, f_r^k \geq 0\}$, such that the following holds:

- The decomposition is complete, i.e., $x_r = \sum_k f_r^k$ holds.
- The decomposition's resource allocations are bounded by $\vec{a}_r$, i.e., $a_r^x \geq \sum_k f_r^k \cdot A(m_r^k, x)$ holds for each resource $x \in G_S$.

Proof. Note that the mapping of each virtual node and each virtual edge is valid by construction: Constraints (6.10) and (6.12) enforce that a node and an edge can only be mapped in a valid fashion. Furthermore, as $G_r^{\mathcal{X}}$ is an arborescence, node mappings are never revoked and each node of G_r will eventually be mapped. The mapping value f_r^k is computed as the minimum of the mapping variables $\mathcal{V}_k$ used for constructing m_r^k. Reducing the values of the mapping variables together with the allocation variables $\vec{a}_r$, the Constraints 6.9 to 6.13 continue to hold.

As the decomposition process continues as long as $x_r > 0$ holds and in the k-th step at least one variable's value is set to 0, it is easy to check that (i) the algorithm terminates with a complete decomposition for which $\sum_k f_r^k = x_r$ holds and (ii) the algorithm has polynomial runtime, as the number of variables for request r is bounded by $\mathcal{O}(|G_r| \cdot |E_S|)$ and in each iteration at least one variable is set to 0. ■

Theorem 6.4. Using the multi-commodity flow formulation, the fractional offline VNEP can be solved optimally in the setting $\langle \mathbf{VE} \,|\, \mathbf{NR} \rangle$ in time $\mathcal{O}(\mathsf{poly}(\sum_{r \in \mathcal{R}} |G_r| \cdot |E_S|))$ when requests are trees.

Proof. The MCF formulation contains $\mathcal{O}(\sum_{r \in \mathcal{R}} |G_r| \cdot |E_S|)$ many variables and accordingly, the time to solve the respective LP is polynomially bounded in the size of the LP. Furthermore, the decomposition algorithms runtime is polynomially bounded by the size of the formulation. Considering the profit variant of the fractional VNEP, the decomposition's completeness (cf. Lemma 6.3) implies that the decomposition's profit equals the objective value of the LP and is hence optimal. Considering the cost variant of the fractional VNEP, by Lemma 6.3 we have that $\sum_k f_r^k = 1$ holds for each request and by the boundedness of the allocation variables, the cost of the convex combination must be less than the LP's objective value; hence, the returned convex combination must also be optimal when considering the cost variant. ■

6.1.3 Limitations of the MCF Formulation

Having shown the decomposability of LP solutions for tree requests, we now show that this does not hold, if the request graphs contain *cycles*. Figure 6.1 gives an example for an LP solution of Formulation 6.3 from which no valid mapping (that is covered) can be extracted. Concretely, considering the mapping of i on u_1 and following the depicted extraction order, k and j must be mapped on u_6 and u_2, at the same time. However, the mapping of j on u_2 only allows for the mapping of k on u_3 and no valid mapping can be extracted and we obtain the following.

Theorem 6.5. Solutions to the LP Formulation 6.3 can (in general) not be decomposed into convex combinations of valid mappings, if request graphs contain cycles. Accordingly, the integrality gap of the LP Formulation 6.3 is unbounded for cyclic request graphs.

Proof. Figure 6.1 depicts an example solution to the LP Formulation 6.3 from which *not a single* valid mapping can be extracted. The validity of the depicted solution is easy to check. As virtual node $i \in V_r$ is mapped onto substrate node $u_1 \in V_S$, and $u_2 \in V_S$ is the only neighboring node with respect to the commodity $z_{r,i,j}$ that hosts $j \in V_r$, a mapping (m_V, m_E) with $m_V(i) = u_1$ and $m_V(j) = u_2$ must exist.

Similarly, $m_V(k) = u_3$ must hold. However, the flow of virtual edge $(k, i) \in E_r$ leaving $u_3 \in V_S$ only leads to $u_4 \in V_S$. Hence the virtual node $i \in V_r$ must be mapped both on u_1 and u_4. As the same argument applies when considering the mapping of i onto u_4, no valid mapping can be extracted.

We now show that the formulation exhibits an unbounded integrality gap. Consider the following restrictions for mapping the virtual links: $E_S^{r,i,j} = \{(u_1, u_2), (u_4, u_5)\}$, $E_S^{r,j,k} = \{(u_2, u_3), (u_5, u_6)\}$, $E_S^{r,k,i} = \{(u_3, u_4), (u_6, u_1)\}$. Note that the solution depicted in Figure 6.1 is still feasible for the MCF LP. Considering the profit variant of the MCF formulation, the LP will attain an objective of b_r. As on the other hand, there does not exist a valid mapping of request r on G_S, the optimal solution achieves a profit of 0. Hence, the integrality gap of the profit formulation is unbounded.

For the cost variant, we add an edge (u_3, u_1) of arbitrarily high cost to the substrate and include this edge in the set of allowed edges for the virtual edge $(k, i) \in E_r$. Hence, there exists only a single valid mapping, which uses this edge (u_3, u_1) while the MCF formulation might still use the LP solution depicted in Figure 6.1. Hence, as the cost of the edge (u_3, u_1) can be arbitrarily high, the integrality gap is unbounded. ■

6.2 LP Formulation for Cactus Request Graphs

In this section, we present a novel LP formulation and its accompanying decomposition algorithm for the class of cactus request graphs, i.e., graphs for which cycles intersect in at most a single node (in its undirected interpretation). Accordingly, these graphs can be uniquely decomposed into cycles and a single forest (cf. Lemma 6.6 below).

Before delving into the details of our cactus LP formulation, we discuss our main insight on how to overcome the limitations of the MCF formulation and accordingly how to derive decomposable formulations. To this end, it is instructive, to revisit the non-decomposable example of Figure 6.1 by applying the decomposition Algorithm 6.4 on the depicted LP solution. Concretely, we consider the acyclic reorientation $G_r^{\mathcal{X}} = (V_r, E_r^{\mathcal{X}}, s_r)$ with $E_r^{\mathcal{X}} = \{(i, k), (i, j), (j, k)\}$, such that i is the root, i.e., $s_r = i$ holds. Assuming that i is initially mapped on node u_1, Algorithm 6.4 will map edges (i, k) and (i, j) first, setting $m_V(k) = u_6$ and $m_V(j) = u_2$ However, when the edge (j, k) is processed, k must be mapped on substrate node $u_3 \neq m_V(k)$ and the algorithm hence fails to produce a valid mapping. Accordingly, to avoid such *diverging* node mappings, our key idea is to decide the mapping location of nodes with more than one incoming edge (with respect to the request's acyclic reorientation $G_r^{\mathcal{X}}$) *a priori*.

By considering only cactus request graphs, the above can be implemented rather easily as exactly one node of each cycle has more than one incoming edge: one only needs to ensure *compatibility* of node mappings for this node. To resolve potential conflicts for the mapping of this unique *cycle target*, our formulation

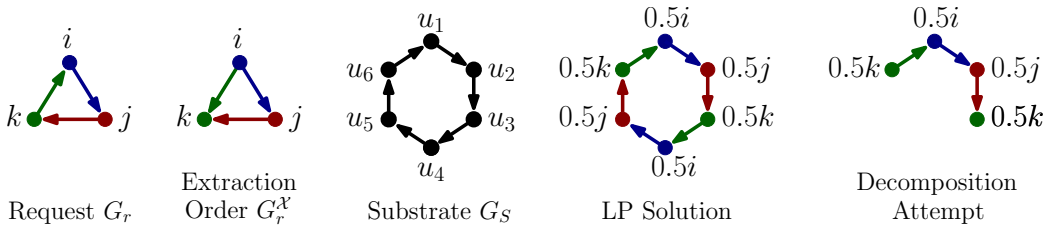


Figure 6.1: Example showing that solutions to the LP Formulation 6.3 cannot be decomposed into convex combinations of valid mappings. The LP solution with $x_r = 1$ is depicted as follows. Substrate nodes are annotated with virtual node mappings: $0.5i$ at node u_1 indicates $y_{r,i}^{u_1} = 1/2$. Substrate edge colors match the color of the virtual edges mapped to it. All virtual edges are also mapped using flow values $1/2$. The color of substrate edge (u_1, u_2) therefore implies that $z_{r,i,j}^{u_1,u_2} = 1/2$ holds.

employs *multiple* copies of the MCF formulation for the respective cyclic subgraph. Specifically, considering a cycle with virtual *target* node k, we instantiate one MCF formulation per substrate node $w \in V_S^{r,k}$ onto which k can be mapped. Accordingly, this yields at most $|V_S|$ many copies and for each of these copies the virtual node k is *fixed* to one specific (substrate) mapping location. Accordingly, as the mapping location of k is fixed to a specific node, valid mappings for the respective cycles can always be extracted from such an MCF copy: the mappings of k cannot possibly diverge.

6.2.1 Cactus Request Graph Decomposition and Notation

Based on the assumed cactus graph nature of request graphs, we consider the following decomposition of request graphs.

Lemma 6.6. Consider a cactus request graph G_r and an acyclic reorientation $G_r^{\mathcal{X}}$ of G_r. The graph $G_r^{\mathcal{X}}$ can be uniquely partitioned into subgraphs $\{G_r^{\mathcal{A},C_1}, \dots, G_r^{\mathcal{A},C_n}\} \sqcup G_r^{\mathcal{A},\mathcal{F}}$, such that the following holds:

(1) The subgraphs $\{G_r^{\mathcal{A},C_1}, \dots, G_r^{\mathcal{A},C_n}\}$ correspond to the (undirected) cycles of G_r and $G_r^{\mathcal{A},\mathcal{F}}$ is the *forest* remaining after removing the cyclic subgraphs. We denote the index set of the cycles by $\mathcal{C}_r = \{C_1, \dots, C_n\}$.

(2) The subgraphs partition the edges of $E_r^{\mathcal{X}}$, i.e., an edge $(i,j) \in E_r^{\mathcal{X}}$ is contained in exactly one of the subgraphs.

(3) The edge set $E_r^{\mathcal{A},C_k}$ of each cycle $C_k \in \mathcal{C}_r$ can itself be partitioned into two branches $\mathcal{B}_1^{C_k}$ and $\mathcal{B}_2^{C_k}$, such that both lead from $s_r^{C_k} \in V_r^{\mathcal{A},C_k}$ to $t_r^{C_k} \in V_r^{\mathcal{A},C_k}$.

Proof. We prove that the lemma holds independently of the chosen acyclic reorientation. Accordingly, let $G_r^{\mathcal{X}}$ be an arbitrary acyclic reorientation of G_r. We first show that $|\delta^-(i)| \leq 2$ holds for each virtual node $i \in V_r$ with respect to the edge set $E_r^{\mathcal{X}}$, i.e., any virtual node has at most 2 incoming edges in $G_r^{\mathcal{X}}$. If $|\delta^-(i)| > 2$ held, then by the definition of the acyclic reorientation there must exist at least 3 paths P_1, P_2, P_3 from the root s_r to i. Let $p_{1,2}, p_{2,3} \in V_r$ be the last common nodes lying on both P_1 and P_2 and P_2 and P_3, respectively. Clearly, from $p_{1,2}$, there exist two (otherwise) node-disjoint paths to i and from $p_{2,3}$ there exist two (otherwise) node-disjoint paths towards i. Accordingly, there exist at least two cycles intersecting either in i and $p_{1,2}$ or i and $p_{2,3}$, which contradicts the assumption on G_r being a cactus graph. Accordingly, $|\delta^-(i)| \leq 2$ holds for all virtual nodes $i \in V_r$ according to $E_r^{\mathcal{X}}$.

Now, consider any node $i \in V_r$ with $|\delta^-(i)| \leq 2$. By performing a graph-search in the opposite direction of edges in $E_r^{\mathcal{X}}$, a unique common ancestor node i' can be determined, such that there exist exactly two paths $\mathcal{B}_1$ and $\mathcal{B}_2$ (branches) from i' to i. The union of these branches represents a single 'cycle' (cf. Statement 3). Removing the identified cycle from the graph, the cactus graph property still holds, as removing edges can never refute it. Accordingly, 'cycles' in the acyclic reorientation $G_r^{\mathcal{A},C_1}, \dots, G_r^{\mathcal{A},C_n}$ can be uniquely identified (decomposed) and after repeated removal only the forest $G_r^{\mathcal{A},\mathcal{F}}$ remains. Hence, the first statement of the lemma follows. Lastly, the second statement of the lemma holds trivially as each edge is either contained in any of the cycles or is part of the remaining forest. ■

Besides the above introduced notation, we denote by $G_r^{C_k}$ and $G_r^{\mathcal{F}}$ the subgraphs that agree with G_r on the edge orientations and use $V_{S,t}^{C_k} = V_S^{r,t_r^{C_k}}$ to denote the substrate nodes on which $t_r^{C_k}$ can be mapped.

6.2.2 LP Formulation for Cactus Request Graphs

Our Formulation 6.5 uses the a priori partition of $G_r^{\mathcal{X}}$ into cycles $G_r^{\mathcal{A},C_k}$ and the forest $G_r^{\mathcal{A},\mathcal{F}}$ to construct MCF formulations for the respective subgraphs: for the subgraph $G_r^{\mathcal{F}}$ a single copy is used (cf. Constraint 6.16) while for the cyclic subgraphs a single MCF formulation is employed *per* potential target

Formulation 6.5: Base Formulation for Cactus Request Graphs

$$\text{Constraints (6.9) - (6.13) for } G_r^{\mathcal{F}} \text{ on variables } (x_r, \vec{y}_r, \vec{z}_r, \vec{a}_r)[\![\mathcal{F}_r]\!] \qquad \forall r \in \mathcal{R} \tag{6.16}$$

$$\text{Constraints (6.9) - (6.13) for } G_r^{C_k} \text{ on variables } (x_r, \vec{y}_r, \vec{z}_r, \vec{a}_r)[\![C_k, w]\!] \qquad \forall r \in \mathcal{R}, C_k \in \mathcal{C}_r, w \in V_{S,t}^{C_k} \tag{6.17}$$

$$x_r = \sum_{u \in V_S^{r,i}} y_{r,i}^u \qquad \forall r \in \mathcal{R}, i \in V_r \tag{6.18}$$

$$y_{r,i}^u = y_{r,i}^u[\![\mathcal{F}]\!] \qquad \forall r \in \mathcal{R}, i \in V_r^{\mathcal{F}}, u \in V_S^{r,i} \tag{6.19}$$

$$y_{r,i}^u = \sum_{w \in t_r^{C_k}} y_{r,i}^u[\![C_k, w]\!] \qquad \forall r \in \mathcal{R}, i \in V_r, u \in V_S^{r,i}, C_k \in \mathcal{C}_r : i \in V_r^{C_k} \tag{6.20}$$

$$0 = y_{r,t_r^{C_k}}^u[\![C_k, w]\!] \qquad \forall r \in \mathcal{R}, C_k \in \mathcal{C}_r, w \in V_{S,t}^{C_k}, u \in V_{S,t}^{C_k} \setminus \{w\} \tag{6.21}$$

$$a_r^u = \sum_{i \in V_r} d_r(i) \cdot y_{r,i}^u \qquad \forall r \in \mathcal{R}, u \in V_S \tag{6.22}$$

$$a_r^{u,v} = a_r^{u,v}[\![\mathcal{F}]\!] + \sum_{C_k \in \mathcal{C}_r, w \in V_{S,t}^{C_k}} a_r^{u,v}[\![C_k, w]\!] \qquad \forall r \in \mathcal{R}, (u,v) \in E_S \tag{6.23}$$

$$\sum_{r \in \mathcal{R}} a_r^x \leq d_S(x) \qquad \forall x \in G_S \tag{6.24}$$

location contained in the set $V_{S,t}^{C_k}$ (cf. Constraint 6.17). We index the variables of these sub-LPs by employing square brackets, i.e., $\alpha[\![\beta]\!]$ refers to the variable α of the sub-LP indexed by β.

To bind together these (at first) independent MCF formulations, we reuse the variables $\vec{x}, \vec{y}$, and $\vec{a}$ introduced already for the MCF formulation. We refer to these variables, which are defined outside of the sub-LP formulations, as *global variables* and do not index these. As we only consider the LP formulation, all variables are continuous.

The different sub-formulations are linked as follows. We employ Constraint 6.18 to enforce the setting of the (global) node mapping variables (cf. Constraint 6.9 of Formulation 6.3). By Constraints 6.19 and 6.20, the node mappings of the sub-LPs for mapping the subgraphs must agree with the global node mapping variables. With respect to cyclic subgraphs, we note that Constraint 6.20 allows for distributing the global node mappings to any of the $|V_{S,t}^{C_k}|$ formulations: only the sum of the node mapping variables must agree with the global node mapping variable. Constraint 6.21 is of crucial importance for the decomposability: considering the sub-LP for cycle C_k and target node $w \in V_{S,t}^{C_k}$, it enforces that the target node $t_r^{C_k}$ of the cycle C_k *must* be mapped on w. Thus, in the sub-LP $[\![C_k, w]\!]$ both branches $\mathcal{B}_1^{C_k}$ and $\mathcal{B}_2^{C_k}$ of cycle C_k are *pre-determined* to lead to the node w. Lastly, for computing node allocations the global node mapping variables are used (cf. Constraint 6.22) and for computing edge allocations the sub-LP formulations' allocations are considered (cf. Constraint 6.23).

6.2.3 Decomposing Solutions to the Cactus LP Formulation

We now argue how to adapt the decomposition Algorithm 6.4 to decompose solutions to the novel Formulation 6.5 for cactus request graphs.

To decompose the LP solution for a request r, the acyclic extraction order $G_r^{\mathcal{X}}$, which was also used for constructing the LP, must be handed over to the decomposition algorithm.

As the LP formulation does not contain (global) edge mapping variables, the edge mapping variables used in Lines 10 and 13 of Algorithm 6.4 must be substituted by edge mapping variables of the respective sub-LP formulations. Concretely, as each edge of the request graph G_r is covered exactly once, it is clear whether a virtual edge $(i,j) \in E_r$ is part of $G_r^{\mathcal{F}}$ or a cyclic subgraph $G_r^{C_k}$. If $(i,j) \in G_r^{\mathcal{F}}$ holds, then the edge mapping variables $z_{r,i,j}^{\cdot,\cdot}[\![\mathcal{F}_r]\!]$ are used. If on the other hand the edge $(i,j) \in E_r$ is covered in the cyclic subgraph $G_r^{C_k}$, then there exist $|V_{S,t}^{C_k}|$ many sub-LPs to choose the respective edge mapping variables from. To ensure the decomposability, we proceed as follows.

If the edge $(i,j) \in E_r^{\mathcal{X}}$ is the first edge of $G_r^{C_k}$ to be mapped in the k-th iteration, the mapping variables $z_{r,i,j}^{\cdot,\cdot}[\![C_k, w]\!]$ belonging to an arbitrary target node w, with $y_{r,i}^{m_V(i)}[\![C_k, w]\!] > 0$, are used. Such a node w exists by Constraint 6.20.

If another edge (i',j') of the same cycle was already mapped in the k-th iteration, the *same* sub-LP as before is considered. Accordingly, the mapping of cycle target nodes cannot conflict, and as these are the only nodes with potential mapping conflicts, the returned mappings are always valid.

To successfully iterate the extraction process, the steps taken in Lines 16 - 18 of Algorithm 6.4 must be adapted to consider the sub-LP variables. Again, as in each iteration at least a single variable of the LP is set to 0 and as the cactus Formulation 6.5 contains at most $\mathcal{O}(|V_S|)$ times more variables than the MCF Formulation 6.3, the decomposition algorithm still runs in polynomial-time. Hence, we conclude that the result of Lemma 6.3 carries over to the LP Formulation 6.5 for *cactus* request graphs and state the following theorem.

Lemma 6.7. Given a solution $(x_r, \vec{y}_r, \vec{z}_r, \vec{a}_r)$ to the LP Formulation 6.5 for a cactus request graph G_r, the solution can be decomposed into a convex combination of valid mappings $\mathcal{D}_r = \{(m_r^k, f_r^k) \mid m_r^k \in \mathcal{M}_r, f_r^k \geq 0\}$, such that:

- The decomposition is complete, i.e., $x_r = \sum_k f_r^k$ holds.
- The decomposition's allocations are bounded by $\vec{a}_r$, i.e., $a_r^x \geq \sum_k f_r^k \cdot A(m_r^k, x)$ holds for $x \in G_S$.

Theorem 6.8. Using the cactus LP formulation, the fractional offline VNEP can be solved optimally in the setting $\langle \mathbf{VE} \,|\, \mathbf{NR} \rangle$ in time $\mathcal{O}(\mathsf{poly}(\sum_{r \in \mathcal{R}} |G_r| \cdot |E_S| \cdot |V_S|))$ when requests are cactus graphs.

Proof. By the above argument, the cactus LP formulation increases in size by at most a factor $\mathcal{O}(|V_S|)$ compared to the MCF formulation. Accordingly, the result follows from Lemma 6.7 by the same observations as made in the proof of Theorem 6.4. ■

6.3 LP Formulations Based on Extraction Width

In this section we present an extension of the above presented LP formulations for cactus requests to solve the fractional VNEP for *arbitrary* request graphs. We first present the high-level idea of our formulation and introduce crucial concepts as extraction confluences and the extraction width. After introducing further notation, we formally present the LP formulation and show the decomposability of its solutions.

6.3.1 Idea and Definitions

Revisiting the shortcomings of the MCF formulation, specifically the example of Figure 6.1, we observe that there exist two virtual paths towards k in $G_r^{\mathcal{X}}$, namely $\langle (i,k) \rangle$ and $\langle (i,j),(j,k) \rangle$. We refer to the combination of two such paths in $G_r^{\mathcal{X}}$ leading from a common virtual node to another common node as an *extraction confluence*:

Definition 6.9 (Extraction Confluence $C_{i,j}^{\mathcal{X}}$)**.** Given an extraction order $G_r^{\mathcal{X}}$, an *extraction confluence* $C_{i,j}^{\mathcal{X}} = P_{i,j}^1 \sqcup P_{i,j}^2$ connects $i \in V_r$ to $j \in V_r$ using two otherwise node-disjoint paths $P_{i,j}^1, P_{i,j}^2 \subseteq E_r^{\mathcal{X}}$. We refer to i as the source and to j as the target of the confluence $C_{i,j}^{\mathcal{X}}$. □

According to the connectivity property of the MCF formulation (cf. Lemma 6.1), partial mappings can always be extended, but the disjoint paths of a confluence might lead to *different* mappings of the confluence's target as depicted in Figure 6.1. However, this *divergence* is only possible when the confluence's target can be mapped on multiple locations and is not fixed.

We use this as follows. Considering a confluence $C_{i,j}^{\mathcal{X}}$, our LP formulation constructs multiple copies of the MCF formulation *for each potential mapping location of the confluence's target.* In each of these copies, the mapping of the confluence's target is fixed to a *specific* substrate node. To generalize this idea to multiple confluences, we label edges with confluence targets as follows.

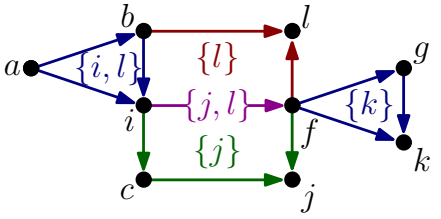


Figure 6.2: Exemplary labeled $G_r^{\mathcal{X}}$.

Definition 6.10 (Extraction Edge Labels)**.** We introduce edge labels $\mathcal{L}_{r,e}^{\mathcal{X}} \subseteq V_r$ for $e \in E_r^{\mathcal{X}}$ as follows. The extraction order edge e is labeled with node j, i.e., $j \in \mathcal{L}_{r,e}^{\mathcal{X}}$ holds, iff. a confluence $C_{i,j}^{\mathcal{X}}$ with target j exists that contains e. We also label the edges in their original orientation accordingly: for edge $e \in E_r$ we set $\mathcal{L}_{r,e} \triangleq \mathcal{L}_{r,e'}^{\mathcal{X}}$ with $e' = \overrightarrow{E}_r^{\mathcal{X}}(e)$. □

The edge labels will be used in our novel LP formulation to instantiate copies of the MCF formulation. Additionally, we introduce *confluence edge bags* which partition outgoing edges.

Definition 6.11 (Confluence Edge Bags)**.** Given an extraction order $G_r^{\mathcal{X}}$, the outgoing edges $\delta_{\mathcal{X}}^+(i)$ of each node $i \in V_r$ are partitioned into a set of edge bags $\mathcal{B}_{r,i}^{\mathcal{X}} = \{B_{r,i}^{\mathcal{X},b}\}_b$, such that two edges $e_1, e_n \in \delta_{\mathcal{X}}^+(i)$ are placed in the same bag $B_{r,i}^{\mathcal{X},b}$, iff. there exists a series of edges $e_2, e_3 \ldots, e_{n-1} \in \delta_{\mathcal{X}}^+(i)$ such that $\mathcal{L}_{r,e_l}^{\mathcal{X}} \cap \mathcal{L}_{r,e_{l+1}}^{\mathcal{X}} \neq \emptyset$ holds for $l \in \{1, \ldots, n-1\}$.

We denote by $\mathcal{L}_{r,i}^{\mathcal{X},b} = \bigcup_{e \in B_{r,i}^{\mathcal{X},b}} \mathcal{L}_{r,e}^{\mathcal{X}}$ the union of labels contained in a bag $B_{r,i}^{\mathcal{X},b} \in \mathcal{B}_{r,i}^{\mathcal{X}}$ and by $\mathcal{L}_{b \cap e}^{\mathcal{X}} = \mathcal{L}_{r,e}^{\mathcal{X}} \cap \mathcal{L}_{r,i}^{\mathcal{X},b}$ the intersection of labels of the bag $B_{r,i}^{\mathcal{X},b}$ and the edge $e \in E_r^{\mathcal{X}}$. □

The size of our formulation will be proven to be exponential in the maximal number of labels contained in any edge bag, and we define the notion of *extraction width* accordingly.

Definition 6.12 (Extraction Width)**.** The width $\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})$ of an extraction order $G_r^{\mathcal{X}}$ is the maximal number of labels contained in an edge bag plus one: $\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}}) = 1 + \max_{i \in V_r, B_{r,i}^{\mathcal{X},b} \in \mathcal{B}_{r,i}^{\mathcal{X}}} |\mathcal{L}_{r,i}^{\mathcal{X},b}|$. Denoting by $\mathcal{X}(G_r)$ the set of all extraction orders of a graph G_r, the extraction width of an arbitrary graph G_r is the minimum width of any extraction order: $\mathrm{ew}(G_r) = \min_{G_r^{\mathcal{X}} \in \mathcal{X}(G_r)} \mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})$. □

Figure 6.2 depicts an example extraction order containing 5 confluences, which can be uniquely identified by their sources and targets: $C_{a,i}^{\mathcal{X}}, C_{i,j}^{\mathcal{X}}, C_{a,l}^{\mathcal{X}}, C_{b,l}^{\mathcal{X}}, C_{f,k}^{\mathcal{X}}$ with, e.g.,$C_{b,l}^{\mathcal{X}} = \{(b,i),(i,f),(f,l)\} \sqcup \{(b,d),(d,l)\}$. According to Definition 6.11, the edge bags of node f are $\mathcal{B}_{r,f}^{\mathcal{X}} = \{B_{r,f}^{\mathcal{X},1} = \{(f,j)\}, B_{r,f}^{\mathcal{X},2} = \{(f,g),(f,k)\}, B_{r,f}^{\mathcal{X},3} = \{(\mathrm{f,l})\}\}$ with the corresponding label sets being $\mathcal{L}_{r,f}^{\mathcal{X},1} = \{j\}$, $\mathcal{L}_{r,f}^{\mathcal{X},2} = \{k\}$, and $\mathcal{L}_{r,f}^{\mathcal{X},3} = \{l\}$. For node i, only a single edge bag $\mathcal{B}_{r,i}^{\mathcal{X}} = \{B_{r,i}^{\mathcal{X},1} = \{(i,c),(i,f)\}\}$ with label set $\mathcal{L}_{r,i}^{\mathcal{X},1} = \{j,l\}$ exists.

6.3.2 Structure of Edge Labels

In the following, we study the structure of extraction confluences and of edge labels. We employ the following notation for indicating edges being reachable from or by nodes in the extraction order.

Definition 6.13 (Reachable Edge Sets)**.** Given an extraction order $G_r^{\mathcal{X}}$, we denote by $E_{r,i}^{\mathcal{X},\text{suc}}, E_{r,j}^{\mathcal{X},\text{pre}} \subseteq E_r^{\mathcal{X}}$ the set of edges which can be reached from $i \in V_r$ and which may lead to $j \in V_r$. We denote by $E_{r,i\rightsquigarrow j}^{\mathcal{X}} = E_{r,i}^{\mathcal{X},\text{suc}} \cap E_{r,j}^{\mathcal{X},\text{pre}}$ the edges lying on a path from i to j. □

The following lemma forms the basis for efficiently computing edge labels.

Lemma 6.14. Edge $e \in E_r^{\mathcal{X}}$ is labeled with $j \in V_r$ iff. there exists a node $i \in V_r$, such that (i) e lies on a path from i to j, i.e., $e \in E_{r,i\rightsquigarrow j}^{\mathcal{X}}$, and (ii) a confluence $C_{i,j}^{\mathcal{X}}$ from i to j exists.

Proof. It is easy to see that the above two conditions are necessary. Clearly, if the first condition does not hold for some node $i \in V_r$, then there cannot exist a confluence from i to j covering the edge e. Secondly, if there does not exist any confluence between i and j, then there cannot exist a confluence from i to j covering e.

We now show that these conditions are also sufficient. First, note that any path from i to j must be contained in $E_{r,i\rightsquigarrow j}^{\mathcal{X}}$. Let $e \in E_{r,i\rightsquigarrow j}^{\mathcal{X}}$ denote any edge for which the above conditions hold. We show that edge e lies on a confluence with target j. By the second condition, there exist two node-disjoint paths $P_{i,j}^1, P_{i,j}^2 \subseteq E_{r,i\rightsquigarrow j}^{\mathcal{X}}$ from i to j. Now, if e lies on either of these paths, then $P_{i,j}^1 \sqcup P_{i,j}^2$ already constitutes a confluence. Hence, assume that e does not lie on either of these paths. Let $e = (k, l)$, i.e., k is the tail and l the head. Furthermore, let $P_{i,k} \subseteq E_{r,i\rightsquigarrow j}^{\mathcal{X}}$ denote any path from i to k and denote by $P_{l,j} \subseteq E_{r,i\rightsquigarrow j}^{\mathcal{X}}$ any path from l to j. Let $P_{i,e,j}$ denote the path obtained from joining $P_{i,k}$, $e = (k, l)$, and $P_{l,j}$.

If $P_{i,e,j}$ only intersects with $P_{i,j}^1$ (or $P_{i,j}^2$), then $P_{i,e,j}$ together with $P_{i,j}^2$ (or $P_{i,j}^1$) constitutes a confluence towards j which covers e, proving our claim. Hence, assume that $P_{i,e,j}$ intersects with both paths. Let k' be the last node on path $P_{i,k}$ which also lies on $P_{i,j}^1$ or $P_{i,j}^2$ and let l' denote the first node on $P_{l,j}$ which also lies on $P_{i,j}^1$ or $P_{i,j}^2$. Assume now w.l.o.g. that both k' and l' lie on path $P_{i,j}^1$, then the subpath of $P_{i,j}^1$ from k' to l' can be substituted with the subpath from k' to l' of $P_{i,e,j}$, yielding the confluence depicted on the left of Figure 6.3. On the other hand, if k' lies on $P_{i,j}^1$ while l' lies on path $P_{i,j}^2$, then there exists a confluence from k' to j covering the edge e: using the suffix of path $P_{i,j}^1$ starting at node k' as first path and the subpath of $P_{i,e,j}$ from k' to l' together with the suffix of $P_{i,j}^2$ starting at l', a confluence is found that covers e. By construction, as the nodes of path $P_{i,e,j}$ between k' and l' do neither lie on $P_{i,j}^1$ nor $P_{i,j}^2$, the paths of the constructed confluence are disjoint (see Figure 6.3 (center) for a visualization). Hence, the two conditions stated in the lemma are also sufficient to decide whether an edge $e \in E_r^{\mathcal{X}}$ is covered by a confluence towards $j \in V_r$ holds. ■

Based on Lemma 6.14, the edge labels can be computed in polynomial-time. Concretely we apply Menger's theorem [Men27] to decide for any combination of virtual nodes $i, j \in V_r$, whether two disjoint paths exist from i to j. If this is the case, then all edges lying in $E_{r,i\rightsquigarrow j}^{\mathcal{X}}$ must be labeled with j and we obtain the following lemma.

Lemma 6.15. The edge labels $\mathcal{L}_{r,e}^{\mathcal{X}}$ can be computed in time $\mathcal{O}(|V_r|^3 \cdot |E_r|)$.

Proof. We argue that the conditions of Lemma 6.14 can be checked in polynomial time. For each potential target node $j \in V_r$ and each source node $i \in V_r$, we check whether two node-disjoint paths exist from i to j by applying Menger's theorem [Men27]: for each node k lying on a path from i to j, we decide whether j is still reachable from i when k is removed. If this is true for each intermediate node, then by Menger's theorem, there exist at least two node-disjoint paths from i to j and hence there must exist a confluence $C_{i,j}^{\mathcal{X}}$. Hence, given the existence of a confluence, all edges in $E_{r,i\rightsquigarrow j}^{\mathcal{X}}$ are labeled by j. At most $\mathcal{O}(|V_r|^2)$ many node pairs need to be considered and the check whether two node-disjoint paths exist can

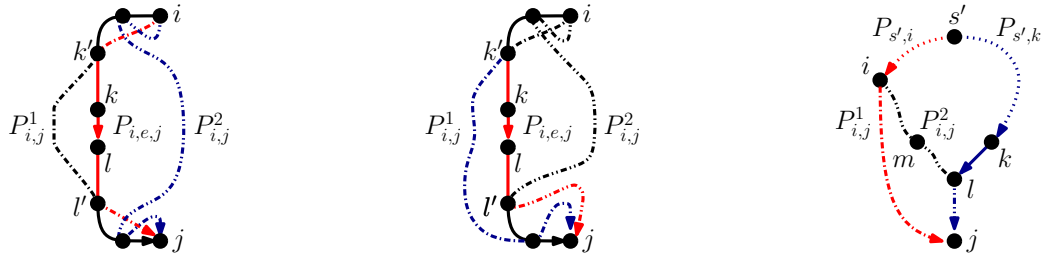


Figure 6.3: Visualization of the constructions used in the proofs of Lemma 6.14 (left and center) and Lemma 6.16 (right) to show that an edge (k, l) is covered by a confluence. The confluence path $P^1_{i,j}$ is dashed with a single dot and the confluence path $P^2_{i,j}$ is dashed with two dots. The constructed confluences consist of the highlighted paths in red and blue.
Lemma 6.14 (left and center): Construction of a confluence when $P^1_{i,j}$ intersects with k' and l' (left). Construction of a confluence when k' lies on $P^1_{i,j}$, while l' lies on $P^2_{i,j}$ (center).
Lemma 6.16 (right): As there must exist a node s' reaching both i and m, a confluence with target j is constructed covering the edge (m, l).

be implemented in time $\mathcal{O}(|V_r| \cdot |E_r|)$. Hence, the overall runtime to compute all labels is bounded by $\mathcal{O}(|V_r|^3 \cdot |E_r|)$. ■

The two following lemmas state important structural properties for edge labels, namely that incoming edges are always labeled the same and that each label has a unique source.

Lemma 6.16. $\mathcal{L}^{\mathcal{X}}_{r,e} = \mathcal{L}^{\mathcal{X}}_{r,e'}$ holds for any pair of incoming edges $e, e' \in \delta^-_{\mathcal{X}}(l)$ of node $l \in V_r$.

Proof. Assume for the sake of contradiction that an edge $e = (m, l)$ is labeled with j, i.e., $j \in \mathcal{L}^{\mathcal{X}}_{r,e}$, and that some other incoming edge $e' = (k, l)$ is not labeled with j, i.e., $j \notin \mathcal{L}^{\mathcal{X}}_{r,e'}$. As the edge e is labeled with j, there must exist some confluence $C^{\mathcal{X}}_{i,j}$ covering e. As the edge $e' = (k, l)$ is not labeled with j, we obtain from Lemma 6.14 that the edge e' is not reachable from i, i.e., $e' \notin E^{\mathcal{X}}_{r,i \rightsquigarrow j}$ holds. As both i and k are reachable from the root s_r of $G^{\mathcal{X}}_r$, there must exist paths $P_{s_r,i}$ and $P_{s_r,k}$ leading from the root s_r to i and k, respectively. Let s' denote the last node that lies on both of these paths. The subpaths $P_{s',i}$ and $P_{s',k}$ of $P_{s_r,i}$ and $P_{s_r,k}$ starting at s' do not use any of the edges in $E^{\mathcal{X}}_{r,i \rightsquigarrow j}$. Hence, joining $P_{s',i}$ with $P^1_{i,j}$ and joining $P_{s',k}$ with $e' = (k, l)$ and the subpath of $P^2_{i,j}$ beginning at node l, a confluence is constructed that covers edge e' (see Figure 6.3 (right) for an visualization of the construction). Therefore, also e' must be labeled with j, yielding a contradiction to our assumption that e' was not labeled by j and all incoming edges must be labeled the same. ■

Lastly, the following lemma shows that any label is introduced only once.

Lemma 6.17. For each label $j \in V_r$ there exists a unique root node $s_j \in V_r$, such that:

(1) Any edge $e \in E^{\mathcal{X}}_r$ being labeled with $j \in \mathcal{L}^{\mathcal{X}}_{r,e}$ is contained in $E^{\mathcal{X}}_{r,s_j \rightsquigarrow j}$.

(2) Any path from s_r (the root of the extraction order) to j passes through s_j.

Proof. Consider two nodes $i, i' \in V_r$ being the sources of confluences $C^{\mathcal{X}}_{i,j}$ and $C^{\mathcal{X}}_{i',j}$ towards j. Assume that neither i occurs in $E^{\mathcal{X}}_{r,i' \rightsquigarrow j}$ nor that i' occurs in $E^{\mathcal{X}}_{r,i \rightsquigarrow j}$. As the graph $G^{\mathcal{X}}_r$ is rooted, there must exist a node s' reaching both i and i' and spawning a confluence towards j. Furthermore, $(E^{\mathcal{X}}_{r,i \rightsquigarrow j} \cup E^{\mathcal{X}}_{r,i' \rightsquigarrow j}) \subseteq E^{\mathcal{X}}_{r,s' \rightsquigarrow j}$ holds in this case. Hence, for any pair of nodes i, i' being sources of confluences towards j, either one of the nodes is reachable from the other, or there exists another node $s' \in V_r$ such that $E^{\mathcal{X}}_{r,i \rightsquigarrow j}$ and $E^{\mathcal{X}}_{r,i' \rightsquigarrow j}$ are contained in $E^{\mathcal{X}}_{r,s' \rightsquigarrow j}$. Clearly, as either i dominates i' or i' dominates i, or there exists some other

node s' dominating both, there must exist a single unique root node $s_j \in V_r$ such that all edges labeled with j lie in $E^{\mathcal{X}}_{r,s_j \rightsquigarrow j}$.

The second claim is immediate: if there was to exist some path from the root s_r to an edge being labeled with j without passing through the unique root $s_j \in V_r$, then there must exist a confluence $C^{\mathcal{X}}_{s',j}$ starting at some other node $s' \in V_r$, such that s' reaches s_j but s_j does not reach s'. Hence, by our above observation s' dominates s_j and s_j cannot be the unique root. Thus, all paths from the root must pass through s_j on their way to j. ■

6.3.3 Decomposable Extraction Width LP Formulation

Our novel Linear Programming Formulation 6.6 is based on the idea to decide the mapping locations of confluence targets a priori. We do so by considering copies or sub-LPs of the MCF formulation (see Constraint 6.25) and we employ the following notation. For an edge $e = (i, j) \in E_r$, we denote by $G_{r,e} = (V_{r,e}, E_{r,e})$ with $V_{r,e} = \{i, j\}$ and $E_{r,e} = \{e\}$ the subgraph of G_r containing only edge e. Variables of sub-LPs are named as before: $\alpha[\![\beta]\!]$ denotes the variable α in the copy identified by β. To denote the combinations of mapping possibilities of labels, we employ $\mathcal{M}(X)$ to denote the function space from the set X to V_S, i.e., $\mathcal{M}(X) = [X \to V_S]$. Accordingly, considering an edge $e \in E_r$ of request $r \in \mathcal{R}$ being labeled by $\mathcal{L}_{r,e}$, we instantiate one copy of the MCF formulation per edge label mapping $m^{\mathcal{L}}_e \in \mathcal{M}(\mathcal{L}_{r,e})$ (cf. Constraint 6.25). For better readability, we write $\langle\!\langle f|Z \rangle\!\rangle : Z \to Y$ to denote $f_{|Z}$, i.e., the (standard) restriction of the function $f : X \to Y$ on the subset $Z \subseteq X$. Hence, $\langle\!\langle f|Z \rangle\!\rangle(z) = f(z)$ holds for $z \in Z$.

To link the LP copies, we employ two types of *global* node mapping variables. We use the global $y^u_{r,i}$ variables already presented in Formulation 6.3 as well as node mapping variables $\gamma^u_{r,i,b,a} \in [0, 1]$ for *edge bags* $B^{\mathcal{X},b}_{r,i} \in \mathcal{B}^{\mathcal{X}}_{r,i}$, each mapping $m^{\mathcal{L}}_a \in \mathcal{M}(\mathcal{L}^{\mathcal{X},b}_{r,i})$ of the labels contained in the respective edge bag, and the mapping locations $u \in V^{r,i}_S$. The classic variables $y^u_{r,i}$ are used for coupling the embedding variable x_r and the sub-LP node mapping variables (see Constraints 6.26 and 6.27) as well as for computing the node allocations (see Constraint 6.31). The node mapping variables for edge bags $\gamma^u_{r,i,b,a}$ are defined for all mappings of their label set $m^{\mathcal{L}}_a \in \mathcal{M}(\mathcal{L}^{\mathcal{X},b}_{r,i})$. As $\mathcal{L}^{\mathcal{X}}_{r,e} \subseteq \mathcal{L}^{\mathcal{X},b}_{r,i}$ holds for all edges $e \in B^{\mathcal{X},b}_{r,i}$, the node mapping variables of an edge bag directly induce node mappings for all edges contained in the respective bag (see Constraint 6.28).

In the following, we argue how 'flows' are induced and accordingly how solutions to the formulation can be decomposed. Figure 6.4 visualizes the workings of Constraints 6.28 to 6.30. Assuming that $x_r > 0$ holds, then by Constraint 6.26 there will exist a substrate node $u \in V^{r,s_r}_S$ onto which the root s_r is placed, i.e., $y^u_{r,s_r} > 0$ holds. Constraint 6.27 distributes the quantity y^u_{r,s_r} over the sub-LP node mapping variables while Constraint 6.28 ensures that these node mapping variables agree with each other. Due to the validity of the MCF Formulation 6.3, by setting the node mapping variable for one of the endpoints of the edge graph $G_{r,e}$, the node mapping variables of the other endpoint of $G_{r,e}$ must be set accordingly. On the other hand, Constraint 6.29 ensures that any incoming edge (according to the extraction order) agrees with the respective node bag variables and hence force the further distribution of 'flows'. The correctness of the formulation then mostly follows from the following observations:

1. Based on the acyclicity of the extraction order and the fact that all nodes can be reached from the root s_r, 'flow' is distributed throughout the whole request graph.

2. A novel edge label j is introduced only exactly once according to Lemma 6.17, namely at the root $s_j \in V_r$ and hence only at node s_j the a priori mapping of j is fixed.

3. For any confluence $C^{\mathcal{X}}_{k,i}$ with target i all incoming edges of i are itself labeled by i (cf. Lemma 6.16).

Formulation 6.6: Novel Decomposable Base Formulation for the VNEP

(6.9) - (6.13) for $G_{r,e}$ on variables $(x_r, \vec{y}_r, \vec{z}_r, \vec{a}_r)[\![e, m_e^{\mathcal{L}}]\!]$ $\quad \forall r \in \mathcal{R}, e \in E_r, m_e^{\mathcal{L}} \in \mathcal{M}(\mathcal{L}_{r,e})$ (6.25)

$$x_r = \sum_{u \in V_S^{r,i}} y_{r,i}^u \qquad \forall r \in \mathcal{R} \tag{6.26}$$

$$y_{r,i}^u = \sum_{m_e^{\mathcal{L}} \in \mathcal{M}(\mathcal{L}_{r,e})} y_{r,i}^u[\![e, m_e^{\mathcal{L}}]\!] \qquad \forall r \in \mathcal{R}, i \in V_r, u \in V_S^{r,i}, e \in E_r : i \in V_{r,e} \tag{6.27}$$

$$y_{r,i}^u[\![\overrightarrow{E}_r(e), m_e^{\mathcal{L}}]\!] = \sum_{\substack{m_a^{\mathcal{L}} \in \mathcal{M}(\mathcal{L}_{r,i}^{\mathcal{X},b}): \\ \langle m_a^{\mathcal{L}} | \mathcal{L}_{b \cap e}^{\mathcal{X}} \rangle = m_e^{\mathcal{L}}}} \gamma_{r,i,b,a}^u \qquad \begin{array}{l} \forall r \in \mathcal{R}, i \in V_r, u \in V_S^{r,i}, B_{r,i}^{\mathcal{X},b} \in \mathcal{B}_{r,i}^{\mathcal{X}}, \\ \qquad e \in B_{r,i}^{\mathcal{X},b}, m_e^{\mathcal{L}} \in \mathcal{M}(\mathcal{L}_{r,e}^{\mathcal{X}}) \end{array} \tag{6.28}$$

$$\sum_{\substack{m_e^{\mathcal{L}} \in \mathcal{M}(\mathcal{L}_{r,e}^{\mathcal{X}}): \\ \langle m_e^{\mathcal{L}} | \mathcal{L}_{b \cap e}^{\mathcal{X}} \rangle = m_{b \cap e}^{\mathcal{L}}}} y_{r,i}^u[\![\overrightarrow{E}_r(e), m_e^{\mathcal{L}}]\!] = \sum_{\substack{m_a^{\mathcal{L}} \in \mathcal{M}(\mathcal{L}_{r,i}^{\mathcal{X},b}): \\ \langle m_a^{\mathcal{L}} | \mathcal{L}_{b \cap e}^{\mathcal{X}} \rangle = m_{b \cap e}^{\mathcal{L}}}} \gamma_{r,i,b,a}^u \qquad \begin{array}{l} \forall r \in \mathcal{R}, i \in V_r, u \in V_S^{r,i}, e \in \delta_{\mathcal{X}}^-(i), \\ \qquad B_{r,i}^{\mathcal{X},b} \in \mathcal{B}_{r,i}^{\mathcal{X}}, m_{b \cap e}^{\mathcal{L}} \in \mathcal{M}(\mathcal{L}_{b \cap e}^{\mathcal{X}}) \end{array} \tag{6.29}$$

$$y_{r,i}^u[\![\overrightarrow{E}_r(e), m_e^{\mathcal{L}}]\!] = 0 \qquad \begin{array}{l} \forall r \in \mathcal{R}, i \in V_r, e \in \delta_{\mathcal{X}}^-(i) : i \in \mathcal{L}_{r,e}^{\mathcal{X}}, \\ m_e^{\mathcal{L}} \in \mathcal{M}(\mathcal{L}_{r,e}^{\mathcal{X}}), u \in V_S^{r,i} \setminus \{m_e^{\mathcal{L}}(i)\} \end{array} \tag{6.30}$$

$$a_r^u = \sum_{i \in V_r} d_r(i) \cdot y_{r,i}^u \qquad \forall r \in \mathcal{R}, u \in V_S \tag{6.31}$$

$$a_r^{u,v} = \sum_{\substack{e \in E_r \\ m_e^{\mathcal{L}} \in \mathcal{M}(\mathcal{L}_{r,e})}} a_r^{u,v}[\![e, m_e^{\mathcal{L}}]\!] \qquad \forall r \in \mathcal{R}, (u,v) \in E_S \tag{6.32}$$

$$\sum_{r \in \mathcal{R}} a_r^x \leq d_S(x) \qquad \forall x \in G_S \tag{6.33}$$

$$\begin{array}{c} x_r \in [0,1], \ \forall r \in \mathcal{R}; \quad y_{r,i}^u \in [0,1], \ \forall r \in \mathcal{R}, i \in V_r, u \in V_S^{r,i}; \quad a_r^x \geq 0, \ \forall r \in \mathcal{R}, x \in G_S \\ \gamma_{r,i,b,a}^u \in [0,1], \ \forall r \in \mathcal{R}, i \in V_r, u \in V_S^{r,i}, B_{r,i}^{\mathcal{X},b} \in \mathcal{B}_{r,i}^{\mathcal{X}}, m_a^{\mathcal{L}} \in \mathcal{M}(\mathcal{L}_{r,i}^{\mathcal{X},b}) \end{array} \tag{6.34}$$

Accordingly, considering a mapping $m_e^{\mathcal{L}} \in \mathcal{M}(\mathcal{L}_{r,e})$ of an incoming edge $e \in \delta_{\mathcal{X}}^-(i)$, Constraint 6.30 explicitly forbids node placements of i to nodes $V_S^{r,i} \setminus \{m_e^{\mathcal{L}}(i)\}$ in the respective sub-LPs. Hence, incoming edges of node i labeled by $m_e^{\mathcal{L}}$ can only map i to $m_e^{\mathcal{L}}(i) \in V_S^{r,i}$.

Theorem 6.18. Considering specific extraction orders $G_r^{\mathcal{X}}$ for each request $r \in \mathcal{R}$, the size of the novel decomposable Formulation 6.6 is bounded by $\mathcal{O}(\sum_{r \in \mathcal{R}} |G_S|^{\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})} \cdot |G_r|)$.

Proof. Consider a single request $r \in \mathcal{R}$ and a fixed extraction order $G_r^{\mathcal{X}}$. There are at most $\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}}) - 1$ many sub-LPs for each edge $e \in E_r^{\mathcal{X}}$, as $|\mathcal{L}_{r,e}^{\mathcal{X}}| \leq \mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}}) - 1$ holds by definition. Otherwise, the formulation's size is dominated by the node bag mapping variables $\gamma_{r,i,b,a}^u$ and the respective Constraints 6.28 - 6.30. Since the bags partition the outgoing edges and encode all potential mappings of the respective label set $\mathcal{L}_{r,i}^{\mathcal{X},b}$ while including the mapping location of the respective virtual node $i \in V_r$, the size of the respective formulation parts is bounded by $\mathcal{O}(|V_r| \cdot |V_S|^{\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})})$. The result is obtained by summing over the requests. ∎

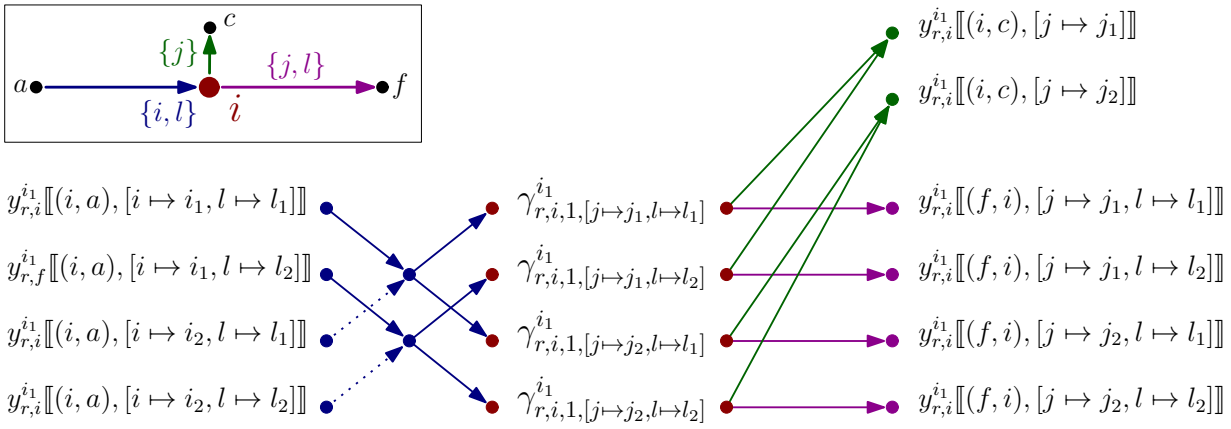


Figure 6.4: Visualization of the relation of the different node mapping variables for the example of Figure 6.2 under the assumption that the virtual nodes $i, j, k \in V_r$ can be mapped only on $i_k, j_k, l_k \in V_S$ for $k \in \{1, 2\}$, respectively. Depicted are only the variables relating to the mapping of i to i_1. To highlight that the LP copies are created upon the original orientation of edges, we assume that $(i, a), (f, i) \in E_r$ holds and that these were reversed for $G_r^{\mathcal{X}}$ depicted in Figure 6.2 on Page 54.
The edges and nodes are to be read as 'flows' for which flow preservation holds. The directions of the edges shall help the reader to follow how the node mapping variables of 'incoming' edges trigger the node mapping variables of 'outgoing' edges. The connections on the left are due to Constraint 6.29 and the connections on the right are due to Constraint 6.28. Note that the dashed edges on the left will be 0 due to Constraint 6.30: in the index of the respective sub-LP, the virtual node i is mapped onto i_2 and hence the respective dashed variables are set to 0.

6.3.4 Decomposition Algorithm for the Extraction Width LP Formulation

We now formally present the decomposition algorithm (see Algorithm 6.7) and prove its correctness. The algorithm builds on the ideas of the decomposition algorithm for the MCF Formulation 6.3 and the cactus LP Formulation 6.5.

Fixing the mapping of the root initially, mappings for the outgoing edges (with respect to the extraction order) are extracted again together with the mappings of the heads of these edges using Lemma 6.1. However, as the edge embeddings are computed using a copy of the MCF formulation for each node mapping function of the edge's labels, the mapping of the edge's labels to substrate nodes must be fixed first. To this end, we employ the node mapping variables $\gamma^u_{r,i,b,a}$ of the edge bags. Concretely, whenever the outgoing edges of $i \in V_r$ are mapped, we show that given the mapping of the virtual node i onto some substrate node $m_V(i) = u$, we can always find a variable $\gamma^u_{r,i,b,a}$ for edge bag $B^{\mathcal{X},b}_{r,i} \in \mathcal{B}^{\mathcal{X}}_{r,i}$ and $m^{\mathcal{L}}_a \in \mathcal{M}(\mathcal{L}^{\mathcal{X},b}_{r,i})$, such that (i) the mapping of the bag $m^{\mathcal{L}}_a$ agrees with the previous node mappings, i.e., $m^{\mathcal{L}}_a(i') = m_V(i')$ holds for all previously mapped virtual nodes i', and that (ii) the variable $\gamma^u_{r,i,b,a}$ is strictly greater 0. Given such a variable $\gamma^u_{r,i,b,a}$ and fixing the node mappings of all the new labels contained in $\mathcal{L}^{\mathcal{X}}_{r,e}$ according to $m^{\mathcal{L}}_a$, mappings for the outgoing edges can always be extracted due to Constraint 6.28. Concretely, after having fixed the mappings of the labels of the outgoing edge e, Constraint 6.28 ensures that for the sub-LP with index $\langle m_V|\mathcal{L}^{\mathcal{X}}_{r,e}\rangle$ the condition $y^u_{r,i} > 0$ holds, thereby allowing the application of Lemma 6.1 to extract the mapping for the edge e. Having given this intuition, we now formally prove its correctness.

Theorem 6.19. For a request $r \in \mathcal{R}$ and its extraction order $G_r^{\mathcal{X}}$, a solution to Formulation 6.6 can be decomposed into $\mathcal{D}_r = \{(f^k_r, m^k_r) \mid m^k_r \in \mathcal{M}_r, f^k_r \geq 0\}$ in time $\mathcal{O}(|G_S|^{2\cdot \mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})+1} \cdot |G_r|^2)$, such that:

- The decomposition is complete, i.e., $x_r = \sum_k f^k_r$ holds.

Algorithm 6.7: Decomposition algorithm for solutions to the novel LP formulation 6.6

Input : Request $r \in \mathcal{R}$, extraction order $G_r^{\mathcal{X}}$, solution to Formulation 6.6
Output: Convex combination $\mathcal{D}_r = \{D_r^k = (f_r^k, m_r^k)\}_k$ of valid mappings

set $\mathcal{D}_r \leftarrow \emptyset$ **and** $k \leftarrow 1$
while $x_r > 0$ **do**
  set $m_r^k = (m_V, m_E) \leftarrow (\emptyset, \emptyset)$
  set $\mathcal{Q} \leftarrow \{s_r\}$
  choose $u \in V_S^{r,s_r}$ **with** $y_{r,s_r}^u > 0$ **and set** $m_V(s_r) \leftarrow u$
  while $|\mathcal{Q}| > 0$ **do**
    choose $i \in \mathcal{Q}$ **and set** $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{i\}$
    foreach $B_{r,i}^{\mathcal{X},b} \in \mathcal{B}_{r,i}^{\mathcal{X}}$ **do**
      let $M^V = (m_V)^{-1}(V_S)$ denote the already mapped nodes
      choose $m_a^{\mathcal{L}} \in \mathcal{M}(\mathcal{L}_{r,i}^{\mathcal{X},b})$, s.t. $\gamma_{r,i,b,a}^{m_V(i)} > 0$ **and** $\langle\!| m_a^{\mathcal{L}} | \mathcal{L}_{r,i}^{\mathcal{X},b} \cap M^V |\!\rangle = \langle\!| m_V | \mathcal{L}_{r,i}^{\mathcal{X},b} \cap M^V |\!\rangle$
      set $m_V(j) \leftarrow m_a^{\mathcal{L}}(j)$ **for all** $j \in \mathcal{L}_{r,i}^{\mathcal{X},b} \setminus M^V$
      foreach $e = (i,j) \in B_{r,i}^{\mathcal{X},b}$ **do**
        if $(i,j) = \overrightarrow{E}_r(i,j)$ **then**
          compute path $P_{r,i,j}^{u,v}$ from $m_V(i) = u$ to $v \in V_S^{r,j}$ according to Lemma 6.1
          $$\text{s.t.} \quad \begin{matrix} y_{r,j}^v [\![(i,j), \langle\!| m_V | \mathcal{L}_{r,e} |\!\rangle]\!] > 0 \quad \textbf{and} \\ z_{r,i,j}^{u',v'} [\![(i,j), \langle\!| m_V | \mathcal{L}_{r,e} |\!\rangle]\!] > 0 \quad \textbf{for all } (u',v') \in P_{r,i,j}^{u,v} \end{matrix}$$
          set $m_E(i,j) \leftarrow P_{r,i,j}^{u,v}$ **and if** $m_V(j) = \emptyset$ **then** $m_V(j) \leftarrow v$
        else
          compute path $P_{r,j,i}^{v,u}$ from $v \in V_S^{r,j}$ to $m_V(i) = u$ according to Lemma 6.1
          $$\text{s.t.} \quad \begin{matrix} y_{r,j}^v [\![\overrightarrow{E}_r(i,j), \langle\!| m_V | \mathcal{L}_{r,e}^{\mathcal{X}} |\!\rangle]\!] > 0 \quad \textbf{and} \\ z_{r,j,i}^{u',v'} [\![\overrightarrow{E}_r(i,j), \langle\!| m_V | \mathcal{L}_{r,e}^{\mathcal{X}} |\!\rangle]\!] > 0 \quad \textbf{for all } (u',v') \in P_{r,j,i}^{u,v} \end{matrix}$$
          set $m_E(\overrightarrow{E}_r(i,j)) \leftarrow P_{r,j,i}^{u,v}$ **and if** $m_V(j) = \emptyset$ **then** $m_V(j) \leftarrow v$
        if $m_E(\overrightarrow{E}_r(e)) \neq \emptyset$ **for all** $e \in \delta_{\mathcal{X}}^-(j)$ **then**
          set $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{j\}$
    set $\mathcal{V}_k \leftarrow \left(\begin{matrix} & \{x_r\} \cup \{y_{r,i}^u \mid i \in V_r, u = m_V(i)\} \\ \cup & \{ \ x_r [\![e, \langle\!| m_V | \mathcal{L}_{r,e} |\!\rangle]\!] \mid e \in E_r \} \\ \cup & \{ \ y_{r,i}^u [\![e, \langle\!| m_V | \mathcal{L}_{r,e} |\!\rangle]\!] \mid e \in E_r, i \in V_{r,e}, u = m_V(i) \} \\ \cup & \{ z_{r,i,j}^{u,v} [\![e, \langle\!| m_V | \mathcal{L}_{r,e} |\!\rangle]\!] \mid e = (i,j) \in E_r, (u,v) \in m_E(i,j) \} \\ \cup & \{ \gamma_{r,i,b,a}^u \mid i \in V_r, u = m_V(i), B_{r,i}^{\mathcal{X},b} \in \mathcal{B}_{r,i}^{\mathcal{X}}, m_a^{\mathcal{L}} = \langle\!| m_V | \mathcal{L}_{r,i}^{\mathcal{X},b} |\!\rangle \} \end{matrix} \right)$
    set $f_r^k \leftarrow \min \mathcal{V}$
    set $v \leftarrow v - f_r^k$ **for all** $v \in \mathcal{V}$
    set $a_r^x \leftarrow a_r^x - f_r^k \cdot A(m_r^k, x)$ **for all** $x \in G_S$
    foreach $(i,j) \in E_r$ **and each** $x \in \{i, j, (i,j)\}$ **do**
      set $a_r^x [\![(i,j), \langle\!| m_V | \mathcal{L}_{r,e} |\!\rangle]\!] \leftarrow a_r^x [\![\langle\!| m_V | \mathcal{L}_{r,e} |\!\rangle]\!] - f_r^k \cdot A(m_r^k, x)$
    add $D_r^k = (f_r^k, m_r^k)$ to $\mathcal{D}_r$ **and set** $k \leftarrow k + 1$
return $\mathcal{D}_r$

- Allocations are bounded by $\vec{a}_r$, i.e., $a_r^x \geq \sum_{(f_r^k, m_r^k) \in \mathcal{D}_r} f_r^k \cdot A(m_r^k, x)$ holds for $x \in G_S$.

Proof. We prove that each iteration yields a valid mapping m_r^k of value $f_r^k > 0$.

First, note that if in Line 10 a suitable mapping $m_a^{\mathcal{L}}$ was found, such that the respective edge bag variable $\gamma_{r,i,b,a}^{m_V(i)}$ is positive, then the requirement of Lemma 6.1 that $y_{r,i}^{m_V(i)}[\![e, \langle m_V | \mathcal{L}_{r,e}^{\mathcal{X}} \rangle]\!] > 0$ holds is always satisfied due to Constraint 6.28.

The initial mapping of the root in Line 5 is always possible due to Constraints 6.26. Furthermore, when considering the edge bags of the root s_r, there will always exist a suitable edge bag variable $\gamma_{r,s_r,b,a}^{m_V(s_r)} > 0$ to choose from due to Constraints 6.27 and 6.28.

Having chosen a suitable mapping for the labels of the edge bag, the extraction of mappings for the outgoing edges $e \in B_{r,i}^{\mathcal{X},b}$ is always feasible, as Constraint 6.28 induces $y_{r,s_r}^{m_V(s_r)} > 0$ in the respective sub-LPs. Also note that the application of Lemma 6.1 safeguards that the head j is mapped positively on some substrate node i, i.e., we have $y_{r,j}^{v}[\![\langle m_V | \mathcal{L}_{r,e}^{\mathcal{X}} \rangle]\!] > 0$.

Given the initial validity of the mapping of the root and its outgoing edges, assume now for the sake of contradiction that the extraction process fails at some point in time. Concretely, we consider the first point in time at which the constructed (partial) mapping $m_r^k = (m_V, m_E)$ is not valid anymore or at which the choose operation in Line 10 fails.

We first consider the case that the mapping m_r^k is not valid (anymore), such that the mapping of an edge $e = (i, j)$ fails to start at $m_V(i)$ or fails to lead to $m_V(j)$. Edges are only mapped in Lines 15 and 18 and we consider w.l.o.g. that the algorithm fails in Line 15. For this type of failure to happen, the node j must have been mapped before as the node j is otherwise validly mapped by the same line of the pseudocode. As j can only be mapped multiple times if j is itself a label and all incoming edges of a node share the same labels (see Lemma 6.16), the edge (i, j) must have been labeled by j. Let v' denote the substrate node in which the path $m_E(i, j)$ ends and for which $v' \neq m_V(i)$ holds. As stated in the beginning of the proof, the requirement of Lemma 6.1 is always valid (if a suitable mapping $m_a^{\mathcal{L}}$ was found) and hence, by applying Lemma 6.1 we obtain $y_{r,j}^{v'}[\![e, \langle m_V | \mathcal{L}_{r,e} \rangle]\!] > 0$. However, Constraint 6.30 clearly forbids the use of this node v' as it does not equal $m_V(i)$ by setting $y_{r,j}^{v'}[\![e, \langle m_V | \mathcal{L}_{r,e} \rangle]\!] = 0$. This is a contradiction, and the only option for the extraction process to fail is hence due to an infeasible choose operation in Line 10.

As argued in the beginning of the proof, the choose operation may not fail for the root. Hence, the node $i \in V_r$ for which Line 10 fails, is not the root and has been reached by at least one incoming edge $(k, i) \in E_r^{\mathcal{X}}$. Assume that the choose operation fails for a specific edge bag $B_{r,i}^{\mathcal{X},b} \in \mathcal{B}_{r,i}^{\mathcal{X}}$.

We first show that $\mathcal{L}_{r,i}^{\mathcal{X},b} \cap M^V \subseteq \mathcal{L}_{r,(k,i)}^{\mathcal{X}}$ holds. Assume for the sake of contradiction, that there exists some label $l \in \mathcal{L}_{r,i}^{\mathcal{X},b} \cap M^V$ such that $l \notin \mathcal{L}_{r,(k,i)}^{\mathcal{X}}$ holds. As the label l is contained in M^V, a mapping was decided for l at some other node $s_l \in V_r$. In particular, Lemma 6.17 specifies that the node s_l is the unique root of all the confluences towards l, such that any other node with an edge being labeled by l must be reachable from s_l. However, as all incoming labels agree on their labels (see Lemma 6.16) and no incoming edge of the node i hence lies on a confluence with target l, we must have $i = s_l$. In this case however, the node mapping of l cannot have been decided before as the algorithm only fixes these node mappings once the choose operation was executed at the respective node s_l. Hence, $\mathcal{L}_{r,i}^{\mathcal{X},b} \cap M^V \subseteq \mathcal{L}_{r,(k,i)}^{\mathcal{X}}$ holds.

As all previous mapping steps have been valid and the mappings were obtained by the application of Lemma 6.1, we know that $y_{r,i}^{u}[\![\overrightarrow{E}_r((k,i)), \langle m_V | \mathcal{L}_{r,(k,i)}^{\mathcal{X}} \rangle]\!] > 0$ holds for $u = m_V(i)$. Consider now the particular mapping $m_{(k,i)}^{\mathcal{L}} = \langle m_V | \mathcal{L}_{b \cap (k,i)}^{\mathcal{X}} \rangle$ which is well-defined, as all labels of the incoming edge (k, i) must have been fixed before extracting the mapping of this edge. From the validity of Constraint 6.29 we obtain that $\sum_{m_a^{\mathcal{L}} \in \mathcal{M}(\mathcal{L}_{r,i}^{\mathcal{X},b}) : \langle m_a^{\mathcal{L}} | \mathcal{L}_{b \cap e}^{\mathcal{X}} \rangle = m_{(k,i)}^{\mathcal{L}}} \gamma_{r,i,b,a}^{u} > 0$ holds, as $y_{r,i}^{u}[\![\overrightarrow{E}_r((k,i)), \langle m_V | \mathcal{L}_{r,(k,i)}^{\mathcal{X}} \rangle]\!]$ is larger than 0. As $\mathcal{L}_{r,i}^{\mathcal{X},b} \cap M^V \subseteq \mathcal{L}_{r,(k,i)}^{\mathcal{X}}$ holds, we know that $\sum_{m_a^{\mathcal{L}} \in \mathcal{M}(\mathcal{L}_{r,i}^{\mathcal{X},b}) : \langle m_a^{\mathcal{L}} | \mathcal{L}_{r,i}^{\mathcal{X},b} \cap M^V \rangle = \langle m_V | \mathcal{L}_{r,i}^{\mathcal{X},b} \cap M^V \rangle} \gamma_{r,i,b,a}^{u} > 0$

holds, since the restriction in the sum's index has been loosened. Hence, the choose operation in Line 10 can always be successfully executed and the mapping constructed in the k-th iteration will always be valid.

It is easy to check that the claims with respect to the completeness and the bounds by the allocation variables also hold: the mapping is always covered by respective mapping variables in $\mathcal{V}_k$ and as the allocations are computed as a function of these mapping variables, the extracted fractional resource allocations are also bounded by $\vec{a}_r$.

Lastly, every time a valid mapping is extracted, a mapping variable's value is set to 0. As the formulation has size $\mathcal{O}(|G_S|^{\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})} \cdot |G_r|)$ (cf. Theorem 6.18) for request r, and this also bounds the number of variables, at most $\mathcal{O}(|G_S|^{\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})} \cdot |G_r|)$ many valid mappings may be recovered. For recovering a single valid mapping, the runtime can be bounded by $\mathcal{O}(|G_S|^{\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})+1} \cdot |G_r|)$, as the **choose** operation in Line 10 is executed at most $|V_r|$ times and the path computations in Lines 14 and 17 can be implemented in time $\mathcal{O}(|E_S|)$ (cf. Lemma 6.1). Hence, the overall runtime of the decomposition algorithm is bounded by $\mathcal{O}(|G_S|^{2\cdot \mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})+1} \cdot |G_r|^2)$. ■

Theorem 6.20. Using the extraction width LP formulation, the fractional offline VNEP can be solved optimally in the setting $\langle \mathbf{VE} \,|\, \mathbf{NR} \rangle$ in time $\mathcal{O}(\mathsf{poly}(\sum_{r\in\mathcal{R}} |G_S|^{\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})} \cdot |G_r|))$ given specific extraction orders $G_r^{\mathcal{X}}$ for each request $r \in \mathcal{R}$.

Proof. The size of the LP formulation was proven to be $\mathcal{O}(\sum_{r\in\mathcal{R}} |G_S|^{\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})} \cdot |G_r|)$ in Lemma 6.18 and the time to solve the LP is polynomial in the size. Accordingly, the result follows from Theorem 6.19 by the same observations as made in the proof of Theorem 6.4. ■

6.3.5 Extraction Width: Graph Classes and Complexity

The runtime to solve the fractional offline VNEP using the LP Formulation 6.6 grows exponentially in the maximal width of any extraction order. Hence, two questions arise: (i) which graph classes have a bounded extraction width, and (ii) how can extraction orders of minimal width be computed? In the following these questions are partially answered. Specifically, for the second question we only show the hardness of computing an extraction order of minimal width. Lastly, in Section 6.3.6 extensions of the approach presented here are discussed, yielding an interesting connection to the treewidth based algorithms presented in Section 6.4.

6.3.5.1 Graph Classes of Bounded Extraction Width

Given the impossibility of polynomial-time approximations for arbitrary request graphs, we now study graph classes that have bounded extraction width. In particular, we show that cactus graph requests and generalizations thereof have bounded extraction width.

Theorem 6.21. Consider a cactus request graph G_r, i.e., one for which cycles in its undirected interpretation intersect in at most a single node. Then $\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}}) \leq 2$ holds for *any* extraction order $G_r^{\mathcal{X}}$. Hence, for cactus request graphs solutions to the fractional offline VNEP can be computed in polynomial-time using Formulation 6.6. for cactus graph requests.

Proof. Consider any extraction order $G_r^{\mathcal{X}}$. $|\mathcal{L}_{r,e}^{\mathcal{X}}| \leq 1$ must hold for all edges $e \in E_r^{\mathcal{X}}$: if this was not the case then two confluences would overlap in e and violate the cactus property. Thus, edge label sets are either equal or disjoint and the maximal edge bag size is 1. ■

While the class of cactus graphs is restrictive, it can be shown that by adding edges parallel to existing ones the width increases by at most the maximum degree of the graph:

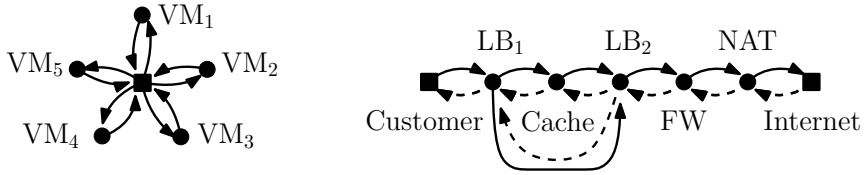


Figure 6.5: Repetition and slight adaptation of the exemplary virtual networks presented in Figure 1.3. Left: a virtual cluster abstraction envisioned in batch processing applications [Bal+11]. Right: a service chain envisioned in 5G networks [Nap+16], where the flows towards the Internet and the ones from the Internet are now drawn using solid and dashed lines, respectively.

Lemma 6.22. Given an arbitrary graph G_r, adding any number of parallel edges (of any direction) for an existing edge does increase the extraction width of G_r by at most the maximum degree of G_r. This also holds true if paths are added instead of edges.

Proof. Consider an extraction order $G_r^{\mathcal{X}}$ minimizing the width. Let $e = (i, j) \in E_r$ be an existing edge and assume without loss of generality that the orientation of the edge e is the same in $G_r^{\mathcal{X}}$. Now, when adding another edge $e' = (i, j)$ or $e'' = (j, i)$ to E_r, we orient the edge the same way as the original edge e. Hence, e' would be introduced to $E_r^{\mathcal{X}}$ as is, and the orientation of e'' would be reversed. Now, if the node j was previously not the target of a confluence, then by introducing e' or e'' a new confluence was created and the size of the edge bag of node i containing the edge $(i, j) \in E_r^{\mathcal{X}}$ increases by one. However, adding e' or e'' cannot introduce confluences beyond that: the addition of a *parallel* edge cannot enable a novel confluence to be created. Hence, arbitrarily many parallel edges can be added while increasing the size of an edge bag of the tail of the edge by at most one per outgoing edge. Hence, arbitrarily many parallel edges can be created for any existing edge while increasing the sizes of a single edge bag per node by at most one per outgoing edge. Hence, the extraction width of the resulting graph has increased by at most the maximum degree of the original graph G_r. ■

Lastly, in Figure 6.5 we revisit the example requests introduced in Chapter 1 and show in the following that these have small extraction widths.

Observation 6.23. The example request graphs depicted in Figure 6.5 have an extraction width of 2 (Virtual Cluster) and 3 (Service Chain) respectively.

Proof. The virtual cluster graph depicted on the left of Figure 6.5 is a cactus graph and hence has width 2 by Lemma 6.21.

Considering the request graph on the right, we note that when only considering the solid edges, the graph is a cactus and the width is hence 2. By adding the 'parallel' dashed edges, the extraction width increases by at most the maximum degree 3 of the cactus graph according to Lemma 6.22. However, considering the proof of Lemma 6.22, we see that for the node LB_1, only 2 outgoing edges exist (according to the solid edges) and hence the width increases by at most 2. Furthermore, as the node LB_2 is already the target of a confluence, the overall extraction width increases by 1 and the width is hence 3. ■

6.3.5.2 Hardness of Computing Extraction Orders.

Given the above examples, we now study the computational complexity of finding extraction orders of minimum width. In fact, we prove the $\mathcal{NP}$-hardness of computing optimal extraction orders. Our prove relies on a reduction from vertex cover. As a first step towards this goal, consider the following lemma.

Lemma 6.24. Consider the half wheel graph G_w depicted in Figure 6.6 and any extraction order $G_w^{\mathcal{X}}$ being rooted at w_c. Letting $\mathrm{VC} = \{w_k \in V_w \mid (w_{k-1}, w_k) \in E_r^{\mathcal{X}} \vee (w_{k+1}, w_k) \in E_r^{\mathcal{X}}\}$ denote the set of nodes which are the target of an edge in the extraction order $G_w^{\mathcal{X}}$, then the following holds: $\mathrm{ew}_{\mathcal{X}}(G_w^{\mathcal{X}}) = |\mathrm{VC}| + 1$.

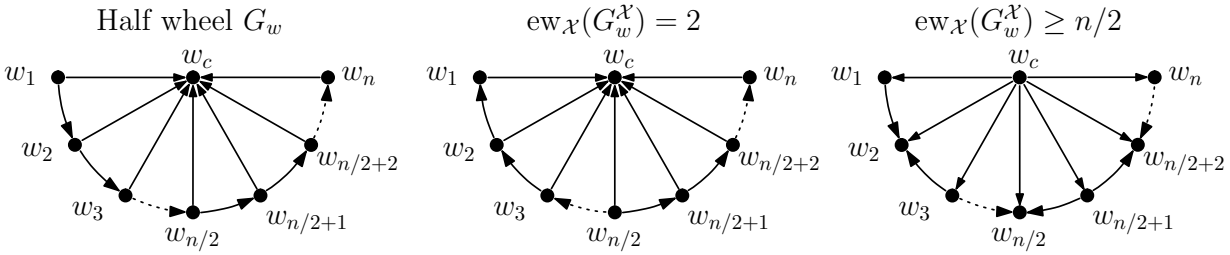


Figure 6.6: Depicted is an arbitrarily oriented 'half wheel' (left) together with two extraction orders: The extraction order in the center is rooted at $w_{n/2}$ and has width 2. The other order on the right is rooted at w_c and has a width of at least $n/2$ (shown below).

Proof. We denote by $G_w^i = (V_w^i, E_W^i)$ the subgraph of $G_w^{\mathcal{X}}$ induced by the set of nodes $\{w_1, \ldots, w_i\} \cup \{w_c\}$. Let $\mathrm{VC}_i = \{w_k \in V_w^i | (w_{k-1}, w_k) \in E_r^{\mathcal{X}} \vee (w_{k+1}, w_k) \in E_r^{\mathcal{X}}\}$.

Via induction over the subgraphs G_w^i it can be seen that the edges $e_k = (w_c, w_k)$, and $e_{k-1} = (w_c, w_{k-1})$ are either both labeled by w_k (if $w_k \in \mathrm{VC}_i$) or by $w_{k-1} \in \mathrm{VC}_i$ (if $w_k \notin \mathrm{VC}_i$) for all $k \in \{2, \ldots, i\}$.

Observing that $\mathrm{VC}_n = \mathrm{VC}$ equals the labels introduced in $G_w^{\mathcal{X}}$ and noting that the edge label sets $\mathcal{L}_{r,e_i}^{\mathcal{X}}$ and $\mathcal{L}_{r,e_{i+1}}^{\mathcal{X}}$ overlap for all $i \in \{1, \ldots, n-1\}$, the root w_c must have a single edge bag containing all the labels contained in $|\mathrm{VC}_i|$. Hence, $\mathrm{ew}_{\mathcal{X}}(G_w^{\mathcal{X}}) \geq |\mathrm{VC}| + 1$ follows. Furthermore, only the nodes contained in $|\mathrm{VC}|$ can be labels (a node not contained in VC has only a single incoming edge) and the result follows. ■

By Lemma 6.24, the following corollary is immediate.

Corollary 6.25. Consider a wheel graph G_w with n outer nodes. For any extraction order $G_w^{\mathcal{X}}$, for which the node w_c is chosen to be the root, $\mathrm{ew}_{\mathcal{X}}(G_w^{\mathcal{X}}) \geq \lfloor n/2 \rfloor + 1$ holds.

The result of Lemma 6.24 can be generalized in the following sense.

Lemma 6.26. Given is a connected, undirected graph $\bar{G} = (\bar{V}, \bar{E})$, we define a directed version $G_{VC} = (V_{VC}, E_{VC})$ with an additional super node $\hat{r}$ as follows: $V_{VC} = \bar{V} \sqcup \{\hat{r}\}$, and $E_{VC} = \{(i,j) | \{i,j\} \in \bar{E}, i < j\} \cup \{(\hat{r}, i) | i \in \bar{V}\}$. The minimal width of an extraction order $G_{VC}^{\mathcal{X}}$ rooted at $\hat{r}$ equals the size of the minimum vertex cover of $\bar{G}$ plus one.

Proof. Let $G_{VC}^{\mathcal{X}}$ be an extraction order of G_{VC} which is rooted at $\hat{r}$. The proof of Lemma 6.24 has shown that whenever a path P in the original graph is considered, all nodes of $G_{VC}^{\mathcal{X}}$ with at least one incoming edge (with respect to the original edge set) are labels of the *same* edge bag of the root $\hat{r}$. As this property holds for any simple path contained in $\bar{G}$ and as $\bar{G}$ is connected, there can only be a single edge bag: if there was more than one edge bag, then there does not exist a path P connecting any of the edges of the first bag to any of the edges in the second bag, refuting the connectivity of $\bar{G}$. Applying Lemma 6.24 for any path P of the original graph, the single edge bag of the root $\hat{r}$ must contain any node having at least one incoming edge according to the original edge set $\bar{E}$. Hence, assuming that $G_{VC}^{\mathcal{X}}$ has minimal width, the width of $G_{VC}^{\mathcal{X}}$ equals the size of the minimum vertex cover of $\bar{G}$ plus one. ■

Lemma 6.26 is the basis of our proof that computing extraction orders of minimal width is $\mathcal{NP}$-hard via a reduction from vertex cover (cf. Theorem 6.28). For the proof of our reduction, we require the following lemma.

Lemma 6.27. Consider a graph $G = (V, E)$ with a corresponding extraction order $G_r^{\mathcal{X}} = (V, E_r^{\mathcal{X}}, s)$. Assume that a node $v \in V$ exists that separates a set of nodes $U \subset V$ from the root node s. Then any edge incident to v and some node $u \in U$ is oriented away from v in the extraction order $G_r^{\mathcal{X}}$, i.e., $(v, u) \in E_r^{\mathcal{X}}$ holds for all $u \in U$.

Proof. Assume for the sake of contradiction that for some node $u \in U$ the edge (u, v) is contained in the extraction order. By the definition of the extraction order all nodes must be reachable from the root s. As the node v separates u from the root, all paths from s to u must contain v. Hence, u must be reachable from v and the edge (u, v) hence creates a loop in $G_r^{\mathcal{X}}$ which contradicts the acyclicity of $G_r^{\mathcal{X}}$. Hence, any edge incident to v and some node $u \in U$ must be directed away from v. ■

Theorem 6.28. Computing an extraction order of minimum width is $\mathcal{NP}$-hard.

Proof. We give a polynomial time reduction of the vertex cover problem to the problem of finding the extraction order of minimum width. We adapt the construction used in Lemma 6.26 slightly, to force the mapping of the root node to $\hat{r}$. Concretely, we add a *half wheel graph (cf. Figure 6.6)* G_w with $2 \cdot |\bar{V}| + 2$ outer nodes to the graph G_{VC} *and identify* the node $\hat{r}$ with the wheel's node w_c, i.e., $\hat{r} = w_c$. Let $G_{VC}^{\mathcal{X}} = (V_{VC}, E_{VC}^{\mathcal{X}}, s_{VC})$ be an extraction order of minimum width. The extraction order's root s_{VC} must be placed on some outer wheel node, as the following case analysis shows.

Root s_{VC} is placed on a wheel node w_i: We first consider the orientations of edges inside the wheel graph. According to Figure 6.6 (center) there is an orientation such that the extraction width inside the wheel graph is 2. The node $\hat{r} = w_c$ separates the original graph $\bar{G}$ from the extraction order's root $s_{VC} = w_i$. Thus, by Lemma 6.27 all edges incident to a node $v \in \bar{V}$ and w_c must be oriented away from w_c. Hence, excluding the outer wheel nodes, the node w_c is a root in the corresponding extraction order. Thus, the width of the extraction order $G_{VC}^{\mathcal{X}}$ – excluding the outer wheel nodes – equals the size of a minimum vertex cover of $\bar{G}$ plus one by Lemma 6.26 and the assumption that $G_{VC}^{\mathcal{X}}$ is of minimal width. Lastly, note that no confluence spanning the wheel graph G_w and the graph $\bar{G}$ exists. Letting VC denote a minimal vertex cover of $\bar{G}$, the width of the extraction order $G_{VC}^{\mathcal{X}}$ equals $\max\{2, |VC|\} \leq |\bar{V}|$.

Root s_{VC} is placed on $\hat{r} = w_c$: In this case, the width of the extraction order $G_{VC}^{\mathcal{X}}$ is at least $|\bar{V}|+1$ based on Corollary 6.25. Hence, as the size of a vertex cover of $\bar{G}$ is always less than $|\bar{V}|$, the placement of the root on $\hat{r}$ contradicts the optimality assumption of $G_{VC}^{\mathcal{X}}$.

Root s_{VC} is placed on a node $v \in \bar{V}$: In this case, the width is again at least $|\bar{V}| + 1$: as the node $\hat{r} = w_c$ separates the outer wheel nodes from the root s_{VC}, all wheel edges incident to w_c are oriented away from w_c. As w_c is hence a root in the extraction order restricted to the wheel graph, the width is at least $|\bar{V}| + 1$ by Lemma 6.24. Thus, the placement of the extraction's root on any node $v \in \bar{V}$ contradicts the optimality of the extraction order $G_{VC}^{\mathcal{X}}$.

Now, let $VC \subseteq \bar{V}$ denote a minimum vertex cover of $\bar{G}$. If $|VC| > 1$ holds, then for any optimal extraction order $G_{VC}^{\mathcal{X}}$, $\mathrm{ew}_{\mathcal{X}}(G_{VC}^{\mathcal{X}}) = |VC|+1$ holds. Furthermore, a minimal vertex cover VC can be recovered from any minimum width extraction order $G_{VC}^{\mathcal{X}}$ by placing any node v in the cover VC whenever at least two edges are oriented towards it in $G_{VC}^{\mathcal{X}}$. As the cases in which the minimal vertex cover is less or equal to 1 can be trivially identified, computing a minimum width extraction order is $\mathcal{NP}$-hard. ■

6.3.6 Concluding Remarks and Known Extraction Width Extensions

The extraction width approach presented in this section is a natural one, considering the flow-oriented definition of extraction orders. While enabling the computation of fractional offline VNEP solutions for arbitrary request graphs, the involved definition of the extraction width renders computing extraction orders of small width hard: only once all edges are oriented in an extraction order, the extraction width can be computed. Besides the lack of suitable algorithms to efficiently compute good extraction orders, the definition of confluence edge bags can also be improved as discussed in the following.

Consider the half wheel graph depicted in Figure 6.6 again and imagine two of these half wheels with n nodes being connected at the center node. In this setting, it is easy to verify that the resulting graph's extraction width is lower bounded by $\Omega(n/2)$: to minimize the extraction width of a single half wheel

graph, the root must be placed on one of its outer nodes; this however implies that the center node of the other graph is a local root in the respective half wheel graph, leading to large extraction widths.

The above example shows a surprising property: while the respective center nodes are separators for the other graphs, joining these has a severe impact. In the master thesis of Döhne [Döh18] several proposals to improve the extraction width approach have been presented. In particular, one very promising approach of Döhne is to redefine the confluence edge bags: instead of simply partitioning the outgoing edges, a hierarchy of labels is introduced (for each confluence edge bag) and the problem of optimally computing these hierarchical label set orders is studied. Based on an interesting connection to join trees and the running intersection property, Döhne shows that minimizing the respective *label set order width* (locally for a node) equals the minimization of the *treewidth* on an graph induced by the label set ordering. Inspired by this finding, solution approaches for the fractional VNEP purely relying on the concept of tree decompositions came into focus. In particular, the results presented next in Section 6.4 are a direct consequence of this line of research and we see several advantages of the presented approach relying purely on tree decompositions: tree decompositions and the notion of treewidth have been studied extensively (see Section 3.3 for an overview) and efficient exact algorithms are known for reasonable request sizes (cf. Evaluation in Section 7.6.3). Nevertheless, pursuing research based on the notion of extraction orders might be worthwhile nonetheless, as related concepts might offer the potential of being faster to solve for some instances.

6.4 LP Formulations Based on Tree Decompositions

While the extraction width LP formulation yields a first XP-algorithm for the fractional VNEP (cf. Theorem 6.20), the inability to efficiently compute the extraction width hinders the application of this result. We now turn towards another parametrized approach. In particular, similar to the extraction order approach, tree decompositions of request graphs are used, which will enable the application of dynamic programming to solve the VMP and in turn will yield an effective separation oracle for the enumerative LP Formulations 6.1 and 6.2. We show the following:

1. We develop the dynamic programming algorithm DynVMP to solve the VMP, which runs in XP-time $\mathcal{O}(|V_S|^{\mathsf{poly}(\mathrm{tw}(G_r))} \cdot \mathsf{poly}(|V_S| \cdot |V_r|))$. Thus, for graphs of bounded treewidth DynVMP runs in polynomial-time.
2. Based on Linear Programming duality, we show that the DynVMP algorithm can be used as *separation oracle* and derive an efficient *column generation* approach for solving the LP Formulations 6.1 and 6.2.
3. With only minor adaptions, we show that the DynVMP algorithm can also be used to solve the VMP approximately when considering latency constraints.

6.4.1 Tree Decompositions of Request Graphs

Since tree decompositions are defined for undirected graphs, we employ undirected interpretations of requests:

Definition 6.29 (Undirected Request Graph $\overline{G}_r$)**.** For a request graph $G_r = (V_r, E_r)$ its undirected interpretation $\overline{G}_r = (V_r, \overline{E}_r)$ is given by $\overline{E}_r = \{\{i, j\} | (i, j) \in E_r\}$ on the original node set V_r. □

Note that directed, antiparallel edges $(i, j), (j, i) \in E_r$ of the original request graph are accordingly represented using only a single undirected edge $\{i, j\} \in \overline{E}_r$. While tree decompositions were introduced in Definition 3.7, we slightly adapt the notation for the consideration of request graphs:

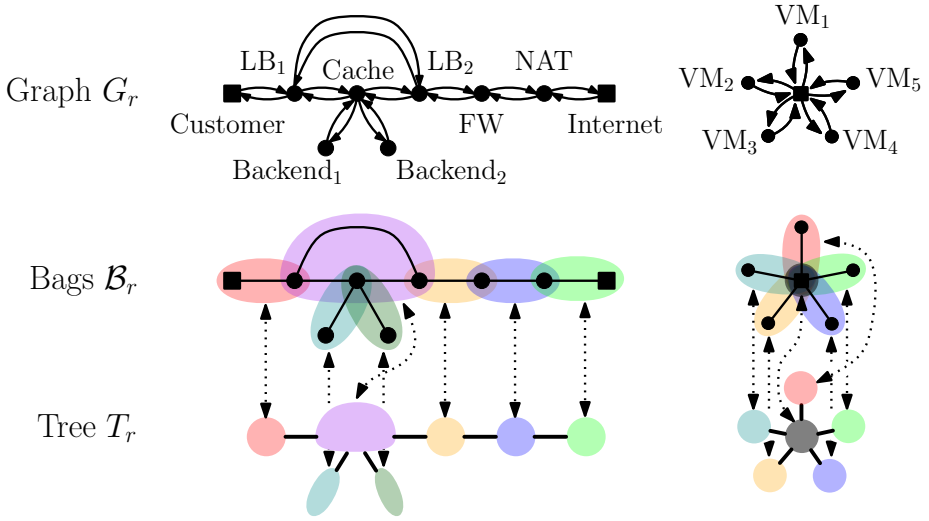


Figure 6.7: Depicted are two exemplary virtual network request graphs together with corresponding tree decompositions: a load-balancing service chain and a virtual cluster with 5 VMs. The *covering* node bags are depicted in the middle, while the resulting trees are depicted on the bottom. The tree decomposition on the left has a width of 2, while the width of the virtual cluster is 1, as it is a tree.

Definition 6.30 (Request Tree Decomposition $\mathcal{T}_r = (T_r, \mathcal{B}_r)$)**.** Given an undirected request graph $\overline{G}_r = (V_r, \overline{E}_r)$, a tree decomposition of $\overline{G}_r$ is a pair $\mathcal{T}_r = (T_r, \mathcal{B}_r)$ consisting of an undirected tree $T_r = (V_T, E_T)$ and a family $\mathcal{B}_r = \{B_t\}_{t \in V_T}$ of subsets $B_t \subseteq V_T$, also referred to as the node bags, for which the following conditions hold:

(1) For all request nodes $i \in V_r$, the set $V_T^{-1}(i) = \{t \in V_T \mid i \in B_t\}$ of tree nodes containing node i is connected in T_r.

(2) Each request node and each (undirected) request edge is contained in at least one of the bags: $\forall i \in V_r. \exists t \in V_T : i \in B_t$ and $\forall \{i,j\} \in \overline{E}_r. \exists t \in V_T : \{i,j\} \subseteq B_t$ hold. □

As noted in Section 3.3, several important graph classes are known to have small treewidths. In Figure 6.7, tree decompositions of the example requests considered throughout this thesis are given: the virtual cluster has a treewidth 1 and the service chain has a treewidth of 2. This also follows from the study of the respective graph classes, as the service chain is outerplanar and the virtual cluster is a tree. However, even if a request graph does not belong to a graph class of bounded treewidth, recent exact algorithms can often compute optimal tree decompositions in a matter of seconds [Tam17].

A tree decomposition naturally groups request nodes together into node bags. As the size of each bag is bounded for graphs of bounded treewidth, this allows to perform more complex operations on the whole bag *in polynomial-time*. In particular, instead of mapping single virtual nodes, we will consider the joint mappings of all request nodes contained in the bags. While the number of mapping possibilities grows exponentially in the node bag's size, it is *polynomial* for graphs of bounded treewidth. Concretely, the number of mapping possibilities for a node bag B_t equals $\prod_{i \in B_t} |V_S^{r,i}| \in \mathcal{O}(|V_S|^{\mathrm{tw}(\mathcal{T}_r)+1})$. We again mathematically represent the space of node bag mappings as follows. We denote by $\mathcal{M}(B_t) = [B_t \to V_S]$ the set of *all* functions from B_t to V_S, i.e., $m_t^V \in \mathcal{M}(B_t)$ maps all virtual nodes of B_t. Given a specific bag mapping $m_t^V \in \mathcal{M}(B_t)$, a cost-optimal valid mapping of the subgraph $G_r[B_t] = (B_t, E_r[B_t])$ induced by B_t, i.e., $E_r[B_t] = \{(i,j) \in E_r \mid i,j \in B_t\}$, is computable in polynomial-time:

Lemma 6.31 (Computation of optimal induced mappings)**.** Given a node bag mapping $m_t^V \in \mathcal{M}(B_t)$, one can check in time $\mathcal{O}(\mathsf{poly}(|B_t| \cdot |G_S|))$ if a valid edge mapping extension m_t^E exists, such that $m_t = (m_t^V, m_t^E)$

is a valid mapping of the induced subgraph $G_r[B_t]$. Furthermore, if such an induced valid mapping exists, the least cost one can be computed in time $\mathcal{O}(\mathsf{poly}(|B_t| \cdot |G_S|))$.

Proof. The validity of the given node mapping m_t^V can be checked by testing whether $m_t^V(i) \in V_S^{r,i}$ holds for each virtual node $i \in B_t$. As the node mappings are fixed, one can compute a shortest *valid* path for each edge $(i,j) \in E_r[B_t]$ by applying e.g. Dijkstra's algorithm, albeit only considering substrate edges contained in $E_S^{r,i,j}$. If valid paths exist for all induced edges $E_r[B_t]$ under the node mapping m_t^V, a cost-optimal edge mapping m_t^E is obtained and otherwise no valid mapping can exist. ■

Besides the above lemma, we employ the following facts for our algorithm.

Fact 6.32 ([FG06])**.** Let $\mathcal{N}(t) \subseteq V_T$ denote the neighboring tree nodes of $t \in V_T$. For any tree node $t \in V_T$ and any pair $t_1, t_2 \in \mathcal{N}(t)$ of neighbors of t with $t_1 \neq t_2$, the following holds: $(B_{t_1} \cap B_{t_2}) \setminus B_t = \emptyset$.

Fact 6.33 ([FG06])**.** Any tree decomposition can be transformed into a *small* one for which $B_{t_1} \not\subseteq B_{t_2}$ holds for all $t_1, t_2 \in V_T$ with $t_1 \neq t_2$. For any small tree decomposition $|V_T| = |\mathcal{B}_r| \leq |V_r|$ holds.

The first fact states that node bags *separate* neighboring node bags from each other, while the second allows to bound the size of the tree $|V_T|$ by the number of original request nodes $|V_r|$.

The following additional notation will be used throughout this section. We employ $B_{t_1 \cap t_2}$ to denote the intersection of the corresponding node bags, i.e., $B_{t_1 \cap t_2} = B_{t_1} \cap B_{t_2}$. Given a node bag mapping m_t^V, we again denote by $\langle m_t^V | V_r' \rangle : V_r' \to V_S$ the restriction of m_t^V to a subset $V_r' \subseteq B_t$, such that $\langle m_t^V | V_r' \rangle(i) = m_t^V(i)$ for $i \in V_r'$.

6.4.2 Dynamic Program DynVMP

We now present the XP-algorithm DynVMP for solving the VMP. We first consider the VMP without latency constraints and afterwards present a minor extension to cater for latencies. The algorithm uses the tree decomposition $\mathcal{T}_r$ of the request graph and applies dynamic programming: starting from the leaves of the tree decomposition, (partial) cost-optimal valid mappings are constructed bottom-up. This is facilitated by Lemma 6.31: for each single node bag mapping $m_t^V \in \mathcal{M}(B_t)$, a cost-optimal induced mapping can be computed in time $\mathcal{O}(\mathsf{poly}(|B_t| \cdot |G_S|))$. Starting at the leaves, these cost-optimal valid mappings are then combined in a bottom-up fashion. Concretely, the algorithm stores for each tree node $t \in V_T$ and each node bag mapping $m_t^V \in \mathcal{M}(B_t)$ the optimal mapping costs in the table $\mathbb{C}[t][m_t^V]$ (infinite costs indicate infeasibility) together with the node mappings in table $\mathbb{M}[t][m_t^V]$ (see Lines 2-4).

The nodes of the tree decomposition are then traversed bottom-up (post-order traversal). Considering a specific tree node $t \in V_T$ with node bag B_t, all node bag mappings $m_t^V \in \mathcal{M}(B_t)$ are enumerated (Line 7). Only if the induced mapping is valid, the mapping is considered and otherwise the corresponding cost $\mathbb{C}[t][m_t^V]$ stays infinite (indicating invalidity). Considering leaves, the (induced) mapping costs of locally valid mappings can be readily computed using Lemma 6.31 by the `InducedCost` function. For nodes having children, the current mapping m_t^V is sought to be extended as cheaply as possible. To this end, all suitable child mappings $m_{t_c}^V \in \mathcal{M}(B_{t_c})$ agreeing with the current mapping m_t^V are considered and according to the cost-optimal one the mapping table $\mathbb{M}[t][m_t^V]$ is updated. Importantly, the different children will never set a mapping of a virtual node $i \in V_r$ twice by Fact 6.32: a request node i is either contained in only a single child bag or in multiple; however, if it is contained in multiple bags, then it must be contained in B_t. Accordingly, if $i \in B_t$ holds, then the mapping of i is already explicitly set by m_t^V and the child mappings cannot disagree on the mapping of i, as only matching mappings were selected in Line 12. Only if for all children valid mappings exist, the cost is updated and otherwise the mapping is considered to be invalid (cf. Lines 23 and 24). Having processed the whole tree, the optimal valid mapping is retrieved at the root node $\hat{t}_r$ or $\perp$ is returned to indicate that none exists.

Algorithm 6.8: DynVMP: Computing Optimal Valid Mappings

Input : substrate G_S, request G_r, tree decomposition $\mathcal{T}_r$
Output: valid mapping of minimal cost or $\perp$ if none exists

1 `PrecomputeShortestValidPaths(`G_r,G_S`)`
2 **foreach** $t \in V_T$ **do** `// initialize tables`
3 **foreach** $m_t^V \in \mathcal{M}(B_t)$ **do**
4 $\mathbb{C}[t][m_t^V] \leftarrow \infty$ **and** $\mathbb{M}[t][m_t^V] \leftarrow (i \mapsto \perp \mid i \in V_r \setminus B_t)$
5 **set** $\mathcal{Q} \leftarrow$ `PostOrderTraversal`$(T_r, \hat{t}_r)$
6 **for** $t \in \mathcal{Q}$ **do** `// traverse tree in post-order`
7 **for** $m_t^V \in \mathcal{M}(B_t)$ **do** `// consider node bag mappings`
8 **if** `InducedMappingLocallyValid`(m_t^V) **then**
9 **set** *children_valid* $\leftarrow$ True
10 **for** $(t, t_c) \in \delta^+(t)$ **do** `// find best child mapping` $\hat{m}_{t_c}^V$
11 **set** $\hat{m}_{t_c}^V \leftarrow \perp$
12 **for** $\begin{pmatrix} m_{t_c}^V \in \mathcal{M}(B_{t_c}) \text{ with} \\ \langle m_{t_c}^V | B_{t_c \cap t} \rangle = \langle m_t^V | B_{t_c \cap t} \rangle \end{pmatrix}$ **do**
13 **if** $\hat{m}_{t_c}^V = \perp$ ***or*** $\mathbb{C}[t_c][m_{t_c}^V] < \mathbb{C}[t_c][\hat{m}_{t_c}^V]$ **then**
14 $\hat{m}_{t_c}^V \leftarrow m_{t_c}^V$
15 **if** $\hat{m}_{t_c}^V \neq \perp$ **then** `// if valid mapping exists`
16 **for** $i \in V_r \setminus B_t$ **do** `// as` m_t^V `fixes` B_t `mapping`
17 **if** $i \in B_{t_c}$ **then** `// as` $m_{t_c}^V$ `maps` i
18 $\mathbb{M}[t][m_t^V](i) \leftarrow \hat{m}_{t_c}^V(i)$
19 **else if** $\mathbb{M}[t_c][\hat{m}_{t_c}^V](i) \neq \perp$ **then**
20 $\mathbb{M}[t][m_t^V](i) \leftarrow \mathbb{M}[t_c][\hat{m}_{t_c}^V](i)$
21 **else** `// induced valid mapping cannot exist`
22 **set** *children_valid* $\leftarrow$ False **and exit for-loop**
23 **if** *children_valid* **then**
24 $\mathbb{C}[t][m_t^V] \leftarrow$ `InducedCost`$(m_t^V \cup \mathbb{M}[t][m_t^V])$
25 **choose** $\hat{m}_{\hat{t}_r}^V \in \mathcal{M}(B_{\hat{t}_r})$ **such that** $\hat{c} \leftarrow \mathbb{C}[\hat{t}_r][\hat{m}_{\hat{t}_r}^V]$ **is minimal**
26 **if** $\hat{c} < \infty$ **then return** `InducedMapping`$(\hat{m}_{\hat{t}_r}^V \cup \mathbb{M}[\hat{t}_r][\hat{m}_{\hat{t}_r}^V])$
27 **else return** $\perp$

Theorem 6.34. The DynVMP algorithm correctly determines whether a valid mapping exists and if so, returns a cost-optimal one. Its runtime is bounded by $\mathcal{O}(|V_r|^3 \cdot |V_S|^{2 \cdot \mathrm{tw}(\mathcal{T}_r)+3})$.

Proof. By the above description of the algorithm, the algorithm returns an optimal valid mapping, if one exists. With respect to the runtime, we first note that $|V_T| \leq |V_r|$ holds when considering small tree decompositions (cf. Fact 6.33). The pre-computation of all shortest valid paths can be implemented in time $\mathcal{O}(|V_r|^2 \cdot |V_S|^3)$ by applying Dijkstra's algorithm for each of the $\mathcal{O}(|V_r|^2)$ request edges for each potential substrate start node. On the other hand, the runtime of the Lines 12 to 22 dominate the main algorithm's runtime. Here, for each of the at most $|V_r|$ tree nodes at most $|V_S|^{\mathrm{tw}(\mathcal{T}_r)+1}$ many mappings m_t^V are considered, for which again at most $|V_r| \cdot |V_S|^{\mathrm{tw}(\mathcal{T}_r)+1}$ many mappings of its children must be considered while adapting the mappings in Lines 17 to 20 may again take $\mathcal{O}(|V_r|)$ time, yielding the claimed overall runtime. ∎

Lastly, we show that the DynVMP algorithm can be used to approximate the cost of valid mappings

under latency constraints. While computing minimum-cost latency-constrained shortest paths (LCSP) is itself an $\mathcal{NP}$-hard problem, a fully polynomial-time approximation scheme exists:

Theorem 6.35 (LCSP FPTAS, Lorenz and Raz [LR01])**.** For any $\varepsilon_{\mathrm{L}} > 0$, an $(1+\varepsilon_{\mathrm{L}})$-optimal path satisfying the latency bound can be computed in time $\mathcal{O}\big(|E_S| \cdot |V_S| \cdot (\log\log|V_S| + 1/\varepsilon_{\mathrm{L}})\big) = \mathsf{time}_{\mathrm{LCSP}}(\varepsilon_{\mathrm{L}})$.

The FPTAS for the LCSP can be used in the DynVMP algorithm to compute approximate latency respecting valid paths in Line 1. As each computed path is $(1+\varepsilon')$-optimal, the resulting mapping is also $(1+\varepsilon')$-optimal and we obtain the following result:

Lemma 6.36. Using the LCSP FPTAS, the DynVMP algorithm finds a $(1+\varepsilon_{\mathrm{L}})$-optimal valid mapping, if one exists. Its runtime is bounded by $\mathcal{O}(|V_r|^2 \cdot (|V_r| \cdot |V_S|^{2\cdot\mathrm{tw}(\mathcal{T}_r)+2} + \mathsf{time}_{\mathrm{LCSP}}(\varepsilon_{\mathrm{L}})))$.

6.4.3 Solving the Fractional Offline VNEP via Column Generation

In the following, we now show how the LP Formulations 6.1 and 6.2 can be solved efficiently for *arbitrary request graphs* by using the DynVMP algorithm. While these primal formulations use exponentially many variables, the respective dual formulations (cf. Formulations 6.9 and 6.10) use a polynomial number of variables $\vec{\lambda}$ (corresponding to Constraints 6.2 and Constraints 6.6, respectively) and $\vec{\mu}$ (corresponding to Constraints 6.3 and 6.7, respectively) while employing exponentially many constraints. However, it is known that such LP formulations can be solved in polynomial-time, as long as *violated* constraints can be identified in polynomial-time by a separation oracle. We will consider the case without latencies first.

Separation in the Absence of Latency Constraints. We start by considering the dual of the *profit*, i.e., Formulation 6.9. The Constraints 6.36 can be separated using DynVMP as follows. First, we interpret the $\vec{\mu}$ variables as resource costs and accordingly define $c_{S,\mu} : G_S \to \mathbb{R}_{\geq 0}$ with $c_{S,\mu}(x) = \mu_x$. Hence, the mapping cost $c_{S,\mu}(m_r)$ of a valid mapping m_r equals $\sum_{x \in G_S} \mu_x \cdot A(m_r, x)$, i.e., the second term of the left-hand side of Constraint 6.36. Accordingly, the DynVMP algorithm can be used to compute cost-optimal mappings $\hat{m}_r$ for each request $r \in \mathcal{R}$. If $c_{S,\mu}(\hat{m}_r) \geq b_r - \lambda_r$ holds, then all valid mappings of request r satisfy Constraint 6.36. On the other hand, if $c_{S,\mu}(\hat{m}_r) < b_r - \lambda_r$ holds, then the constraint corresponding to the mapping $\hat{m}_r$ is added to the Linear Program 6.9. Notably, the initial primal containing no mappings is valid and by setting (arbitrarily) $\vec{\lambda} = \vec{\mu} = \vec{0}$ the first mappings (columns) can be generated. Iteratively separating the violated constraints then as long as one exists, an optimal LP solution can be computed.

For practical applications, the following lemma is helpful in terminating the separation process once a solution of sufficient quality has been found:

Lemma 6.37 (Approximate Solutions to the Profit LP)**.** Let $\vec{\mu}, \vec{\lambda}$ be the dual variables corresponding to a primal solution of the profit LP Formulation 6.1 and let $\epsilon > 0$. If $c_{S,\mu}(m_r) \cdot (1+\epsilon) \geq b_r - \lambda_r$ holds for all $m_r \in \mathcal{M}_r$ and each $r \in \mathcal{R}$, then the primal LP solution is $1/(1+\epsilon)$-optimal.

Proof. This follows from *weak duality*, as scaling the $\vec{\mu}$ variables by a factor of $(1+\epsilon)$ yields a feasible dual solution while increasing the objective by at most a factor $(1+\epsilon)$. ■

The separation process for the dual of the cost variant (cf. Formulation 6.10) can be implemented very similarly. One notable difference is that constructing a feasible primal LP solution is not trivial, as each request must be fully embedded and the capacity constraints may not be violated. However, using the above result for the profit variant such a solution can be readily computed by introducing profits $b_r = 1$ for each request $r \in \mathcal{R}$ and solving the profit Formulation 6.1 optimally. If the obtained profit is less than $|\mathcal{R}|$, then no feasible solution can exist for the cost optimization task. On the other hand, if the obtained profit equals $|\mathcal{R}|$, the generated solution can be used to initialize the primal LP solution for the cost optimization.

LP Formulation 6.9: Dual Enumerative Formulation for the Profit Offline VNEP

$$\min \sum_{r\in\mathcal{R}} \lambda_r + \sum_{x\in G_S} \mu_x \cdot d_S(x) \tag{6.35}$$

$$\lambda_r + \sum_{x\in G_S} \mu_x \cdot A(m_r^k, x) \geq b_r \qquad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \tag{6.36}$$

$$\lambda_r \geq 0 \qquad \forall r \in \mathcal{R} \tag{6.37}$$

$$\mu_x \geq 0 \qquad \forall x \in G_S \tag{6.38}$$

LP Formulation 6.10: Dual Enumerative Formulation for the Cost Offline VNEP

$$\max \sum_{r\in\mathcal{R}} \lambda_r + \sum_{x\in G_S} \mu_x \cdot d_S(x) \tag{6.39}$$

$$\lambda_r + \sum_{x\in G_S} \mu_x \cdot A(m_r^k, x) \leq c_S(m_r^k) \qquad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \tag{6.40}$$

$$\lambda_r \in \mathbb{R} \qquad \forall r \in \mathcal{R} \tag{6.41}$$

$$\mu_x \leq 0 \qquad \forall x \in G_S \tag{6.42}$$

Having argued that a primal feasible solution can always be found for the cost variant, we now discuss the separation procedure to optimize the costs. First note that the dual for the cost subtly varies from the dual for the profit variant: the variables have different domains and the Constraints 6.40 have a different right-hand side as well as a different sense. We begin by transforming Constraint 6.40 equivalently as follows:

$$\lambda_r + \sum_{x\in G_S} \mu_x \cdot A(m_r^k, x) \leq c_S(m_r^k) \tag{6.43}$$

$$\Leftrightarrow \quad -c_S(m_r^k) + \sum_{x\in G_S} \mu_x \cdot A(m_r^k, x) \leq -\lambda_r \tag{6.44}$$

$$\Leftrightarrow \quad -\sum_{x\in G_S} c_S(x) \cdot A(m_r^k, x) + \sum_{x\in G_S} \mu_x \cdot A(m_r^k, x) \leq -\lambda_r \tag{6.45}$$

$$\Leftrightarrow \quad \sum_{x\in G_S} (c_S(x) - \mu_x) \cdot A(m_r^k, x) \geq \lambda_r \tag{6.46}$$

The Equation 6.45 follows directly from the definition of the cost of a mapping (cf. Definition 2.8). Observing that $\mu_x \leq 0$ holds for the dual of the cost, we can now define new resource costs $c_{S,\mu} : G_S \to \mathbb{R}_{\geq 0}$ with $c_{S,\mu}(x) = c_S(x) - \mu_x$. Given these positive costs, the DynVMP algorithm can again be employed to separate the respective constraints as described above, and an optimal LP solution for the dual of the cost can be computed. Again, also approximate solutions can be computed for the cost variant:

Lemma 6.38 (Approximate Solutions to the Cost LP)**.** Let $\vec{\mu}, \vec{\lambda}$ be the dual variables corresponding to a primal solution of the cost LP Formulation 6.2 and let $\epsilon > 0$. If $c_{S,\mu}(m_r) \cdot (1+\epsilon) \geq \lambda_r$ holds for all $m_r \in \mathcal{M}_r$ and each $r \in \mathcal{R}$, then the primal LP solution is $(1+\epsilon)$-optimal.

Proof. This follows from *weak duality*, as scaling the $\vec{\lambda}$ variables by a factor of $1/(1+\epsilon)$ yields a feasible dual solution while decreasing the objective by at most a factor $1/(1+\epsilon)$. ■

Given the ability to solve the primal LP Formulation 6.2 optimally in the absence of latency constraints, we obtain the following result:

Theorem 6.39. Using the column generation approach to solve LP Formulations 6.1 and 6.2 using the DynVMP algorithm as separation oracle, the fractional offline VNEP can be solved optimally in the setting $\langle \mathbf{VE} \,|\, \mathbf{NR} \rangle$ in time $\mathcal{O}\left(\mathsf{poly}\left(\sum_{r\in\mathcal{R}} |V_r|^3 \cdot |V_S|^{2\cdot \mathrm{tw}(\mathcal{T}_r)+3}\right)\right)$ given specific tree decompositions $\mathcal{T}_r$ for each request $r \in \mathcal{R}$.

Proof. The runtime of the column generation LP is polynomial in the runtime of the separation oracle. Accordingly, the result follows as solutions to the respective LPs naturally model the respective fractional offline VNEP variants. ■

Separation in the Presence of Latency Constraints. Lastly, we turn towards computing solutions to the fractional offline VNEP under latency constraints. In this case, the adapted DynVMP algorithm using the LCSP FPTAS is employed to *approximately* separate the Constraints 6.36 *with latencies*. Again, we consider the profit variant first. For each request $r \in \mathcal{R}$ a $(1+\varepsilon_{\mathrm{L}})$-optimal mapping $\tilde{m}_r$ is computed during the column generation process and respective columns are added as long as $c_{S,\mu}(\tilde{m}_r) < b_r - \lambda_r$ holds. After the separation procedure did not return any more new columns – due to the approximate nature of the separation – some of the Constraints 6.36 might still be violated. In fact, only $c_{S,\mu}(m_r^k) \cdot (1+\varepsilon_{\mathrm{L}}) \geq b_r - \lambda_r$ holds for all mappings $m_r^k \in \mathcal{M}_r$. However, by Lemma 6.37, we obtain that the respective solution is $(1+\varepsilon_{\mathrm{L}})$-optimal and we state the following theorem:

Theorem 6.40. When considering the VNEP setting $\langle \mathbf{VE} \,|\, \mathbf{NRL} \rangle$, $1/(1+\varepsilon_{\mathrm{L}})$-approximate solutions to the profit variant of the fractional offline VNEP can be computed for any $\varepsilon_{\mathrm{L}} > 0$ in time $\mathcal{O}\left(\mathsf{poly}\left(\sum_{r\in\mathcal{R}} |V_r|^2 \cdot \left(|V_r| \cdot |V_S|^{2\cdot \mathrm{tw}(\mathcal{T}_r)+2} + \mathsf{time}_{\mathrm{LCSP}}(\varepsilon_{\mathrm{L}})\right)\right)\right)$ by solving the LP Formulation 6.1 using the adapted DynVMP algorithm, employing the LCSP FPTAS to approximate latency-obeying paths (cf. Lemma 6.36), as separation oracle.

Proof. The runtime of the column generation LP is polynomial in the runtime of the separation oracle and by the above arguments, the approximation factor of the separation oracle (cf. Lemma 6.36) carries over to the solution of the profit variant of the fractional offline VNEP. ■

Considering the column generation process for the cost variant essentially the same holds true. Before

Formulation 6.11: Primal Enumerative LP for Warmstarting the Primal Cost LP

$$\max \mathcal{X} \tag{6.47}$$

$$\sum_k f_r^k = \mathcal{X} \qquad \forall r \in \mathcal{R} \tag{6.48}$$

$$\sum_k f_r^k \cdot A(m_r^k, x) \leq d_S(x) \qquad \forall x \in G_S \tag{6.49}$$

$$\mathcal{X} \leq 1 \tag{6.50}$$

$$f_r^k \geq 0 \qquad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \tag{6.51}$$

$$\mathcal{X} \geq 0 \qquad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \tag{6.52}$$

Formulation 6.12: Dual Enumerative LP for Warmstarting the Primal Cost LP

$$\nu + \sum\nolimits_{x\in G_S} \mu_x \cdot d_S(x) \tag{6.53}$$

$$\lambda_r + \sum\nolimits_{x\in G_S} \mu_x \cdot A(m_r^k, x) \geq 0 \qquad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \tag{6.54}$$

$$\nu - \sum_{r\in\mathcal{R}} \lambda_r \geq 1 \tag{6.55}$$

$$\lambda_r \in \mathbb{R} \qquad \forall r \in \mathcal{R} \tag{6.56}$$

$$\mu_x \geq 0 \qquad \forall x \in G_S \tag{6.57}$$

$$\nu \geq 0 \tag{6.58}$$

discussing the separation procedure, we note that computing an initially feasible primal LP solution is slightly more challenging, as the primal profit LP can only be solved approximately. While the profit of the profit LP is lower bounded by $b_r/(1+\varepsilon_{\mathrm{L}})$, one can in general not establish a lower bound for the cumulative weights of each request as Lemma 6.37 only pertains to the whole objectives.

We propose a more involved LP approach to generate primal solutions in this setting. Concretely, we seek to construct a solution for which the cumulative weight $\sum_k f_r^k$ is lower bounded by some value for each request $r \in \mathcal{R}$. Specifically, we propose the usage of the LP Formulations 6.11 (primal) and 6.12 (dual). Compared to the original profit LP, the 'warmstart' LP has to embed each request *to the same extent* $\mathcal{X} \in [0,1]$. Considering the task to separate Constraint 6.54, we note that this is essentially equal to the previously studied separation task of the profit LP. Furthermore, the lemma on approximate solutions (cf. Lemma 6.37) still holds. Accordingly, $(1+\varepsilon_{\mathrm{L}})$-approximate solutions to the primal LP Formulation 6.11 can be computed. When $\mathcal{X} < 1/(1+\varepsilon_{\mathrm{L}})$ holds, then provably no feasible primal LP solution can exist for the cost LP. If on the other hand $\mathcal{X} \geq 1/(1+\varepsilon_{\mathrm{L}})$ holds, then by scaling the solution by the factor $1/\mathcal{X}$, i.e., scaling each weight f_r^k for each request and each mapping $m_r^k \in \mathcal{M}_r$ by $1/\mathcal{X}$, a feasible primal LP solution for the cost optimization is obtained which violates capacity constraints by at most a factor $(1+\varepsilon_{\mathrm{L}})$. Increasing the substrate capacities accordingly, the primal cost LP Formulation 6.2 can be warmstarted.

Considering the separation process to optimize this initially constructed LP solution we note the following. For each request $r \in \mathcal{R}$ a $(1+\varepsilon_{\mathrm{L}})$-optimal mapping $\tilde{m}_r$ is computed during the column generation process and respective columns are added as long as $c_{S,\mu}(\tilde{m}_r) < \lambda_r$ holds. After the separation procedure did not return any more new columns – due to the approximate nature of the separation – some of the Constraints 6.40 might still be violated. In fact, only $c_{S,\mu}(m_r^k) \cdot (1+\varepsilon_{\mathrm{L}}) \geq \lambda_r$ holds for all mappings $m_r^k \in \mathcal{M}_r$. However, now by Lemma 6.38, we obtain that the respective solution is $(1+\varepsilon_{\mathrm{L}})$-optimal.

Thus, the following result is obtained when including latency constraints. Note that the additional approximation factors (β, γ) for the resource violations arise due to the way the initial feasible primal LP solution was generated. If a feasible LP solution is provided externally, the respective factors are 1.

Theorem 6.41. When considering the VNEP setting $\langle \mathbf{VE} \,|\, \mathbf{NRL} \rangle$, (α, β, γ)-approximate solutions, with $\alpha = \beta = \gamma = 1+\varepsilon_{\mathrm{L}}$ to the cost variant of the fractional offline VNEP can be computed for any $\varepsilon_{\mathrm{L}} > 0$ in time $\mathcal{O}\left(\mathsf{poly}\left(\sum_{r\in\mathcal{R}} |V_r|^2 \cdot \left(|V_r| \cdot |V_S|^{2\cdot \mathrm{tw}(\mathcal{T}_r)+2} + \mathsf{time}_{\mathrm{LCSP}}(\varepsilon_{\mathrm{L}})\right)\right)\right)$ by first computing an initial primal LP solution using LP Formulation 6.11 and then solving the cost variant of LP Formulation 6.2. For the solution of both LPs the adapted DynVMP algorithm is used, employing the LCSP FPTAS, as separation oracle.

6.5 Summary and Novelty of Contributions

In this chapter several approaches to compute solutions to the fractional offline VNEP in XP-time have been proposed. Based on the computational complexity results presented in Section 5, we also know that XP-time algorithms are the best one can hope for (unless $\mathcal{P} = \mathcal{NP}$). Importantly, only the column generation approach discussed last allows for the inclusion of latencies. Overall, we believe that the presented DynVMP algorithm is of interest in its own right. As noted in the Observation 2.11, when request demands are small compared to the substrate capacities, solving the VMP optimally (or approximately) also yields optimal (or approximate) solutions for the VNEP.

Overall, the results presented in this Chapter are facilitated by the careful definition of the notion of valid mappings: if one would require valid mappings to be feasible, the respective fractional offline VNEP would remain $\mathcal{NP}$-hard. Accordingly, in the work of Jarray and Karmouch [JK15], which only generated feasible valid mappings in the separation, the respective separation procedure must be performed using exact approaches. Indeed, in future work, one might compare the separation approach of [JK15] with our presented XP-time algorithms to analyze trade-offs in runtime and solution quality.

"Probability is a mathematical discipline whose aims are akin to those, for example, of geometry or analytical mechanics. In each field we must carefully distinguish three aspects of the theory: (a) the formal logical content, (b) the intuitive background, and (c) the applications. The character, and the charm, of the whole structure cannot be appreciated without considering all three aspects in their proper relation."

– William Feller

7

XP-Approximations for the Offline VNEP and Evaluation of Derived Heuristics

As shown in Chapter 6, the fractional offline VNEP can be solved (respectively approximated) in XP-time. Concretely, considering the parametrizations of the extraction width and the treewidth, the fractional offline VNEP in the setting $\langle \mathbf{VE} \,|\, \mathbf{NR} \rangle$ can be solved in polynomial-time when the respective parameters are fixed. Additionally, considering the setting $\langle \mathbf{VE} \,|\, \mathbf{NRL} \rangle$, using the adapted column generation approach presented in Section 6.4, which employs treewidth as parametrization, the fractional offline VNEP can be approximated in XP-time.

In this chapter we apply randomized rounding on the computed solutions to the fractional offline VNEP to obtain tri-criteria (α, β, γ)-(XP)-approximations for the profit and the cost variant of the VNEP, i.e., the returned solutions will in general exceed node (factor β) and edge (factor γ) capacities. While the obtained algorithms are at first probabilistic in nature, deterministic approximations (with the same approximation guarantees) are presented as well by *derandomizing* the respective approximations.

In Sections 7.1 and 7.2 the approximation algorithms for the cost and the profit variant are presented together with the analysis of their probabilistic guarantees. In Section 7.3 derandomizations for all approximations are presented, hence obtaining deterministic XP-time approximations. As the approximation guarantees on the resource augmentations are highly dependent on various parameters, in Section 7.4 the approximation results are summarized in terms of easily computable parameters as the maximal demand-to-capacity ratio and the maximal size of any request graph and the size of the substrate network (cf. Tables 7.1 to 7.3).

Turning to the practical applications of the approximation framework and concentrating on the profit variant, several heuristics are derived in Section 7.5. Importantly, among the studied heuristics are also ones not exceeding capacity constraints. The performance of these heuristics is then evaluated in Section 7.6. Specifically, Section 7.6.2 presents results obtained from employing the LP formulation for cactus request graphs (cf. Section 6.2), while in Section 7.6.3 results based on the column generation LP (cf. Section 6.4) are discussed.

7.1 Approximations for the Offline Cost VNEP

We first present the cost approximation in the setting without latencies, i.e., $\langle \mathbf{VE} \,|\, \mathbf{NR} \rangle$. The pseudo-code is given as Algorithm 7.1. The algorithm computes an optimal fractional VNEP solution, prunes costly mappings, and then rounds solutions until an (α, β, γ)-approximate solution is obtained or the maximal number of rounding tries are exceeded. As we will show, the algorithm yields solutions of cost not more than 2 times the optimal cost. By probabilistic analysis, values for the respective approximation factors will be derived below for which the algorithm returns solutions *with high probability.*

Algorithm 7.1: Randomized Rounding Approximation for the Offline Cost VNEP $\langle$ **VE** | **NR** $\rangle$

1 $\{\mathcal{D}_r\}_{r\in\mathcal{R}} \leftarrow$ **compute** optimal solution to the cost fractional offline VNEP
2 **foreach** $r \in \mathcal{R}$ **do** // postprocess decomposition: prune costly mappings
3 **let** $\mathsf{WC}_r = \sum_{(f_r^k, m_r^k)\in\mathcal{D}_r} f_r^k \cdot c(m_r^k)$
4 **remove** tuples (f_r^k, m_r^k) from $\mathcal{D}_r$ with $c(m_r^k) > 2 \cdot \mathsf{WC}_r$
5 **normalize** weights of $\mathcal{D}_r$, such that $\sum_{(f_r^k, m_r^k)} f_r^k = 1$ holds again
6 **do** // perform rounding; as probabilities sum to 1, each request is embedded
7 construct solution by **choosing** mapping m_r^k **with probability** f_r^k **for all** $r \in \mathcal{R}$
8 **while** the solution is *not* $(\alpha = 2, \beta, \gamma)$-approximate and maximal rounding tries are not exceeded;

7.1.1 Deterministic Guarantee for the Cost

To obtain an approximation of the cost, the decision which of the mappings m_r^k to choose for request $r \in \mathcal{R}$ cannot be (purely) left to chance. Intuitively, this is due to the fact that the computed optimal fractional solution, may contain mappings of arbitrarily high cost but negligible weight. Hence, if the possibility exists to choose such a mapping in the rounding step, no reasonable bound on the rounded cost can be given in general. Accordingly, the costly fractional mappings are pruned for each request while not losing too much weight in the convex combination. Concretely, given a request $r \in \mathcal{R}$, we denote by $\mathsf{WC}_r = \sum_k f_r^k \cdot c_S(m_r^k)$ the *weighted (averaged) cost* of request $r \in \mathcal{R}$ and the algorithm removes all mappings m_r^k from the convex combinations for which $c_S(m_r^k) > 2 \cdot \mathsf{WC}_r$ hold. Intuitively, the weight of pruned mappings will be less than $1/2$, as otherwise the WC_r would need to be higher. The following lemma proves this observation:

Lemma 7.1. The sum of the weights f_r^k of the mappings m_r^k with cost smaller than two times WC_r is at least $1/2$ for each request $r \in \mathcal{R}$.

Proof. Let $\lambda_r = \sum_{(f_r^k, m_r^k)\in\mathcal{D}_r : c_S(m_r^k) \leq 2\cdot\mathsf{WC}_r} f_r^k$ denote the sum of the weights of the mappings of cost bounded by $2 \cdot \mathsf{WC}_r$. For the sake of contradiction, assume that $\lambda_r < 1/2$ holds for any request $r \in \mathcal{R}$. By the definition of WC_r and the assumption on λ_r, we obtain the following contradiction:

$$\mathsf{WC}_r = \sum_{(f_r^k, m_r^k)\in\mathcal{D}_r} f_r^k \cdot c_S(m_r^k) \tag{7.1}$$

$$\geq \sum_{(f_r^k, m_r^k)\in\mathcal{D}_r : c_S(m_r^k) > 2\cdot\mathsf{WC}_r} f_r^k \cdot c_S(m_r^k) \tag{7.2}$$

$$\geq \sum_{(f_r^k, m_r^k)\in\mathcal{D}_r : c_S(m_r^k) > 2\cdot\mathsf{WC}_r} f_r^k \cdot 2 \cdot \mathsf{WC}_r \tag{7.3}$$

$$\geq (1 - \lambda_r) \cdot 2 \cdot \mathsf{WC}_r > \mathsf{WC}_r \tag{7.4}$$

The Inequality 7.2 follows as only a subset of the requests is considered and Inequality 7.3 follows as all the considered decompositions have a cost of at least two times WC_r. The first inequality of Equation 7.4 then follows as $(1 - \lambda_r) > 1/2$ holds by assumption. As Equation 7.4 yields a contradiction, indeed $\lambda_r \geq 1/2$ must hold for $r \in \mathcal{R}$. ∎

We note that mappings are pruned from $\mathcal{D}_r$, the cost of the optimal solution equals the sum of the (fractional) costs of the convex combinations:

Lemma 7.2. Letting c_{frac} denote the cost of the optimal fractional VNEP solution computed in Line 1, we have:

$$\sum_{r\in\mathcal{R}} \mathsf{WC}_r = \sum_{r\in\mathcal{R}} \sum_k f_r^k \cdot c_S(m_r^k) = c_{\mathsf{frac}} \,. \tag{7.5}$$

Proof. Follows from the definition of the fractional offline VNEP. ■

Considering the above, one the following deterministic approximation guarantee is immediate.

Lemma 7.3. The cost of any solution returned by the randomized rounding scheme is upper bounded by two times the optimal cost.

Proof. Let c_{frac} denote the cost of the optimal fractional solution computed in Line 1. By allowing to select only decompositions m_r^k for which $c_S(m_r^k) \leq 2 \cdot \mathsf{WC}_r$ holds and denoting the selected mapping by $\hat{m}$ we have $c_S(\hat{m}_r) \leq 2 \cdot \mathsf{WC}_r$. Hence $\sum_{r \in \mathcal{R}} c_S(\hat{m}_r) \leq 2 \cdot \sum_{r \in \mathcal{R}} \mathsf{WC}_r$ holds. Together with Lemma 7.2 we obtain $\sum_{r \in \mathcal{R}} c_S(\hat{m}_r) \leq 2 \cdot c_{\mathsf{frac}}$. As c_{frac} is a lower bound for the minimum cost c_{opt} of the optimal solution the lemma holds. ■

7.1.2 Bounding Resource Allocations

In the following, we analyze the probability that a rounded solution exceeds substrate capacities by a certain factor. We model the allocations on resource $x \in G_S$ by request $r \in \mathcal{R}$ as random variable $A_{r,x} \in [0, A_{\mathsf{max}}(r,x)]$. By definition and under the simplifying assumption that no pair of mappings induces the same allocations on any resource, we have $\mathsf{Pr}[A_{r,x} = A(m_r^k, x)] = f_r^k$ and $\mathsf{Pr}[A_{r,x} = 0] = 1 - \sum_k f_r^k$. Furthermore, we denote by $A_x = \sum_{r \in \mathcal{R}} A_{r,x}$ the random variable capturing the overall allocations on resource $x \in G_S$. By definition $\mathsf{E}[A_x] = \sum_{r \in \mathcal{R}} \sum_k f_r^k \cdot A(m_r^k, x)$ holds. Utilizing the Chernoff bounds proven in Appendix A, the following lemma yields general bounds used throughout this section for exceeding the capacity of a single resource.

Lemma 7.4. Consider a resource $x \in G_S$ and let $\Delta_x > 0$ be chosen in such a way that $\Delta_x \cdot d_S(x)$ is an upper bound on the expected allocations on x, i.e., $\Delta_x \cdot d_S(x) \geq \mathsf{E}(A_x)$ holds. Letting $\hat{\mu}_x = \Delta_x \cdot d_S(x) / A_{\mathsf{max}}(x)$, the following holds for any $n \geq 3$ and any $c \geq 3$.

$$\mathsf{Pr}\left[A_x \geq 2 \cdot c \cdot \Delta_x \cdot \hat{\mu}_x^{-1} \cdot \frac{\ln n}{\ln \ln n} \cdot d_S(x)\right] \leq 1/n^c \text{, if} \quad \hat{\mu}_x < \frac{\ln n}{\ln \ln n} \tag{7.6}$$

$$\mathsf{Pr}\left[A_x \geq 2 \cdot c \cdot \Delta_x \cdot d_S(x)\right] \leq 1/n^c \text{, if} \quad \frac{\ln n}{\ln \ln n} \leq \hat{\mu}_x < 3 \cdot c \cdot \ln n \tag{7.7}$$

$$\mathsf{Pr}\left[A_x \geq (1+\varepsilon) \cdot \Delta_x \cdot d_S(x)\right] \leq 1/n^c \text{, if} \quad \hat{\mu}_x \geq 3 \cdot c \cdot \ln n / \varepsilon^2 \text{ , } \varepsilon \in (0,1)\,. \tag{7.8}$$

Proof. We first scale all random variables by the factor $1/A_{\mathsf{max}}(x)$ to apply the Chernoff bounds of Theorems A.1 and A.3. Accordingly, we introduce the scaled random variables $A'_{r,x} = A_{r,x}/A_{\mathsf{max}}(x) \in [0,1]$, $r \in \mathcal{R}$, and $A'_x = \sum_{r \in \mathcal{R}} A'_{r,x} = A_x / A_{\mathsf{max}}(x)$. We first prove Equation 7.6 by considering two cases, namely $\hat{\mu}_x < 1$ and $\hat{\mu}_x \geq 1$.

For $\hat{\mu}_x < 1$, we apply Theorem A.3 on the sum A'_x using $\hat{\mu} = 1$, $\rho = 2$, $\xi = 1$, and $c \geq 3$. Clearly, as $\hat{\mu}_x < 1 = \hat{\mu}$ holds, and as we assume $n \geq 3$ to hold, all requirements of Theorem A.3 are met. Accordingly, $\mathsf{Pr}[A'_x \geq 2 \cdot c \cdot \ln n / \ln \ln n] \leq 1/n^c$ is obtained. Now as, $A'_x = A_x / A_{\mathsf{max}}(x)$ holds, $\mathsf{Pr}[A_x \geq 2 \cdot c \cdot \ln n / \ln \ln n \cdot A_{\mathsf{max}}(x)] \leq 1/n^c$ follows. Observing that $A_{\mathsf{max}}(r,x) = \Delta_x \cdot d_S(x) / \hat{\mu}_x$ holds, Equation 7.6 follows for $\hat{\mu}_x < 1$.

For $1 \leq \hat{\mu}_x \leq \ln n / \ln \ln n$, we again apply Theorem A.3, while choosing the parameters slightly differently. In particular, we set $\hat{\mu} = \hat{\mu}_x$, $\rho = 2$, $\xi = \hat{\mu}_x^{-1}$, and $c \geq 3$. Again, it is easy to check that these settings satisfy all requirements of Theorem A.3. Hence, the following holds:

$$\mathsf{Pr}[A'_x \geq 2 \cdot c \cdot \hat{\mu}_x^{-1} \cdot \ln n / \ln \ln n \cdot \hat{\mu}_x] = \mathsf{Pr}[A'_x \geq 2 \cdot c \cdot \ln n / \ln \ln n] \leq 1/n^{c \cdot \hat{\mu}_x^{-1} \cdot \hat{\mu}_x} = 1/n^c\,.$$

By the same arguments as above, Equation 7.6 then follows also for $\hat{\mu}_x \leq \ln n / \ln \ln n$.

Equation 7.7 is a corollary of our above considerations for $1 \leq \hat{\mu}_x \leq \ln n / \ln \ln n$. Specifically, fixing $\xi = \ln \ln n / \ln n$ while otherwise applying Theorem A.3 using the same parameters, i.e., $\hat{\mu} = \hat{\mu}_x$, $\rho = 2$, and $c \geq 3$, the result readily follows, as ξ cancels out the factor $\ln n / \ln \ln n$.

Lastly, to obtain Equation 7.8, we employ Theorem A.1, specifically its second inequality. To this end, assume that $\hat{\mu}_x \geq 3 \cdot c \cdot \ln n / \varepsilon^2$ holds for some $\varepsilon \in (0,1)$. Setting $\delta = \varepsilon$ and using $\hat{\mu} = \hat{\mu}_x$, Theorem A.1 yields $\mathsf{Pr}[A'_x \geq (1+\varepsilon) \cdot \hat{\mu}_x] \leq e^{-\varepsilon^2 \cdot (3 \cdot c \cdot \ln n / \varepsilon^2)/3} = 1/n^c$. Again, as $A_x = A'_x \cdot A_{\mathsf{max}}(x)$ and $A_{\mathsf{max}}(x) = \Delta_x \cdot d_S(x) / \hat{\mu}_x$ holds, $\mathsf{Pr}[A_x \geq (1+\varepsilon) \cdot \Delta_x \cdot d_S(x)] \leq 1/n^c$ follows, concluding the proof. ■

Given the above lemma that a single resource $x \in G_S$ exceeds its capacity by a certain factor, the following lemma establishes general bounds that *any* resource exceeds its capacity.

Lemma 7.5. Let $\Delta_V, \Delta_E > 0$ be chosen minimally s.t. $\Delta_V \cdot d_S(u) \geq \mathsf{E}[A_u]$ and $\Delta_E \cdot d_S(u,v) \geq \mathsf{E}[A_{u,v}]$ hold for $u \in V_S$ and $(u,v) \in E_S$, respectively. Furthermore, let $\hat{\mu}_V = \min_{u \in V_S} \Delta_V \cdot d_S(u) / A_{\mathsf{max}}(u)$ and $\hat{\mu}_E = \min_{(u,v) \in E_S} \Delta_E \cdot d_S(u,v) / A_{\mathsf{max}}(u,v)$. Considering a substrate graph of n nodes, i.e., $n = |V_S|$, we introduce the following generic function to denote the maximal capacity augmentation:

$$\Lambda_{\mathsf{gen}}(c, n, \Delta, \hat{\mu}) = \begin{cases} 2 \cdot c \cdot \Delta \cdot \hat{\mu}^{-1} \cdot \frac{\ln n}{\ln \ln n} & \text{if } \hat{\mu} < \frac{\ln n}{\ln \ln n} \\ 2 \cdot c \cdot \Delta & \text{if } \frac{\ln n}{\ln \ln n} \leq \hat{\mu} < 3 \cdot c \cdot \ln n \\ (1 + \sqrt{3 \cdot c \cdot \ln n / \hat{\mu}}) \cdot \Delta & \text{if } \hat{\mu} \geq 3 \cdot c \cdot \ln n \,. \end{cases} \tag{7.9}$$

The function Λ_{gen} is monotonically decreasing for $\hat{\mu}$ when c, n, and Δ are fixed. The following holds for any $c, n \geq 3$:

$$\mathsf{Pr}\left[\begin{matrix} \exists u \in V_S : \quad A_u \geq \Lambda_{\mathsf{gen}}(n, c, \Delta_V, \hat{\mu}_V) \cdot d_S(u) \\ \vee \, \exists (u,v) \in E_S : A_{u,v} \geq \Lambda_{\mathsf{gen}}(n, c, \Delta_E, \hat{\mu}_E) \cdot d_S(u,v) \end{matrix} \right] \leq 1/n^{c-2} \,. \tag{7.10}$$

Proof. First convince yourself that the function Λ_{gen} is monotonically decreasing when increasing $\hat{\mu}$ with c, n, Δ being fixed. This is apparent on the intervals $(0, 3 \cdot c \cdot \ln n)$ and $(3 \cdot c \cdot \ln n, \infty)$. Furthermore, for the only discontinuous point $\hat{\mu}_0 = 3 \cdot c \cdot \ln n$, we have $\lim_{\hat{\mu} \nearrow \hat{\mu}_0} \Lambda_{\mathsf{gen}}(c, n, \Delta, \hat{\mu}) = 2 \cdot c \cdot \Delta \geq 2 \cdot \Delta = \Lambda_{\mathsf{gen}}(c, n, \Delta, \hat{\mu}_0)$ as $c \geq 3$, hence showing that Λ_{gen} is monotonically decreasing in $\hat{\mu}$.

Consider a node resource $u \in V_S$. Denoting by $\Delta_u = \Delta_x$ and $\hat{\mu}_u = \hat{\mu}_x$ the values introduced in Lemma 7.4 for $x = u$, we observe that $\Delta_u \leq \Delta_V$ holds. As the function Λ_{gen} captures the resource augmentation factors according to Lemma 7.4, by the same lemma we obtain that $\mathsf{Pr}[A_u \geq \Lambda_{\mathsf{gen}}(c, n, \Delta_u, \hat{\mu}_u) \cdot d_S(x)] \leq 1/n^c$ holds. As *decreasing* $\hat{\mu}_u$ only increases Λ_{gen}, also $\mathsf{Pr}[A_x \geq \Lambda_{\mathsf{gen}}(c, n, \Delta_u, \hat{\mu}_V) \cdot d_S(x)] \leq 1/n^c$ holds. Applying the same argument for edge resources, the following holds for any node resource $u \in V_S$ and any edge resource $(u,v) \in E_S$:

$$\begin{aligned} \mathsf{Pr}[A_u \geq \Lambda_{\mathsf{gen}}(n, c, \Delta_V, \hat{\mu}_V) \cdot d_S(u)] &\leq 1/n^c \\ \mathsf{Pr}[A_{u,v} \geq \Lambda_{\mathsf{gen}}(n, c, \Delta_E, \hat{\mu}_E) \cdot d_S(u,v)] &\leq 1/n^c \,. \end{aligned}$$

Observing that at most n^2 substrate resources exist, namely n nodes and at most $n \cdot (n-1)$ many edges, and applying a union bound on these at most n^2 many resources, each of which having a probability of less than $1/n^c$ to violate the resource bound, Equation B.3 follows. ■

Given the general Lemma 7.5, we obtain the following corollary for the cost approximation to exceed resources.

Corollary 7.6. Assume that $n = |V_S| \geq 3$ holds. Letting $\hat{\mu}_V^{\mathsf{cost}} = \min_{u \in V_S} 2 \cdot d_S(u) / A_{\mathsf{max}}(u)$ and $\hat{\mu}_E^{\mathsf{cost}} = \min_{(u,v) \in E_S} 2 \cdot d_S(u,v) / A_{\mathsf{max}}(u,v)$ and introducing

$$\Lambda_{\mathsf{cost}}(n, \Delta, \hat{\mu}) = \begin{cases} 6 \cdot \Delta \cdot \hat{\mu}^{-1} \cdot \frac{\ln n}{\ln \ln n} & \text{if } \hat{\mu} < \frac{\ln n}{\ln \ln n} \\ 6 \cdot \Delta & \text{if } \frac{\ln n}{\ln \ln n} \leq \hat{\mu} < 9 \cdot \ln n \\ (1 + \sqrt{9 \cdot \ln n / \hat{\mu}}) \cdot \Delta & \text{if } \hat{\mu} \geq 9 \cdot \ln n \,, \end{cases} \tag{7.11}$$

the following holds for the probability to exceed resources in the cost approximation according to Algorithm 7.1:

$$\mathsf{Pr}\left[\begin{array}{c}\exists u \in V_S: \quad A_u \geq \Lambda_{\mathsf{cost}}(n,2,\hat{\mu}_V^{\mathsf{cost}}) \cdot d_S(u) \\ \vee\, \exists (u,v) \in E_S: A_{u,v} \geq \Lambda_{\mathsf{cost}}(n,2,\hat{\mu}_E^{\mathsf{cost}}) \cdot d_S(u,v)\end{array}\right] \leq 1/n\,. \tag{7.12}$$

Proof. The result follows from Lemma 7.5. First observe that $\mathsf{E}[A_x] \leq 2 \cdot d_S(x)$ holds for any substrate resource $x \in G_S$. This follows from the observation that the initial fractional solution computed in Line 1 of Algorithm 7.1 is feasible. Specifically, $\sum_{r\in\mathcal{R}} \sum_{(f_r^k, m_r^k)\in\mathcal{D}_r} f_r^k \cdot A(m_r^k, x) \leq d_S(x)$ holds initially. After removing mappings of less than half of the weight and normalizing the remaining weights, $\sum_{r\in\mathcal{R}} \sum_{(f_r^k, m_r^k)\in\mathcal{D}_r} f_r^k \cdot A(m_r^k, x) \leq 2 \cdot d_S(x)$ holds. Hence, $\mathsf{E}[A_x] \leq 2 \cdot d_S(x)$ holds for any resource $x \in G_S$ and accordingly, the choices of $\Delta_V = \Delta_E = 2$ and $\hat{\mu}_V^{\mathsf{cost}}$ and $\hat{\mu}_E^{\mathsf{cost}}$ satisfy the requirements of Lemma 7.5. Employing $c = 3$, the result follows. ■

7.1.3 Cost Approximation without Latencies

Given Lemma 7.3 and Corollary 7.6, the following result is obtained for the cost variant of the offline VNEP $\langle\, \mathbf{VE} \,|\, \mathbf{NR} \,\rangle$.

Theorem 7.7. Assume $n = |V_S| \geq 3$. Consider $\hat{\mu}_V^{\mathsf{cost}}, \hat{\mu}_E^{\mathsf{cost}}$ and the function Λ_{cost} as introduced in Corollary 7.6. Let $\alpha = 2$, $\beta = \Lambda_{\mathsf{cost}}(n, 2, \hat{\mu}_V^{\mathsf{cost}})$, and $\gamma = \Lambda_{\mathsf{cost}}(n, 2, \hat{\mu}_E^{\mathsf{cost}})$ and let $N \in \mathbb{N}$ be the maximal number of rounding tries to execute. Algorithm 7.1 returns an (α, β, γ)-approximate solution for the *cost* variant of the offline VNEP $\langle\, \mathbf{VE} \,|\, \mathbf{NR} \,\rangle$ with the success probability being lower bounded by $1 - (1/3)^N$, i.e., with high probability. The runtime of Algorithm 7.1 is polynomial in the time to construct a fractional offline VNEP solution.

Proof. We consider the probability of the rounding step failing to produce an (α, β, γ)-approximate solution. By Corollary 7.6 the cumulative probability that the rounded solution exceeds the capacity of any node or edge resource by factors of β or γ is upper bounded by 1/3 for $n \geq 3$. As by Lemma 7.3 any constructed solution will have a cost of at most two times the minimal cost, the probability to not construct an (α, β, γ)-approximate solution within N rounding tries is lower bounded by $1 - (1/3)^N$.

With respect to the runtime of Algorithm 7.1, we note that each performed operation is polynomially bounded in the cumulative number of returned mappings, which in turn is bounded by the runtime of the algorithm to compute optimal fractional offline VNEP solutions. ■

7.1.4 Cost Approximation with Latencies

We now turn to the VNEP setting $\langle\, \mathbf{VE} \,|\, \mathbf{NRL} \,\rangle$, i.e., including latency constraints. As shown in Section 6.4, in this setting only $(\alpha = 1 + \varepsilon_{\mathrm{L}}, \beta = 1 + \varepsilon_{\mathrm{L}}, \gamma = 1 + \varepsilon_{\mathrm{L}})$-approximate fractional solutions to the offline VNEP can be computed (cf. Theorem 6.41). In the following, we argue that even using such an approximate solution the Algorithm 7.1 can still readily be applied.

Specifically, considering the deterministic guarantee for the profit, we observe that the rounded solution will always have a cost of at most $\alpha = 2 \cdot (1 + \varepsilon_{\mathrm{L}})$ times the cost of an optimal solution. Furthermore, after pruning costly mappings and normalizing the weights of the convex combinations, $\mathsf{E}(A_x) \leq 2 \cdot (1 + \varepsilon_{\mathrm{L}}) \cdot d_S(x)$ holds for any resource $x \in G_S$. Accordingly, by employing the same arguments as in the proof of Theorem 7.7, the following corollary is obtained:

Theorem 7.8. Assume that $n = |V_S| \geq 3$ holds and let $\varepsilon_{\mathrm{L}} > 0$, $\hat{\mu}_V^{\mathsf{cost,L}} = \min_{u\in V_S} 2 \cdot (1+\varepsilon_{\mathrm{L}}) \cdot d_S(u) / A_{\mathsf{max}}(u)$, and $\hat{\mu}_E^{\mathsf{cost,L}} = \min_{(u,v)\in E_S} 2 \cdot (1+\varepsilon_{\mathrm{L}}) \cdot d_S(u,v) / A_{\mathsf{max}}(u,v)$. Let $\alpha = 2 \cdot (1+\varepsilon_{\mathrm{L}})$, $\beta = \Lambda_{\mathsf{cost}}(n, 2 \cdot (1+\varepsilon_{\mathrm{L}}), \hat{\mu}_V^{\mathsf{cost,L}})$, and $\gamma = \Lambda_{\mathsf{cost}}(n, 2 \cdot (1 + \varepsilon_{\mathrm{L}}), \hat{\mu}_E^{\mathsf{cost,L}})$, with the function Λ_{cost} being defined as in Corollary 7.6.

In the setting $\langle \mathbf{VE} \,|\, \mathbf{NRL} \rangle$ Algorithm 7.1 yields an (α, β, γ)-approximate solution with high probability, when computing an $(\alpha = 1 + \varepsilon_{\mathrm{L}}, \beta = 1 + \varepsilon_{\mathrm{L}}, \gamma = 1 + \varepsilon_{\mathrm{L}})$-approximate fractional offline VNEP solution in Line 1. Specifically, the success probability is lower bounded by $1-(1/3)^N$ and the runtime of Algorithm 7.1 is polynomial in the time to compute the $(1 + \varepsilon_{\mathrm{L}}, 1 + \varepsilon_{\mathrm{L}}, 1 + \varepsilon_{\mathrm{L}})$-approximate fractional offline VNEP solution

7.2 Approximating the Profit Variant

In the following, the approximation for the profit variant of the offline VNEP is presented. Again we first consider the VNEP setting $\langle \mathbf{VE} \,|\, \mathbf{NR} \rangle$, i.e., without latencies.

The pseudo-code of our approximation for the profit is presented as Algorithm 7.2. In contrast to the approximation of the cost variant, the algorithm first performs a preprocessing in Lines 1-3, which removes all requests which cannot be fully (fractionally) embedded in the absence of other requests. As they cannot be fully embedded, these requests can never be part of any feasible solution and can hence be removed without further consideration. In Lines 3-7 the randomized rounding scheme is applied: a solution to the fractional offline VNEP is computed and then rounded. Notably and in contrast to the approximation of the cost variant, as the sum of the weights $\sum_k f_r^k$ for a request $r \in \mathcal{R}$ may not sum to 1, a request r is rejected with probability $1 - \sum_k f_r^k$. The rounding procedure is again iterated as long as the constructed solution is not of sufficient quality or until the maximal number of rounding tries is exceeded. In the following we discuss the parameters α, β, and γ for which solutions can be found *with high probability*.

7.2.1 Bounding the Profit

Employing the *discrete* random variable $B_r \in \{0, b_r\}$ to model the profit achieved by (potentially) embedding request $r \in \mathcal{R}$, we have $\mathsf{Pr}[B_r = b_r] = \sum_{(f_r^k, m_r^k) \in \mathcal{D}_r} f_r^k$ and $\mathsf{Pr}[B_r = 0] = 1 - \sum_{(f_r^k, m_r^k) \in \mathcal{D}_r} f_r^k$. Hence, the overall profit achieved is $B = \sum_{r \in \mathcal{R}} B_r$ with $\mathsf{E}[B] = \sum_{r \in \mathcal{R}} b_r \cdot \sum_{(f_r^k, m_r^k) \in \mathcal{D}_r} f_r^k$. Denoting by $b_{\mathsf{frac}} = \sum_{r \in \mathcal{R}} b_r \cdot \sum_{(f_r^k, m_r^k) \in \mathcal{D}_r} f_r^k$ the profit achieved by the optimal fractional solution, we have $b_{\mathsf{frac}} = \mathbb{E}[B]$.

By removing any request, which cannot be fully fractionally embedded (in the absence of other requests) in the preprocessing step, we know that $b_{\mathsf{frac}} \geq \max_{r \in \mathcal{R}} b_r = b_{\mathsf{max}}$ holds: the fractional solution pertaining to the whole set of requests must achieve at least the profit that was achieved by embedding each request individually. By applying a Chernoff bound, the following is obtained.

Lemma 7.9. The probability of achieving less than a factor of $\alpha = 1 - 2/3 \cdot \sqrt{b_{\mathsf{max}}/b_{\mathsf{frac}}} \geq 1/3$ of the profit of the optimal solution is upper bounded by $exp(-2/9) \approx 0.8007 \leq 65/81$.

Algorithm 7.2: Randomized Rounding Approximation for the Offline Profit VNEP $\langle \mathbf{VE} \,|\, \mathbf{NR} \rangle$

1 **foreach** $r \in \mathcal{R}$ **do** // preprocess requests
2 **compute** optimal solution to the fractional profit VNEP for the request set $\{r\}$
3 **remove** request r from the set $\mathcal{R}$ if the fractional solution does not attain a profit of b_r
4 $\{\mathcal{D}_r\}_{r \in \mathcal{R}} \leftarrow$ **compute** optimal solution to the fractional offline VNEP for request set $\mathcal{R}$
5 **do** // perform randomized rounding
6 construct solution by **choosing** mapping m_r^k **with probability** f_r^k **for all** $r \in \mathcal{R}$
7 **while** the solution is *not* (α, β, γ)-approximate and maximal rounding tries are not exceeded;

Proof. Let $b_{\mathsf{max}} = \max_{r \in \mathcal{R}} b_r$ denote the maximum benefit of the requests. We consider the random variables $B'_r = B_r / b_{\mathsf{max}}$, such that $B'_r \in [0,1]$ holds. Let $B' = \sum_{r \in \mathcal{R}} B'_r$ denote the total profit achieved after scaling down the profits. Clearly, $\mathsf{E}[B'] = \mathsf{E}[B]/b_{\mathsf{max}} = b_{\mathsf{frac}}/b_{\mathsf{max}}$ holds. Applying the Chernoff bound of Theorem A.2 with $\delta = 2/3 \cdot \sqrt{b_{\mathsf{max}}/b_{\mathsf{frac}}}$ and $\tilde{\mu} = \mathsf{E}[B'] = b_{\mathsf{frac}}/b_{\mathsf{max}}$, the following is obtained:

$$\begin{aligned}
\mathsf{Pr}[B' \leq (1-\delta)\cdot\tilde{\mu}] =& \mathsf{Pr}[B' \leq (1 - 2/3 \cdot \sqrt{b_{\mathsf{max}}/b_{\mathsf{frac}}}) \cdot b_{\mathsf{frac}}/b_{\mathsf{max}}] && \text{(plugging in values)}\\
\leq& e^{-\delta^2 \cdot \tilde{\mu}/2} && \text{(by Theorem A.2)}\\
=& e^{-4/9 \cdot b_{\mathsf{max}}/b_{\mathsf{frac}} \cdot (b_{\mathsf{frac}}/b_{\mathsf{max}})/2} && \text{(plugging in values)}\\
=& e^{-2/9}
\end{aligned}$$

Denoting the optimal profit of the Integer Program by b_{opt} and observing that $b_{\mathsf{opt}} \leq b_{\mathsf{frac}}$ holds as the fractional solution will always obtain a profit larger than the optimal (integral) solution, we have $b_{\mathsf{opt}} \leq b_{\mathsf{frac}} = \mathbb{E}(B)$. Hence, letting $\alpha = 1 - 2/3 \cdot \sqrt{b_{\mathsf{max}}/b_{\mathsf{frac}}}$, we obtain

$$\mathsf{Pr}[B \leq \alpha \cdot b_{\mathsf{opt}}] \leq \mathsf{Pr}[B \leq \alpha \cdot b_{\mathsf{frac}}] = \mathsf{Pr}[B' \leq \alpha \cdot b_{\mathsf{frac}}/b_{\mathsf{max}}] = \mathsf{Pr}[B' \leq \alpha \cdot \tilde{\mu}] \leq exp(-2/9)\,.$$

Since $b_{\mathsf{frac}} \geq b_{\mathsf{max}}$ holds, the approximation factor α is lower bounded by $1/3$.

Lastly, by using the Taylor series expansion of the function $f(x) = e^x$, we obtain that $e^x \leq 1 + x + x^2/2$ holds for $x < 0$. Accordingly, $\exp(-2/9) \leq 65/81$ holds. ■

7.2.2 Probabilistic Guarantee for Resource Augmentations

In the following, the probability that a rounded solution exceeds substrate capacities by a certain factor is analyzed analogously to Section 7.1.2. In particular, the same random variables $A_{r,x} \in [0, A_{\mathsf{max}}(r,x)]$ and $A_x = \sum_{r \in \mathcal{R}} A_{r,x}$ are employed to denote the cumulative allocations induced by request $r \in \mathcal{R}$ on resource $x \in G_S$ and the overall allocations induced on resource x, respectively. Similarly, to the analysis of the cost variant, we note that $\mathsf{E}[A_x] = \sum_{r \in \mathcal{R}} \sum_k f_r^k \cdot A(m_r^k, x)$ holds by the feasibility of the computed fractional solution. Furthermore, as the rounding procedure is the same as in Algorithm 7.1, Lemma B.3 is again applicable and we obtain the following corollary:

Corollary 7.10. Assume that $n = |V_S| \geq 3$ holds. Using $\hat{\mu}_V^{\mathsf{prof}} = \min_{u \in V_S} d_S(u)/A_{\mathsf{max}}(u)$, $\hat{\mu}_E^{\mathsf{prof}} = \min_{(u,v) \in E_S} d_S(u,v)/A_{\mathsf{max}}(u,v)$ and introducing the function

$$\Lambda_{\mathsf{prof}}(n, \hat{\mu}) = \begin{cases} 8 \cdot \hat{\mu}^{-1} \cdot \frac{\ln n}{\ln \ln n} & \text{if } \hat{\mu} < \frac{\ln n}{\ln \ln n} \\ 8 & \text{if } \frac{\ln n}{\ln \ln n} \leq \hat{\mu} < 12 \cdot \ln n \\ (1 + \sqrt{12 \cdot \ln n / \hat{\mu}}) & \text{if } \hat{\mu} \geq 12 \cdot \ln n\,, \end{cases} \tag{7.13}$$

the following holds for the probability to exceed resources in the profit approximation according to Algorithm 7.2:

$$\mathsf{Pr}\left[\begin{array}{c} \exists u \in V_S : \quad A_u \geq \Lambda_{\mathsf{prof}}(n, \hat{\mu}_V^{\mathsf{prof}}) \cdot d_S(u) \\ \vee\, \exists (u,v) \in E_S : A_{u,v} \geq \Lambda_{\mathsf{prof}}(n, \hat{\mu}_E^{\mathsf{prof}}) \cdot d_S(u,v) \end{array}\right] \leq 1/n^2\,. \tag{7.14}$$

Proof. The result follows from Lemma 7.5. First observe that $\mathsf{E}[A_x] \leq d_S(x)$ holds for any substrate resource $x \in G_S$. This follows from the observation that for the initial fractional solution computed in Line 4 of Algorithm 7.2 is feasible. Hence, the values $\Delta_V = \Delta_E = 1$ and $\hat{\mu}_V^{\mathsf{prof}}$ and $\hat{\mu}_E^{\mathsf{prof}}$ satisfy the requirements of Lemma 7.5. Setting $c = 4$ the result follows. ■

7.2.3 Profit Approximation without Latencies

Given the probabilistic bounds for achieving an α-fraction of the optimal profit (cf. Lemma 7.9) and for violating resource capacities by factors more than β and γ (cf. Corollary 7.10), the following approximation result is obtained.

Theorem 7.11. Assume that $n = |V_S| \geq 3$. Let $\alpha = 1 - 2/3 \cdot \sqrt{b_{\mathsf{frac}}/b_{\mathsf{max}}} \geq 1/3$, $\beta = \Lambda_{\mathsf{prof}}(n, \hat{\mu}_V^{\mathsf{prof}})$, and $\gamma = \Lambda_{\mathsf{prof}}(n, \hat{\mu}_E^{\mathsf{prof}})$ with $\hat{\mu}_V^{\mathsf{prof}}$, $\hat{\mu}_E^{\mathsf{prof}}$, and the function Λ_{prof} defined in Corollary 7.10. Let $N \in \mathbb{N}$ be the maximal number of rounding tries to execute. Algorithm 7.2 returns an (α, β, γ)-approximate solution for the *profit* variant of the offline VNEP $\langle\, \mathbf{VE} \,|\, \mathbf{NR} \,\rangle$ with the success probability being lower bounded by $1 - (74/81)^N$, i.e., with high probability. The runtime of Algorithm 7.1 is polynomial in the time to construct a fractional offline VNEP solution.

Proof. We consider the probability of the rounding step failing to produce an (α, β, γ)-approximate solution. By Corollary 7.10 the cumulative probability that the rounded solution exceeds the capacity of any node or edge resource by factors of β or γ is upper bounded by $1/n^2 \leq 1/9$ for $n \geq 3$. As by Lemma 7.9 a rounded solution achieves at most an α-fraction of the optimal profit with probability 65/81, the cumulative probability that no (α, β, γ)-approximate solution is found is upper bounded by 74/81. Hence, the probability to construct such an approximate solution within N rounding tries is lower bounded by $1 - (74/81)^N$.

With respect to the runtime of Algorithm 7.2, we note that each performed operation is polynomially bounded in the cumulative number of returned mappings, which in turn is bounded by the runtime of the algorithm to compute the optimal fractional offline VNEP solutions. ■

7.2.4 Profit Approximation with Latencies

We now turn towards the setting when additionally latencies are considered, i.e., the VNEP variant $\langle\, \mathbf{VE} \,|\, \mathbf{NRL} \,\rangle$. As the respective fractional offline VNEP can only be approximately solved, Algorithm 7.2 must be adapted accordingly. The pseudo-code of the approximation with latencies is given as Algorithm 7.3.

In contrast to Algorithm 7.2, a request is removed in the preprocessing step only if less than $1/(1+\varepsilon_{\mathrm{L}})$ of its profit was achieved, as only in this case it is guaranteed, that the respective request cannot be fully embedded in an integral solution. Furthermore, as the solution computed in Line 4 is only an approximate one, the analysis needs to be slightly adapted. To this end, we employ the same set of random variables $B_r \in \{0, b_r\}$, $r \in \mathcal{R}$, and $B = \sum_{r \in \mathcal{R}} B_r$, where $B_r = b_r$ denotes the event that request r is embedded and $B_r = 0$ holds, if the request was rejected. Furthermore, we again denote by $b_{\mathsf{max}} = \max_{r \in \mathcal{R}} b_r$ the maximum benefit of the requests and by b_{frac} the now $1/(1+\varepsilon_{\mathrm{L}})$-optimal objective of the LP solution. We slightly adapt Lemma 7.9 as follows:

Algorithm 7.3: Randomized Rounding Approximation for the Offline Profit VNEP $\langle\, \mathbf{VE} \,|\, \mathbf{NRL} \,\rangle$

Input : Requests $\mathcal{R}$, Substrate G_S, LCSP approximation factor ε_{L}

1 **foreach** $r \in \mathcal{R}$ **do** `// preprocess requests`

2 $\quad \tilde{\mathcal{D}}_r \leftarrow$ **compute** $1/(1+\varepsilon_{\mathrm{L}})$-approximate solution to fractional profit VNEP for request set $\{r\}$

3 $\quad$ **remove** request r from the set $\mathcal{R}$ if $\sum_{(\tilde{f}_r^k, \tilde{m}_r^k) \in \tilde{\mathcal{D}}_r} f_r^k < 1/(1+\varepsilon_{\mathrm{L}})$

4 $\{\mathcal{D}_r\}_{r \in \mathcal{R}} \leftarrow$ **compute** $1/(1+\varepsilon_{\mathrm{L}})$-approximate solution to fractional offline VNEP for request set $\mathcal{R}$

5 **do** `// perform randomized rounding`

6 $\quad$ construct solution by **choosing** mapping m_r^k **with probability** f_r^k **for all** $r \in \mathcal{R}$

7 **while** the solution is *not* (α, β, γ)-approximate and maximal rounding tries are not exceeded;

Lemma 7.12. The probability of achieving less than a factor of $\alpha = (1+\varepsilon_\mathrm{L})^{-1} \cdot (1 - 2/3 \cdot \sqrt{b_\mathsf{max}/b_\mathsf{frac}}) \geq 1/3 \cdot (1+\varepsilon_\mathrm{L})^{-3/2}$ of the profit of the optimal solution is upper bounded by $exp(-2/9) \approx 0.8007 \leq 65/81$.

Proof. Using the same argument as in the proof of Lemma 7.9, $\mathsf{Pr}[B' \leq (1-\delta)\cdot\tilde{\mu}] \leq e^{-2/9}$ holds for $\tilde{\mu} = b_\mathsf{frac}/b_\mathsf{max}$ and $\delta = 2/3 \cdot \sqrt{b_\mathsf{max}/b_\mathsf{frac}}$. However, two things need to be noted:

1. As the solution computed in Line 4 is only $1/(1+\varepsilon_\mathrm{L})$-approximate and denoting by b_opt the optimal solution, we have $b_\mathsf{opt} \leq (1+\varepsilon_\mathrm{L}) \cdot b_\mathsf{frac}$. Hence, $\mathsf{Pr}[B \leq (1-\delta)\cdot(1+\varepsilon_\mathrm{L})^{-1} \cdot b_\mathsf{opt}] \leq e^{-2/9}$ follows from $\mathsf{Pr}[B' \leq (1-\delta)\cdot\tilde{\mu}] \leq e^{-2/9}$ for the actual achieved profit B.

   Hence, for the approximation guarantee $\alpha = (1-\delta)\cdot(1+\varepsilon_\mathrm{L})^{-1}$ holds.

2. Secondly, by the preprocessing step only $b_\mathsf{frac} \geq b_\mathsf{max}/(1+\varepsilon_\mathrm{L})$ holds true. Hence, for the minimum value of b_frac, i.e., $b_\mathsf{frac} = b_\mathsf{max}/(1+\varepsilon_\mathrm{L})$, the following holds for the approximation factor α:

$$\alpha \geq (1-\delta)\cdot(1+\varepsilon_\mathrm{L})^{-1} = (1 - 2/3 \cdot \sqrt{1/(1+\varepsilon_\mathrm{L})}) \cdot (1+\varepsilon_\mathrm{L})^{-1} = 1/3 \cdot (1+\varepsilon_\mathrm{L})^{-3/2}\,.$$

Using the above and the same Taylor series expansion to bound $e^{-2/9}$, the result follows. ■

Given the above lemma, it is easy to see that the approximation result of Theorem 7.11 carries over to the case with latencies under minor changes of the approximation factors:

Theorem 7.13. Assume $n = |V_S| \geq 3$ and let $\alpha = (1+\varepsilon_\mathrm{L})^{-1}\left(1 - 2/3 \cdot \sqrt{b_\mathsf{frac}/b_\mathsf{max}}\right)$, such that $\alpha \geq 1/3 \cdot (1+\varepsilon_\mathrm{L})^{-3/2}$ holds, $\beta = \Lambda_\mathsf{prof}(n, \hat{\mu}_V^\mathsf{prof})$, and $\gamma = \Lambda_\mathsf{prof}(n, \hat{\mu}_E^\mathsf{prof})$ with $\hat{\mu}_V^\mathsf{prof}$, $\hat{\mu}_E^\mathsf{prof}$, and the function Λ_prof being as defined in Corollary 7.10. Let $N \in \mathbb{N}$ be the maximal number of rounding tries to execute. Algorithm 7.3 returns an (α,β,γ)-approximate solution for the *profit* variant of the offline VNEP $\langle\, \mathbf{VE} \,|\, \mathbf{NRL} \,\rangle$ with the success probability being lower bounded by $1-(74/81)^N$, i.e., with high probability. The runtime of Algorithm 7.1 is polynomial in the time to construct a fractional offline VNEP solution.

Proof. We employ the same union bound as for the proof of Theorem 7.11. By Lemma 7.12 at least an α fraction of the optimal profit is obtained with probability of at most $e^{-2/9} \leq 65/81$. Based on the feasibility of the solution, for any resource $x \in G_S$ still $\mathsf{E}[A_x] \leq d_S(x)$ holds. Hence, Corollary 7.10 can be readily applied to obtain that the probability to violate any resource by a factor of more than β (nodes) or γ (edges) is upper bounded by 1/9. Using the same union bound argument, the probability to successfully return an (α,β,γ)-approximate solution is lower bounded by $1-(74/81)^N$. Noting, again, that the runtime of the approximation is dominated by the time to construct the approximate fractional offline VNEP solutions, the result follows. ■

7.2.5 Trading Off Capacity Violations with the Obtained Profit

A well-known technique to reduce capacity violations is to a priori scale down the substrate's capacities by a certain factor, thereby reducing the a posteriori violations [RT87; CLN04; ERS16a]. In the following this technique is applied to the approximations of the profit VNEP. Notably, as scaling down capacities will proportionally reduce the achieved profit and thereby the embedding values, this technique is not applicable for the cost variant, as in this case all requests need to be fully embedded.

In the following, we start by analyzing under which conditions capacities can be scaled without disproportionately reducing the profit.

Lemma 7.14. Consider an offline profit VNEP instance. Denoting by $\delta_\mathrm{max} = \max_{x \in G_S} d_\mathsf{max}(x)/d_S(x) \leq 1$ the maximal demand to capacity ratio (cf. Definition 2.4) and let $\varepsilon_\mathsf{scal} \in [\delta_\mathrm{max}, 1]$. If there exists a fractional solution of profit b_frac under the original capacities, then there exists a fractional solution of profit $\varepsilon_\mathsf{scal} \cdot b_\mathsf{frac}$ under the scaled capacities $d'_S(x) = \varepsilon_\mathsf{scal} \cdot d_S(x)$.

Proof. Denote by $\{\mathcal{D}_r = \{(f_r^k, m_r^k)\}_k\}_{r\in\mathcal{R}}$ the solution under the original capacities d_S of profit b_{frac}. The solution $\{\tilde{\mathcal{D}}_r = \{(\tilde{f}_r^k = \varepsilon_{\mathsf{scal}} \cdot f_r^k, m_r^k)\}\}_{r\in\mathcal{R}}$ clearly achieves a profit of $\varepsilon_{\mathsf{scal}} \cdot b_{\mathsf{frac}}$ and is feasible with respect to the adapted capacities d'_S. Also note that with respect to the adapted capacities all scaled mappings are still valid, i.e., the maximal demand on any substrate resource never exceeds the provided capacity. This holds as $\varepsilon_{\mathsf{scal}} \in [\delta_{\max}, 1]$ was enforced. Hence, even for the scaled capacities, the ratio of maximal demand to maximal capacity still lies below 1. ■

Using the above observation, we can now slightly adapt the approximation algorithms for the profit variant of the VNEP. Specifically, having computed a fractional solution, the weights are scaled together with the capacities. Using the exact same analysis as before, but now with respect to the scaled capacities, the following theorem is obtained.

Theorem 7.15. Assume that $n = |V_S| \geq 3$ and let $\varepsilon_{\mathsf{scal}} \in [\delta_{\max}, 1]$. Furthermore, let $\hat{\mu}_V^{\mathsf{scal}} = \min_{u\in V_S} \varepsilon_{\mathsf{scal}} \cdot d_S(u)/A_{\mathsf{max}}(u)$ and $\hat{\mu}_E^{\mathsf{scal}} = \min_{(u,v)\in E_S} \varepsilon_{\mathsf{scal}} \cdot d_S(u,v)/A_{\mathsf{max}}(u,v)$ be upper bounds on the expected allocations *after* scaling and

$$\Lambda_{\mathsf{prof}}^{\mathsf{scal}}(n, \varepsilon, \hat{\mu}) = \begin{cases} 8 \cdot (\hat{\mu})^{-1} \cdot \varepsilon \cdot \frac{\ln n}{\ln \ln n} & \text{if } \hat{\mu} < \frac{\ln n}{\ln \ln n} \\ 8 \cdot \varepsilon & \text{if } \frac{\ln n}{\ln \ln n} \leq \hat{\mu} < 12 \cdot \ln n \\ (1 + \sqrt{12 \cdot \ln n / \hat{\mu}}) \cdot \varepsilon & \text{if } \hat{\mu} \geq 12 \cdot \ln n\,. \end{cases} \tag{7.15}$$

Considering the profit variant of the offline VNEP, Algorithms 7.2 and 7.3 can be adapted as follows to trade off capacity augmentations with the achieved profit. After having computed the optimal or approximate fractional offline VNEP solution in Line 4, the weights are scaled for each request by the factor $\varepsilon_{\mathsf{scal}}$ and afterwards randomized rounding is applied on this scaled solution. The adapted approximation yields $(\alpha = \varepsilon_{\mathsf{scal}} \cdot \alpha', \beta = \Lambda_{\mathsf{prof}}^{\mathsf{scal}}(n, \varepsilon_{\mathsf{scal}}, \hat{\mu}_V^{\mathsf{scal}}), \gamma = \Lambda_{\mathsf{prof}}^{\mathsf{scal}}(n, \varepsilon_{\mathsf{scal}}, \hat{\mu}_E^{\mathsf{scal}}))$-approximate solutions with high probability. Specifically, the success probability is lower bounded by $1 - (74/81)^N$, where N denotes the number of rounding tries, and the runtime of the adapted approximations is still polynomial in the time to construct the fractional offline VNEP solution.

Proof. Denoting by B and B^{scal} the profit of the unscaled and scaled solution, respectively, and noting that $\mathsf{E}[B^{\mathsf{scal}}] = \varepsilon_{\mathsf{scal}} \cdot \mathsf{E}[B]$ holds, we have

$$\mathsf{Pr}[B \leq c \cdot \mathsf{E}[B]] = \mathsf{Pr}[B \cdot \varepsilon_{\mathsf{scal}} \leq c \cdot \mathsf{E}[B] \cdot \varepsilon_{\mathsf{scal}}] = \mathsf{Pr}[B^{\mathsf{scal}} \leq c \cdot \mathsf{E}[B^{\mathsf{scal}}]]\,.$$

Hence, scaling the weights results in an accordingly scaled approximation factor $\alpha = \varepsilon_{\mathsf{scal}} \cdot \alpha'$.

For the resource augmentations factors β and γ, consider the following. By Lemma 7.14, the scaled solution is a feasible fractional solution with respect to the scaled capacities d'_S. Hence, Corollary 7.10 can be readily applied with respect to the *scaled* capacities d'_S. By considering the expectations $\hat{\mu}_V^{\mathsf{scal}}$, $\hat{\mu}_E^{\mathsf{scal}}$ with respect to the scaled capacities, probabilistic bounds for $A_u \geq \Lambda_{\mathsf{prof}}(n, \hat{\mu}_V^{\mathsf{scal}}) \cdot d'_S(u)$ and $A_{u,v} \geq \Lambda_{\mathsf{prof}}(n, \hat{\mu}_E^{\mathsf{scal}}) \cdot d'_S(u,v)$ are obtained for node and edge resources $u \in V_S$ and $(u,v) \in E_S$, respectively. As $d'_S(x) = \varepsilon_{\mathsf{scal}} \cdot d_S(x)$ holds, the probabilistic bounds translate to probabilistic bounds for $A_u \geq \Lambda_{\mathsf{prof}}(n, \hat{\mu}_V^{\mathsf{scal}}) \cdot \varepsilon_{\mathsf{scal}} \cdot d_S(u)$ and $A_{u,v} \geq \Lambda_{\mathsf{prof}}(n, \hat{\mu}_E^{\mathsf{scal}}) \cdot \varepsilon_{\mathsf{scal}} \cdot d_S(u,v)$ with respect to the original capacities d_S. As $\Lambda_{\mathsf{prof}}^{\mathsf{scal}}(n, \varepsilon_{\mathsf{scal}}, \hat{\mu}^{\mathsf{scal}}) = \varepsilon_{\mathsf{scal}} \cdot \Lambda_{\mathsf{prof}}(n, \hat{\mu}^{\mathsf{scal}})$ holds, the result follows. ■

7.3 Derandomization: Deterministic Approximations

All above derived results (cf. Theorems 7.7, 7.8, 7.11, 7.13, and 7.15) rely on randomized rounding and are hence probabilistic in nature. However, it is well-known that algorithms relying on randomized rounding can often be *derandomized* using a technique called *conditional probabilities* or *conditional expectation*

by employing *pessimistic estimators* [Rag86; Rag88; MR95; WS11; AS16; MU17]. In the following we shortly revisit this concept and then show how to derandomize any of our randomized rounding algorithms presented above.

7.3.1 Overview on the Method of Conditional Expectation

In all of the randomized rounding algorithms, first for each request $r \in \mathcal{R}$ a convex combination of mappings $\mathcal{D}_r = \{(f_r^k, m_r^k)\}_k$ is constructed and then one of the mappings m_r^k is chosen (or potentially none in the profit case). The construction of solutions can accordingly be probabilistically captured in the following way. Let $n = |\mathcal{R}|$ denote the number of requests, $\mathcal{R} = \{r_1, \ldots, r_n\}$, and let $\hat{k}_i = |\mathcal{D}_{r_i}|$ denote the number of found mappings for the i-th request r_i. Furthermore, denote by $\hat{K}_i = \{0, 1, \ldots, \hat{k}_r\}$ the index set (including 0) for the i-th request r_i. Denote by $Z_i \in \hat{K}_i$ the random variable pertaining to the choice of mapping for request r_i, such that $Z_i = k$ if and only if the k-th mapping was chosen to embed it and $Z_i = 0$ holds if none of the mappings was chosen. Clearly, $\mathsf{Pr}[Z_i = k] = f_r^k$ and $\mathsf{Pr}[Z_i = 0] = 1 - \sum_k f_r^k$ holds.

Now, let $\mathcal{F} : \hat{K}_1 \times \ldots \times \hat{K}_n \to \mathbb{R}_{\geq 0}$ denote a real-valued function over the choices of selected mappings. Proving the performance guarantees of the randomized rounding approximations, probabilistic bounds for events such as 'no resource exceeds its capacity by factors more than β or γ' were proven. In fact, considering the resource augmentation example, consider the following function:

$$\mathcal{F}(z_1, \ldots, z_n) = \begin{cases} 0 & \text{, if no resource exceeds its capacity by factors } \beta \text{ or } \gamma \\ & \text{according to the embedding indicated by } z_1, \ldots, z_n \\ 1 & \text{, otherwise} \end{cases} .$$

This function is the *failure* indicator function for constructing a suitable (β, γ)-approximate solution. The probabilistic analyses of the different approximations have shown that

$$\mathsf{E}[\mathcal{F}(Z_1, \ldots, Z_n)] = \mathsf{Pr}[\text{a resource exceeds its capacity by factors more than } \beta \text{ or } \gamma] < 1$$

holds, hence proving the existence of a (β, γ)-approximate solution. Using the above notation such a solution is now represented as an $|\mathcal{R}|$-tuple $(z_1, z_2, \ldots, z_n) \in \hat{K}_1 \times \ldots \times \hat{K}_n$. The task of the derandomization process is to deterministically construct such a solution. Since, trying out all $\prod_{i \in [n]} (\hat{k}_i + 1)$ potential combinations of mapping choices is computationally prohibitive, a more involved technique needs to be applied. In the following, the method of conditional expectations is revisited to deterministically construct such solutions in polynomial-time.

The method of conditional expectations requires to efficiently evaluate the following function:

$$\mathsf{E}_{Z_{i+1}, \ldots, Z_n}[\mathcal{F}(z_1, \ldots, z_i, Z_{i+1}, \ldots, Z_n)] = \mathsf{Pr}[\mathcal{F}(Z_1, \ldots, Z_n) \mid Z_1 = z_1, \ldots, Z_i = z_i] .$$

If the above function can always, i.e., for any prefix length i and any choice of $(z_1, \ldots, z_i)$, be efficiently computed, the method of conditional expectations constructs a solution as follows. Assuming that $\mathsf{E}_{Z_1, \ldots, Z_n}[\mathcal{F}(Z_1, \ldots, Z_n)] = \epsilon < 1$ initially holds, there must exist a choice z_1 for Z_1 such that $\mathsf{E}_{Z_2, \ldots, Z_n}[\mathcal{F}(z_1, Z_2, \ldots, Z_n)] \leq \epsilon < 1$ holds. Given that the conditional expectation $\mathsf{E}_{Z_2, \ldots, Z_n}[\mathcal{F}(z_1, \ldots, Z_n)]$ can be computed by our assumption for any value $z_1 \in \hat{K}_1$, one may pick any mapping z_1 satisfying $\mathsf{E}_{Z_2, \ldots, Z_n}[\mathcal{F}(z_1, \ldots, Z_n)] \leq \epsilon < 1$ and can then iteratively select appropriate values $z_2, z_3, \ldots, z_n$, such that the conditional expectation only decreases:

$$\begin{aligned} 1 > \epsilon = & \mathsf{E}_{Z_1, \ldots, Z_n}[\mathcal{F}(Z_1, \ldots, Z_n)] && \text{(by assumption)} \\ \geq & \mathsf{E}_{Z_2, \ldots, Z_n}[\mathcal{F}(z_1, Z_2, \ldots, Z_n)] && \text{(choice of appropriate } z_1\text{)} \\ \geq & \mathsf{E}_{Z_3, \ldots, Z_n}[\mathcal{F}(z_1, z_2, Z_3, \ldots, Z_n)] && \text{(choice of appropriate } z_2\text{)} \\ & \vdots \\ \geq & \mathsf{E}[\mathcal{F}(z_1, \ldots, z_n)] = \mathcal{F}(z_1, \ldots, z_n) && \text{(choice of appropriate } z_n\text{)} \end{aligned}$$

As the conditional expectation only decreases and at the end no randomness remains, since for each random variable specific values have been selected, $\mathcal{F}(z_1, \ldots, z_n) \leq \epsilon < 1$ must hold. Indeed, as the function is binary, actually $\mathcal{F}(z_1, \ldots, z_n) = 0$ must hold, proving that the constructed solution indeed satisfies the respective resource augmentation factors. To apply the above method, one has to *efficiently* compute $\mathsf{Pr}[\mathcal{F}(Z_1, \ldots, Z_n) \mid Z_1 = z_1, \ldots, Z_i = z_i]$, which is hard to do exactly for general distributions. However, as shown by Raghavan [Rag88], it often suffices to consider *pessimistic estimators* to upper bound the *failure* probability.

7.3.2 Pessimistic Estimators: Idea and Application

In the following the notion of pessimistic estimators is first exemplarily introduced and then applied to efficiently bound the failure probabilities within the VNEP approximations, namely not obtaining a sufficiently large profit and exceeding resources beyond factors β and γ.

To explain the notion of pessimistic estimators, we consider the following example. Let $X = \sum_{i \in [n]} X_i$ be a sum of independent random variables $X_i \in [0, 1]$ and let the failure function be

$$\mathcal{G}(x_1, \ldots, x_n) = \begin{cases} 0 & \text{if } \sum_{i \in [n]} x_i < (1+\delta) \cdot \hat{\mu} \\ 1 & \text{if } \sum_{i \in [n]} x_i \geq (1+\delta) \cdot \hat{\mu} \end{cases},$$

for $\hat{\mu} \geq \mathsf{E}[Z]$. While computing $\mathsf{Pr}[\mathcal{G}(X_1, \ldots, X_n) \mid X_1 = x_1, \ldots, X_i = x_i]$ exactly is generally hard, Raghavan [Rag88] proposed to apply well-known techniques stemming from the analysis of Chernoff bounds to bound the probability. In particular, for $\mathsf{Pr}[X \geq (1+\delta) \cdot \hat{\mu}]$ the proof of Theorem A.1 yields that the following holds for any $t > 0$:

$$\mathsf{Pr}\left[X = \sum\nolimits_{i \in [n]} X_i \geq (1+\delta) \cdot \hat{\mu}\right] \leq \frac{\mathsf{E}\left[\exp(t \cdot X)\right]}{\exp(t \cdot (1+\delta) \cdot \hat{\mu})} = \frac{\prod_{i \in [n]} \exp(t \cdot \mathsf{E}[X_i])}{\exp(t \cdot (1+\delta) \cdot \hat{\mu})}. \tag{7.16}$$

Hence, one can choose the following function as the pessimistic estimator:

$$\mathcal{E}(x_1, \ldots, x_n) = \prod_{i \in [n]} \exp(t \cdot x_i) / \exp(t \cdot (1+\delta) \cdot \hat{\mu}).$$

Then the following holds for the conditional expectation for $X_1 = x_1, \ldots, X_i = x_i$:

$$\begin{aligned} \mathsf{E}_{X_{i+1}, \ldots, X_n}[\mathcal{G}(x_1, \ldots, x_i, X_{i+1}, \ldots, X_n)] \leq & \mathsf{E}_{X_{i+1}, \ldots, X_n}[\mathcal{E}(x_1, \ldots, x_i, X_{i+1}, \ldots, X_n)] \\ = & \frac{\prod_{j \in [i]} \exp(t \cdot x_j) \cdot \prod_{j=i+1}^{n} \exp(t \cdot \mathsf{E}[X_i])}{\exp(t \cdot (1+\delta) \cdot \hat{\mu})} \end{aligned} \tag{7.17}$$

Given the above inequality, the conditional expectation under the pessimistic estimator $\mathcal{E}$ can be computed efficiently as long as the expectation $\mathsf{E}[X_i]$ is known or if it can be efficiently computed. The last 'trick' is now the following: if $\mathsf{Pr}\left[X \geq (1+\delta) \cdot \hat{\mu}\right] = \mathsf{E}[\mathcal{G}(X_1, \ldots, X_n)] = \epsilon < 1$ was *proven* using a Chernoff bound, then for the pessimistic estimator $\mathcal{E}$ initially also $\mathsf{E}_{X_1, \ldots, X_n}[\mathcal{E}(X_1, \ldots, X_n)] \leq \varepsilon < 1$ holds by the remaining parts of the proof of the *same* Chernoff bounds. Importantly, this implies that for the pessimistic estimator $\mathcal{E}$ of $\mathcal{G}$ the same value t as in the Chernoff analysis needs to be used. Hence, considering the above example, according to the proof of Theorem A.1, $t = \ln(1+\delta)$ has to be used within the pessimistic estimator $\mathcal{E}$.

In the following, the probabilistic (union) bounds studied in Sections 7.1 and 7.2 are translated into corresponding pessimistic estimators.

7.3.2.1 Generic Pessimistic Estimator for Resource Augmentations

We start by considering pessimistic estimators for the resource augmentations, which will be used for all discussed randomized rounding algorithms. Given that all bounds on resource augmentations are based on Lemmas 7.4 and Lemma 7.5, we analogously first introduce a general pessimistic estimator and will then later on adapt it to the respective profit and cost approximations.

Lemma 7.16. Consider a resource $x \in G_S$ and let $\Delta_x > 0$ be chosen in such a way that $\Delta_x \cdot d_S(x)$ is an upper bound on the expected allocations on $x \in G_S$, i.e., $\Delta_x \cdot d_S(x) \geq \mathsf{E}(A_x)$ holds. Let $\hat{\mu}_x = \Delta_x \cdot d_S(x)/A_{\mathsf{max}}(x)$ and $n \geq 3$ and $c \geq 3$. Let $A'_x(r,k)$ denote the allocation of the k-th mapping of request $r \in \mathcal{R}$ on resource x divided by the maximal allocations, i.e.:

$$A'_x(r,k) = \begin{cases} A(m_r^k, x)/A_{\mathsf{max}}(x) & \text{if } k \in \hat{K}_r \setminus \{0\} \\ 0 & \text{otherwise} \end{cases}.$$

Let $\vec{z} = (z_1, \ldots, z_n) \in \hat{K}_1 \times \ldots \times \hat{K}_n$ denote a selection of mappings, such that if $z_i = k$ holds, then request r_i is embedded using its $z_i = k$-th mapping m_r^k for $k > 0$ and is rejected if $k = 0$. Let $\mathcal{F}^{A_x}_{n,c,\Delta_x,\hat{\mu}_x}(\vec{z}) \in \{0,1\}$ denote the failure function which is 1 iff. the allocation on resource x under mapping selection $\vec{z}$ exceeds the capacity by a factor more than $\Lambda_{\mathsf{gen}}(n,c,\Delta_x,\hat{\mu}_x)$ (cf. Lemma 7.5). The following function $\mathcal{E}^{A_x}_{n,c,\Delta_x,\hat{\mu}_x}$ is a pessimistic estimator of $\mathcal{F}^{A_x}_{n,c,\Delta_x,\hat{\mu}_x}$:

$$\mathcal{E}^{A_x}_{n,c,\Delta_x,\hat{\mu}_x}(\vec{z}) = \begin{cases} \dfrac{\prod_{r_i \in \mathcal{R}} \exp(\ln \lambda \cdot A'_x(r_i, z_i))}{\exp(\lambda \cdot \ln \lambda)} & \text{if } \hat{\mu}_x < 1 \\ & \text{with } \lambda = 2 \cdot c \cdot \ln n / \ln \ln n \\ \dfrac{\prod_{r_i \in \mathcal{R}} \exp(\ln \lambda \cdot A'_x(r_i, z_i))}{\exp(\lambda \cdot \ln \lambda \cdot \hat{\mu}_x)} & \text{if } 1 \leq \hat{\mu}_x < \ln n / \ln \ln n \\ & \text{with } \lambda = 2 \cdot c \cdot \hat{\mu}_x^{-1} \cdot \ln n / \ln \ln n \\ \dfrac{\prod_{r_i \in \mathcal{R}} \exp(\ln \lambda \cdot A'_x(r_i, z_i))}{\exp(\lambda \cdot \ln \lambda \cdot \hat{\mu}_x)} & \text{if } \ln n / \ln \ln n \leq \hat{\mu}_x < 3 \cdot c \ln n \\ & \text{with } \lambda = 2 \cdot c \\ \dfrac{\prod_{r_i \in \mathcal{R}} \exp(\ln \lambda \cdot A'_x(r_i, z_i))}{\exp(\lambda \cdot \ln \lambda \cdot \hat{\mu}_x)} & \text{if } 3 \cdot c \cdot \ln n \leq \hat{\mu}_x \\ & \text{with } \lambda = 1 + \sqrt{3 \cdot c \cdot \ln n / \hat{\mu}_x} \end{cases} \tag{7.18}$$

Specifically, letting $\vec{Z} = (Z_1, \ldots, Z_n) \in \hat{K}_1 \times \ldots \times \hat{K}_n$ denote the vector of random variables indicating the selection of mappings, i.e., $\mathsf{Pr}[Z_i = k] = f_r^k$ and $\mathsf{Pr}[Z_i = 0] = 1 - \sum_k f_r^k$, the following holds:

(1) $\mathsf{E}_{\vec{Z}}[\mathcal{F}^{A_x}_{n,c,\Delta_x,\hat{\mu}_x}(\vec{Z})] \leq \mathsf{E}_{\vec{Z}}[\mathcal{E}^{A_x}_{n,c,\Delta_x,\hat{\mu}_x}(\vec{Z})] \leq 1/n^c$

(2) $\mathsf{E}_{Z_{i+1},\ldots,Z_n}[\mathcal{F}^{A_x}_{n,c,\Delta_x,\hat{\mu}_x}(z_1,\ldots,z_i,Z_{i+1},\ldots,Z_n)] \leq \mathsf{E}_{Z_{i+1},\ldots,Z_n}[\mathcal{E}^{A_x}_{n,c,\Delta_x,\hat{\mu}_x}(z_1,\ldots,z_i,Z_{i+1},\ldots,Z_n)]$

(3) $\mathsf{E}_{Z_{i+1},\ldots,Z_n}[\mathcal{E}^{A_x}_{n,c,\Delta_x,\hat{\mu}_x}(z_1,\ldots,z_i,Z_{i+1},\ldots,Z_n)]$ can be computed in time polynomial in the number of returned mappings for any prefix length i and any choice of assignments $z_1,\ldots,z_i$.

Proof. We first note that the function $A'_x(r,k)$ correctly computes the (scaled) allocations on resource x and that for the random variable $A'_{r,x}$, introduced in the analysis of resource augmentations in Section 7.1.2, $A'_{r_i,x} = A'_x(r_i, Z_i)$ holds. Hence, the application of the Chernoff bounds on $A'_x = \sum_r A'_{r,x}$ can be equally stated in terms of the sum of the random variables $\vec{Z}$ via $\sum_{r_i \in \mathcal{R}} A'_x(r_i, Z_i)$.

Next, we note that the different cases of Equation 7.18 are all derived by tracing the origins of the results of Lemma 7.4 back to the application of the Chernoff bound of Theorem A.1. For example, in Lemma 7.4 for the first case, i.e., if $\hat{\mu}_x < 1$ holds, the upper bound used in the analysis was set to 1 and accordingly it does not occur in the denominator. Notably, according to this analysis, the bounds λ slightly differ

compared to the value of Λ_{gen}, as, e.g., the factor Δ_x does not occur in the function $E^{A_x}_{n,c,\Delta_x,\hat{\mu}_x}$ while being used in Λ_{gen}. This difference is due to the fact that the Chernoff bounds use the scaled random variables A'_x albeit introducing additional factors as Δ_x when restating these bounds in terms of the original random variables A_x as $A_{\mathsf{max}}(x) = \Delta_x \cdot d_S(x)/\hat{\mu}_x$ holds.

By the argument that the different cases of Equation 7.18 stem from the analysis of the Chernoff bound used in Lemma 7.4, the first and second statements of this lemma follow. The third statement of this lemma follows from the applicability of Equation 7.17, as the expectation $\mathsf{E}[A'_x(r_i, Z_i)] = \sum_{(f^k_r, m^k_r) \in \mathcal{D}_{r_i}} f^k_r \cdot A(m^k_r, x)/A_{\mathsf{max}}(x)$ can be readily computed in time polynomial in the size of $\sum_{r \in \mathcal{R}} |\mathcal{D}_r|$. ■

The above introduced pessimistic estimator for exceeding the capacity of a single resource can easily be generalized to obtain a pessimistic estimator for the failure probability of exceeding the capacity of *any* resource. Intuitively, and reflecting the union bound argument of Lemma 7.5, this generalized pessimistic estimator is obtained by simply summing up the respective estimators for each resource.

Corollary 7.17. Let $\Delta_x > 0$ be chosen in such a way that $\Delta_x \cdot d_S(x)$ is an upper bound on the expected allocations on $x \in G_S$, i.e., $\Delta_x \cdot d_S(x) \geq \mathsf{E}(A_x)$ holds. Let $\hat{\mu}_x = \Delta_x \cdot d_S(x)/A_{\mathsf{max}}(x)$ and $n \geq 3$ and $c \geq 3$. Let Δ_V, Δ_x, $\hat{\mu}_V$, $\hat{\mu}_E$, and Λ_{gen} be as defined in Lemma 7.5.

We denote by $\mathcal{F}^A_{n,c,\Delta_V,\Delta_E,\hat{\mu}_V,\hat{\mu}_E}(\vec{z}) \in \{0,1\}$ the function indicating the failure of constructing a solution such that the allocations on each node resource $u \in V_S$ are bounded by a factor $\Lambda_{\mathsf{gen}}(n, c, \Delta_V, \hat{\mu}_V)$ and the allocations on each edge resource $(u, v) \in E_S$ are bounded by a factor $\Lambda_{\mathsf{gen}}(n, c, \Delta_E, \hat{\mu}_E)$. The function

$$\mathcal{E}^A_{n,c,\Delta_V,\Delta_E,\hat{\mu}_V,\hat{\mu}_E}(\vec{z}) = \sum_{u \in V_S} \mathcal{E}^{A_u}_{n,c,\Delta_V,\hat{\mu}_u}(\vec{z}) + \sum_{(u,v) \in E_S} \mathcal{E}^{A_{u,v}}_{n,c,\Delta_E,\hat{\mu}_{u,v}}(\vec{z}) \tag{7.19}$$

is a pessimistic estimator for $\mathcal{F}^A_{n,c,\Delta_V,\Delta_E,\hat{\mu}_V,\hat{\mu}_E}$, such that the following holds:

(1) $\mathsf{E}_{\vec{Z}}[\mathcal{F}^A_{n,c,\Delta_V,\Delta_E,\hat{\mu}_V,\hat{\mu}_E}(\vec{Z})] \leq \mathsf{E}_{\vec{Z}}[\mathcal{E}^A_{n,c,\Delta_V,\Delta_E,\hat{\mu}_V,\hat{\mu}_E}(\vec{Z})] \leq 1/n^{c-2}$

(2) $\mathsf{E}_{Z_{i+1},\ldots,Z_n}[\mathcal{F}^A_{n,c,\Delta_V,\Delta_E,\hat{\mu}_V,\hat{\mu}_E}(z_1, ..., Z_{i+1}, ...)] \leq \mathsf{E}_{Z_{i+1},\ldots,Z_n}[\mathcal{E}^A_{n,c,\Delta_V,\Delta_E,\hat{\mu}_V,\hat{\mu}_E}(z_1, ..., Z_{i+1}, ...)]$

(3) $\mathsf{E}_{Z_{i+1},\ldots,Z_n}[\mathcal{E}^{A_x}_{n,c,\Delta_x,\hat{\mu}_x}(z_1, \ldots, z_i, Z_{i+1}, \ldots, Z_n)]$ can be computed in time polynomial in the number of returned mappings for any prefix length i and any choice of assignments $z_1, \ldots, z_i$.

Note that in each summand of the above pessimistic estimator $\mathcal{F}^A_{n,c,\Delta_V,\Delta_E,\hat{\mu}_V,\hat{\mu}_E}$ the maximal values Δ_V and Δ_E are used while employing the resource specific values $\hat{\mu}_u$ and $\hat{\mu}_{u,v}$. This again reflects the analysis of Lemma 7.5. Analogous to the application of Lemma 7.5 to obtain specific bounds for the different approximation algorithms (cf. Corollaries 7.6 and 7.10), the above presented generic pessimistic estimator will eventually be slightly adapted by considering specific values Δ_V and Δ_E etc.

7.3.2.2 Pessimistic Estimator for the Obtained Profit

While the objective approximation guarantee is deterministic for the cost approximations, it is probabilistic for the profit approximations. Hence, to derandomize the approximations for the profit, the additional failure source of not achieving an α-factor of the profit has to be taken into account using an additional pessimistic estimator. While the pessimistic estimator for the resource augmenations was based on the Chernoff bound of Theorem A.1, the profit bounds are obtained via the different Chernoff bound stated in Theorem A.2. However, the respective analysis only changes slightly and we state the following:

Lemma 7.18. Let $b_{\mathsf{max}} = \max_{r \in \mathcal{R}}$ denote the maximal profit of any request, b_{frac} denote the profit achieved by the fractional solution and $\tilde{\mu} = b_{\mathsf{frac}}/b_{\mathsf{max}}$. Also, let $B'(r,k)$ denote the scaled profit of request r when embedding it using the k-th mapping, i.e.:

$$B'(r,k) = \begin{cases} b_r/b_{\mathsf{max}} & \text{if } k \in \hat{K}_r \setminus \{0\} \\ 0 & \text{otherwise} \end{cases}.$$

Let $\vec{z} = (z_1, \ldots, z_n) \in \hat{K}_1 \times \ldots \times \hat{K}_n$ denote a selection of mappings, such that if $z_i = k$ holds, then request r_i is embedded using its $z_i = k$-th mapping m_r^k for $k > 0$ and is rejected if $k = 0$. Let $\mathcal{F}_{\tilde{\mu}}^B(\vec{z}) \in \{0,1\}$ denote the failure function which is 1 iff. the cumulative profit under mapping selection $\vec{z}$ is less than a factor $\alpha = 1 - 2/3 \cdot \sqrt{\tilde{\mu}^{-1}}$ of $\tilde{\mu}$. The following function $\mathcal{E}_{\tilde{\mu}}^B$ is a pessimistic estimator for $\mathcal{F}_{\tilde{\mu}}^B$:

$$\mathcal{E}_{\tilde{\mu}}^B(\vec{z}) = \frac{\prod_{r_i \in \mathcal{R}} \exp(\ln \alpha \cdot B'(r_i, z_i))}{\exp(\alpha \cdot \ln \alpha \cdot \tilde{\mu})}. \tag{7.20}$$

Specifically, letting $\vec{Z} = (Z_1, \ldots, Z_n) \in \hat{K}_1 \times \ldots \times \hat{K}_n$ denote the vector of random variables indicating the selection of mappings, i.e., $\mathsf{Pr}[Z_i = k] = f_r^k$ and $\mathsf{Pr}[Z_i = 0] = 1 - \sum_k f_r^k$, the following holds:

(1) $\mathsf{E}_{\vec{Z}}[\mathcal{F}_{\tilde{\mu}}^B(\vec{Z})] \leq \mathsf{E}_{\vec{Z}}[\mathcal{E}_{\tilde{\mu}}^B(\vec{Z})] \leq \exp(-2/9)$

(2) $\mathsf{E}_{Z_{i+1},\ldots,Z_n}[\mathcal{F}_{\tilde{\mu}}^B(z_1, \ldots, z_i, Z_{i+1}, \ldots, Z_n)] \leq \mathsf{E}_{Z_{i+1},\ldots,Z_n}[\mathcal{E}_{\tilde{\mu}}^B(z_1, \ldots, z_i, Z_{i+1}, \ldots, Z_n)]$

(3) $\mathsf{E}_{Z_{i+1},\ldots,Z_n}[\mathcal{E}_{\tilde{\mu}}^B(z_1, \ldots, z_i, Z_{i+1}, \ldots, Z_n)]$ can be computed in time polynomial in the number of returned mappings for any prefix length i and any choice of assignments $z_1, \ldots, z_i$.

Proof. We first note that the function $B'(r,k)$ correctly computes the (scaled) profit by using the k-th mapping of request r to embed it. Hence, considering the random variable B'_r introduced in Section 7.2.1, $B'_{r_i} = B'(r_i, Z_i)$ holds. Thus, the application of the Chernoff bound on $B' = \sum_{r \in \mathcal{R}} B'_r$ can be equivalently stated by using $\sum_{r_i \in \mathcal{R}} B'(r_i, Z_i)$.

The estimator is based on the analysis of the application of the Chernoff bound of Theorem A.2 in Lemma 7.9. Specifically, the bound $\mathsf{Pr}[X \leq (1-\delta) \cdot \tilde{\mu}] \leq \mathsf{E}[\exp(t \cdot X)] / \exp(t \cdot (1-\delta) \cdot \tilde{\mu})$ was obtained in Theorem A.2 by the application of Markov's inequality. By choosing $\delta = 2/3 \cdot \sqrt{\tilde{\mu}^{-1}}$ and $t = \ln(1-\delta)$, the validity of Equation 7.20 with respect to the second statement is obtained.

The first statement follows from the further application of the Chernoff bound of Theorem A.2 in Lemma 7.9.

With respect to the last statement on the computational tractability of computing the conditional expectation of $\mathcal{E}_{\tilde{\mu}}^B$, we note that $\mathsf{E}[B'(r_i, Z_i)] = \mathsf{E}[B'_{r_i}] = b_r/b_{\mathsf{max}} \cdot \sum_k f_r^k$ holds. As the random variables in $\vec{Z}$ are independent, the conditional expectation can be computed as follows based on Equation 7.20:

$$\begin{aligned} &\mathsf{E}_{Z_{i+1},\ldots,Z_n}[\mathcal{F}_{\tilde{\mu}}^B(z_1, \ldots, z_i, Z_{i+1}, \ldots, Z_n)] \\ \leq &\mathsf{E}_{Z_{i+1},\ldots,Z_n}[\mathcal{E}_{\tilde{\mu}}^B(z_1, \ldots, z_i, Z_{i+1}, \ldots, Z_n)] \\ = &\frac{\prod_{j \in [i]} \exp(\ln \alpha \cdot B'(r_j, z_j) \cdot \prod_{j=i+1}^{n} \exp(\ln \alpha \cdot \mathsf{E}[B'(r_i, Z_i)])}{\exp(\alpha \cdot \ln \alpha \cdot \tilde{\mu})}. \end{aligned} \tag{7.21}$$

■

Algorithm 7.4: Generic Deterministic Rounding using the Method of Conditional Expectations

Input : Set of requests $\mathcal{R}$, substrate G_S, fractional solution $\{\mathcal{D}_r\}_{r\in\mathcal{R}}$,
failure function $\mathcal{F}: \hat{K}_1 \times ... \times \hat{K}_n \to \{0,1\}$, s.t. $\mathcal{F}(\vec{z}) = 1$ iff. $\vec{z}$ is *not* (α,β,γ)-approximate,
pessimistic estimator $\mathcal{E}: \hat{K}_1 \times ... \times \hat{K}_n \to \mathbb{R}_{\geq 0}$ of $\mathcal{F}$, such that
(1) $\mathsf{E}_{\vec{Z}}[\mathcal{F}(\vec{Z})] \leq \mathsf{E}_{\vec{Z}}[\mathcal{E}(\vec{Z})] < 1$,
(2) $\mathsf{E}_{Z_{i+1},...,Z_n}[\mathcal{F}(z_1,\ldots,z_i,Z_{i+1},\ldots,Z_n)] \leq \mathsf{E}_{Z_{i+1},...,Z_n}[\mathcal{E}(z_1,\ldots,z_i,Z_{i+1},\ldots,Z_n)]$,
(3) $\mathsf{E}_{Z_{i+1},...,Z_n}[\mathcal{E}(z_1,\ldots,z_i,Z_{i+1},\ldots,Z_n)]$ can be efficiently computed.

[**Reminder:** $Z_i \in \hat{K}_i$ denote the mapping selection random variable for $r_i \in \mathcal{R}$ with $\mathsf{Pr}[Z_i = k] = f_r^k$ and $\mathsf{Pr}[Z_i = 0] = 1 - \sum_k f_r^k$]

foreach $r_i \in \mathcal{R}$ **do**
1 **for** $k \leftarrow 0$ **to** $\hat{K}_i$ **do** // iterate over all mapping indices for request r_i
2 **if** $\mathsf{Pr}[Z_i = k] > 0$ **then** // only consider mappings of positive probability
3 **set** $z_i = k$ //(temporarily) select k-th mapping for request r_i
4 **if** $\mathsf{E}_{Z_{i+1},...,Z_n}[\mathcal{E}(z_1,...,z_i,Z_{i+1},...,Z_n)] \leq \mathsf{E}_{Z_i,...,Z_n}[\mathcal{E}(z_1,...,z_{i-1},Z_i,...,Z_n)]$ **then**
// failure estimate lies below expectation: choose k-th mapping
5 **if** $k = 0$ **then**
6 **set** $\hat{m}_{r_i} \leftarrow \emptyset$ // request r_i is not embedded
7 **else**
8 **set** $\hat{m}_{r_i} \leftarrow m_r^k$ // request r_i is embedded using m_r^k
9 **break** // continue with the next request
10 **return** $\{\hat{m}_{r_i}\}_{r_i\in\mathcal{R}}$

7.3.3 Deterministic Approximation Results

In the following, we show that all probabilistic randomized rounding approximations can be derandomized using the above introduced pessimistic estimators. We first discuss the general deterministic approximation scheme using a general, unspecified pessimistic estimator (see Algorithm 7.4) and then afterwards discuss the actually chosen estimators for the different approximations.

Algorithm 7.4 implements the method of conditional expectation (cf. discussion in Section 7.3.1). Specifically, besides being given the fractional solution as convex combination $\mathcal{D}_r$ for each request $r \in \mathcal{R}$, a failure function $\mathcal{F}: \hat{K}_1 \times \ldots \times \hat{K}_n \to \{0,1\}$ together with a pessimistic estimator $\mathcal{E}: \hat{K}_1 \times \ldots \times \hat{K}_n \to \mathbb{R}_{\geq 0}$ has to be specified. The failure function will be 1 if the constructed solution is *not* (α,β,γ)-approximate. Using the method of conditional expectation, i.e., constructing a solution by iteratively considering the estimation of the failure estimator and choosing any of the mappings such that the estimate of the failure expectation does not increase, an (α,β,γ)-approximate solution is obtained as long as the initial failure estimation expectation lies below 1. The following theorem formalizes the correctness of Algorithm 7.4.

Theorem 7.19. Algorithm 7.4 constructs an (α,β,γ)-approximate VNEP solution in time $\mathcal{O}\left(\mathsf{poly}\left(|G_S| \cdot \sum_{r\in\mathcal{R}} |\mathcal{D}_r|\right)\right)$ when its prerequisites are met.

Proof. The algorithm iteratively tries all mapping possibilities for each request. Under the assumption that initially $\mathsf{E}_{\vec{Z}}[\mathcal{E}(\vec{Z})] = \epsilon < 1$ holds, the estimated failure expectation decreases after having selected a mapping for a request. Specifically, the following holds:

$$\begin{aligned} 1 > \epsilon =& \mathsf{E}_{Z_1,...,Z_n}[\mathcal{F}(Z_1,\ldots,Z_n)] && \text{(by assumption)} \\ \geq& \mathsf{E}_{Z_2,...,Z_n}[\mathcal{F}(z_1,Z_2,\ldots,Z_n)] && \text{(choice of appropriate } z_1\text{)} \\ \geq& \mathsf{E}_{Z_3,...,Z_n}[\mathcal{F}(z_1,z_2,Z_3,\ldots,Z_n)] && \text{(choice of appropriate } z_2\text{)} \end{aligned}$$

$$\vdots$$

$$\geq \mathsf{E}[\mathcal{F}(z_1, \ldots, z_n)] = \mathcal{F}(z_1, \ldots, z_n) \qquad \text{(choice of appropriate } z_n\text{)}$$

As $\mathcal{F}(z_1, \ldots, z_n) \leq \epsilon < 1$ holds after processing each request, $\mathcal{F}(z_1, \ldots, z_n) = 0$ has to hold, proving that the constructed solution is indeed (α, β, γ)-approximate. We note that the probabilistic method establishes that appropriate mappings z_1, z_2, etc. can always be found.

With respect to the runtime, we note that all (potential) expectations used in the pessimistic estimator $\mathcal{E}$ can be computed once a priori. Furthermore, we assume that the pessimistic estimator $\mathcal{E}$ uses only up to $\mathcal{O}(|G_S|)$ many estimators itself, as there are at most $|G_S|$ substrate resources to consider for bounding the failure probability of obtaining a (β, γ)-approximate solution while at most one estimator will be used to bound the probability of not obtaining an α-approximate solution. As overall at most $\mathcal{O}\left(\sum_{r \in \mathcal{R}} |\mathcal{D}_r|\right)$ expectations have to be computed for each of the considered pessimistic estimators and the evaluation of $\mathcal{E}$ afterwards lies in $\mathcal{O}\left(\mathsf{poly}\left(|G_S| \cdot \sum_{r \in \mathcal{R}} |\mathcal{D}_r|\right)\right)$, the result follows. ■

Given the generic deterministic rounding scheme, we now proceed to show that all probabilistic approximation results stated in this chapter can indeed be derandomized. To this end, it suffices to construct appropriate pessimistic estimators using Corollary 7.17 and Lemma 7.18.

Theorem 7.20. Assume that $n = |V_S| \geq 3$ and let $\varepsilon_{\mathrm{L}} > 0$. Let $\hat{\mu}_V^{\mathsf{cost}}, \hat{\mu}_E^{\mathsf{cost}}, \hat{\mu}_V^{\mathsf{cost,L}}, \hat{\mu}_E^{\mathsf{cost,L}}$ and the function $\Lambda_{\mathsf{cost}}(n, \Delta, \hat{\mu})$ be as defined Corollary 7.6 and Theorem 7.8.

For the offline cost VNEP the following *deterministic* (α, β, γ)-approximations exist:

Setting	$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$	$\boldsymbol{\gamma}$
$\langle$ **VE** $\mid$ **NR** $\rangle$	2	$\Lambda_{\mathsf{cost}}(n, 2, \hat{\mu}_V^{\mathsf{cost}})$	$\Lambda_{\mathsf{cost}}(n, 2, \hat{\mu}_E^{\mathsf{cost}})$
$\langle$ **VE** $\mid$ **NRL** $\rangle$	$2 \cdot (1 + \varepsilon_{\mathrm{L}})$	$\Lambda_{\mathsf{cost}}(n, 2 \cdot (1 + \varepsilon_{\mathrm{L}}), \hat{\mu}_V^{\mathsf{cost,L}})$	$\Lambda_{\mathsf{cost}}(n, \cdot(1 + \varepsilon_{\mathrm{L}}), \hat{\mu}_E^{\mathsf{cost,L}}))$

The runtime of the respective deterministic approximations is polynomial in the time to construct the optimal and $(1 + \varepsilon_{\mathrm{L}}, 1 + \varepsilon_{\mathrm{L}}, 1 + \varepsilon_{\mathrm{L}})$-approximate fractional solutions, respectively.

Proof. To obtain the result the rounding procedure of Algorithm 7.1 (Lines 6 to 8) is adapted by using the deterministic rounding of Algorithm 7.4. To this end, in the setting $\langle$ **VE** $\mid$ **NR** $\rangle$, the failure function $\mathcal{F}_{\mathsf{cost}}$, with $\mathcal{F}_{\mathsf{cost}}(\vec{z}) = 1$ iff. the solution indicated by $\vec{z}$ exceeds the capacity on any resource by a factor larger than $\Lambda_{\mathsf{cost}}(n, 2, \hat{\mu}_V^{\mathsf{cost}})$ (for nodes) or $\Lambda_{\mathsf{cost}}(n, 2, \hat{\mu}_E^{\mathsf{cost}})$ (for edges), is used. According to Corollary 7.17 and the definition of Λ_{cost}, the function $\mathcal{E}_{\mathsf{cost}} = \mathcal{E}^A_{n,3,2,2,\hat{\mu}_V^{\mathsf{cost}},\hat{\mu}_E^{\mathsf{cost}}}$ is a pessimistic estimator for $\mathcal{F}_{\mathsf{cost}}$. As $\mathcal{E}_{\mathsf{cost}}$ satisfies all prerequisites of the deterministic rounding algorithm by Corollary 7.17, the deterministic approximation result follows from the correctness of the deterministic rounding algorithm (cf. Theorem 7.19).

By the same arguments a slightly adapted result is obtained for the setting $\langle$ **VE** $\mid$ **NRL** $\rangle$. Exchanging again Lines 6-8 of Algorithm 7.1 with the execution of Algorithm 7.4 the result is obtained using the following adapted failure indicator $\mathcal{F}_{\mathsf{cost}}$ and its adapted pessimistic estimator $\mathcal{E}_{\mathsf{cost}}$. Specifically, let $\mathcal{F}_{\mathsf{cost}}(\vec{z}) = 1$ iff. the solution indicated by $\vec{z}$ exceeds the capacity on any resource by a factor larger than $\Lambda_{\mathsf{cost}}(n, 2 \cdot (1 + \varepsilon_{\mathrm{L}}), \hat{\mu}_V^{\mathsf{cost,L}})$ (for nodes) or $\Lambda_{\mathsf{cost}}(n, 2 \cdot (1 + \varepsilon_{\mathrm{L}}), \hat{\mu}_E^{\mathsf{cost,L}})$ (for edges). Accordingly, by choosing $\mathcal{E}_{\mathsf{cost}} = \mathcal{E}^A_{n,3,2\cdot(1+\varepsilon_{\mathrm{L}}),2\cdot(1+\varepsilon_{\mathrm{L}}),\hat{\mu}_V^{\mathsf{cost,L}},\hat{\mu}_E^{\mathsf{cost,L}}}$ the result follows again by Corollary 7.17 and Theorem 7.19. ■

Theorem 7.21. Assume that $n = |V_S| \geq 3$. Let $\varepsilon_{\mathsf{scal}}$ and ε_{L} be chosen such that $\delta_{\max} \leq \varepsilon_{\mathsf{scal}} \leq 1$ and $\varepsilon_{\mathrm{L}} > 0$ hold. Let $\hat{\mu}_V^{\mathsf{scal}}, \hat{\mu}_E^{\mathsf{scal}}$ and the function $\Lambda_{\mathsf{prof}}^{\mathsf{scal}}$ be as defined in Theorem 7.15.

Denoting by b_{frac} the profit achieved by the fractional solution (before scaling down the weights) and by b_{max} the maximal benefit of any request, for the offline profit VNEP the following *deterministic* (α, β, γ)-approximations exist:

Setting	$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$	$\boldsymbol{\gamma}$
$\langle\, \mathbf{VE} \,|\, \mathbf{NR} \,\rangle$	$\varepsilon_{\mathsf{scal}} \cdot \left(1 - \frac{2}{3}\sqrt{\frac{b_{\mathsf{frac}}}{b_{\mathsf{max}}}}\right) \geq \varepsilon_{\mathsf{scal}}/3$	$\Lambda^{\mathsf{scal}}_{\mathsf{prof}}(n, \varepsilon_{\mathsf{scal}}, \hat{\mu}^{\mathsf{scal}}_V)$	$\Lambda^{\mathsf{scal}}_{\mathsf{prof}}(n, \varepsilon_{\mathsf{scal}}, \hat{\mu}^{\mathsf{scal}}_E)$
$\langle\, \mathbf{VE} \,|\, \mathbf{NRL} \,\rangle$	$\varepsilon_{\mathsf{scal}} \cdot \left(\frac{1 - \frac{2}{3}\sqrt{\frac{b_{\mathsf{frac}}}{b_{\mathsf{max}}}}}{(1+\varepsilon_{\mathrm{L}})}\right) \geq \varepsilon_{\mathsf{scal}}/3 \cdot (1+\varepsilon_{\mathrm{L}})^{-3/2}$	$\Lambda^{\mathsf{scal}}_{\mathsf{prof}}(n, \varepsilon_{\mathsf{scal}}, \hat{\mu}^{\mathsf{scal}}_V)$	$\Lambda^{\mathsf{scal}}_{\mathsf{prof}}(n, \varepsilon_{\mathsf{scal}}, \hat{\mu}^{\mathsf{scal}}_E)$

The runtime of the respective approximations is polynomial in the time to construct the optimal and $(1+\varepsilon_{\mathrm{L}})$-approximate fractional solutions, respectively.

Proof. The results stated in this Theorem are based on the Theorem 7.15 which generalizes the results of Theorems 7.11 and 7.13 by scaling down the convex combination weights by a factor $\varepsilon_{\mathsf{scal}} \in [\delta_{\max}, 1]$ after having computed the LP solution. Specifically, by Theorem 7.15 the Algorithms 7.2 and 7.3 are adapted such that for $(f^k_r, m^k_r) \in \mathcal{D}_r$ the weight f^k_r is multiplied by the factor $\varepsilon_{\mathsf{scal}}$ for all requests $r \in \mathcal{R}$. Notably, by choosing $\varepsilon_{\mathsf{scal}} = 1$ the original results stated in Theorems 7.11 and 7.13 are obtained.

Given this scaling, the results of this Theorem are obtained by employing the deterministic rounding of Algorithm 7.4 instead of applying randomized rounding. To deterministically round the solution, the following failure indicator function $\mathcal{F}^{\mathsf{scal}}_{\mathsf{prof}}$ is used. For a mapping selection $\vec{z}$ we set $\mathcal{F}^{\mathsf{scal}}_{\mathsf{prof}}(\vec{z}) = 1$ iff. the solution $\vec{z}$ achieves a profit of less than $b'_{\mathsf{frac}} \cdot (1 - 2/3 \cdot \sqrt{b_{\mathsf{max}}/b'_{\mathsf{frac}}})$ with $b'_{\mathsf{frac}} = \varepsilon_{\mathsf{scal}} \cdot b_{\mathsf{frac}}$, or if the solution $\vec{z}$ exceeds the capacity of any resource by a factor more than $\beta = \Lambda^{\mathsf{scal}}_{\mathsf{prof}}(n, \varepsilon_{\mathsf{scal}}, \hat{\mu}^{\mathsf{scal}}_V)$ (for nodes) or $\gamma = \Lambda^{\mathsf{scal}}_{\mathsf{prof}}(n, \varepsilon_{\mathsf{scal}}, \hat{\mu}^{\mathsf{scal}}_E)$ (for edges).

Using this failure function both for the setting $\langle\, \mathbf{VE} \,|\, \mathbf{NR} \,\rangle$ and $\langle\, \mathbf{VE} \,|\, \mathbf{NRL} \,\rangle$, we employ the following pessimistic estimator:

$$\mathcal{E}^{\mathsf{scal}}_{\mathsf{prof}}(\vec{z}) = \mathcal{E}^{B}_{\tilde{\mu}^{\mathsf{scal}}}(\vec{z}) + \mathcal{E}^{A}_{n,4,1,1,\hat{\mu}^{\mathsf{scal}}_V,\hat{\mu}^{\mathsf{scal}}_E}(\vec{z})\,. \tag{7.22}$$

Importantly, the sub-estimators $\mathcal{E}^{B}_{\tilde{\mu}^{\mathsf{scal}}}$ of Lemma 7.18 and $\mathcal{E}^{A}_{n,4,1,1,\hat{\mu}^{\mathsf{scal}}_V,\hat{\mu}^{\mathsf{scal}}_E}$ of Corollary 7.17 are constructed with respect to the scaled profits and capacities by using $\tilde{\mu}^{\mathsf{scal}} = b'_{\mathsf{frac}}/b_{\mathsf{max}}$, and $\hat{\mu}^{\mathsf{scal}}_V$ and $\hat{\mu}^{\mathsf{scal}}_E$.

By the probabilistic analyses of Lemmas 7.9 and 7.12 and Theorem 7.15 the sub-estimator $\mathcal{E}^{B}_{\tilde{\mu}^{\mathsf{scal}}}$ correctly bounds the failure probability to not achieve at least an α factor of the optimal profit, with $\alpha = \varepsilon_{\mathsf{scal}} \cdot (1 - 2/3 \cdot \sqrt{b_{\mathsf{frac}}/b_{\mathsf{max}}})$ for $\langle\, \mathbf{VE} \,|\, \mathbf{NR} \,\rangle$ and $\alpha = \varepsilon_{\mathsf{scal}} \cdot (1+\varepsilon_{\mathrm{L}})^{-1} \cdot (1 - 2/3 \cdot \sqrt{b_{\mathsf{frac}}/b_{\mathsf{max}}})$ for the variant $\langle\, \mathbf{VE} \,|\, \mathbf{NRL} \,\rangle$. Considering the approximation factor for the variant $\langle\, \mathbf{VE} \,|\, \mathbf{NRL} \,\rangle$, we note that the additional factor $(1+\varepsilon_{\mathrm{L}})^{-1}$ was only introduced due to the approximate nature of the computed fractional solution but is otherwise not reflected in the analysis of Theorem 7.15.

By the probabilistic analyses of Lemma 7.10 and Theorem 7.15 the sub-estimator $\mathcal{E}^{A}_{n,4,1,1,\hat{\mu}^{\mathsf{scal}}_V,\hat{\mu}^{\mathsf{scal}}_E}$ correctly bounds the failure probability to exceed resource capacities by factors of β or γ for nodes and edges, respectively.

Given Corollary 7.17 and Lemma 7.18, we have that $\mathsf{E}_{\vec{Z}}[\mathcal{E}^{\mathsf{scal}}_{\mathsf{prof}}(\vec{Z})] \leq \exp(-2/9) \leq 65/81$ and $\mathsf{E}_{\vec{Z}}[\mathcal{E}^{A}_{n,4,1,1,\hat{\mu}^{\mathsf{scal}}_V,\hat{\mu}^{\mathsf{scal}}_E}(\vec{Z})] \leq 1/9$ hold. Thus, $\mathsf{E}_{\vec{Z}}[\mathcal{E}^{\mathsf{scal}}_{\mathsf{prof}}(\vec{Z})] \leq 74/81$ holds initially.

Accordingly, the constructed estimator $\mathcal{E}^{\mathsf{scal}}_{\mathsf{prof}}$ satisfies all requirements of Theorem 7.19: it correctly bounds the failure probability $\mathcal{F}^{\mathsf{scal}}_{\mathsf{prof}}$, the initial (unconditional) expectation lies below 1, and the conditional expectation $\mathsf{E}_{Z_{i+1},\ldots,Z_n}[\mathcal{E}^{\mathsf{scal}}_{\mathsf{prof}}(z_1,\ldots,z_i,Z_{i+1},\ldots,Z_n)]$ can be efficiently computed given the efficient computability of the sub-estimators. Hence, the result follows. ∎

7.4 Summary and Discussion of Approximation Results

In all of the above approximation results the resource augmentation factors β and γ depend on the values $\hat{\mu}_V$ and $\hat{\mu}_E$, i.e., the minimal upper bounds of the expected resource allocations divided by the maximal request allocation. Specifically, in Lemma 7.4 these bounds were used for the first time for a single resource $x \in G_S$ and $\hat{\mu}_x = \Delta_x \cdot d_S(x)/A_{\mathsf{max}}(x)$ was defined. Accordingly, to compute $\hat{\mu}_x$ (and thereby $\hat{\mu}_V$ and $\hat{\mu}_E$), one has to exactly compute $A_{\mathsf{max}}(x)$, which in general requires the enumeration of *all* valid mappings of any request. As this is generally computationally prohibitive, in the following general bounds for the value $A_{\mathsf{max}}(x)$ are derived and the approximation results are restated in terms of these upper bounds. Furthermore, in this section the runtime for computing (approximate) fractional offline VNEP solutions are considered as well, showing that all presented approximations indeed have a XP-runtime with respect to the parameters of the treewidth and the extraction width. Lastly, at the end of this section, we analyze under which conditions constant factor approximations can be obtained.

To bound $A_{\mathsf{max}}(x)$ we define the following. We denote by $\kappa_V = \max_{r \in \mathcal{R}} |V_r|$ and by $\kappa_E = \max_{r \in \mathcal{R}} |E_r|$ the maximal number of nodes and edges of any request graph and by $\kappa_{\mathsf{max}} = \max\{\kappa_V, \kappa_E\}$ the maximum of either. Denoting by δ_V, δ_E, and $\delta_{\max}$ the maximal demand-to-capacity ratios for nodes, edges, and all resources (cf. Definition 2.4), respectively, the following holds:

$$A_{\mathsf{max}}(u) \leq \kappa_V \cdot \delta_V \cdot d_S(u) \quad \text{for all } u \in V_S \,, \text{and} \tag{7.23}$$

$$A_{\mathsf{max}}(u,v) \leq \kappa_E \cdot \delta_E \cdot d_S(u,v) \text{ for all } (u,v) \in E_S \,. \tag{7.24}$$

Intuitively, the above bounds hold, as each node or edge resource may be used by at most κ_V or κ_E virtual elements of the same request and each of these elements may induce an allocation of at most δ_V or δ_E times the resource's capacity. To further simplify the presentation of the results, the following unified bound holds for all resources, nodes and edges alike by considering the respective maxima of κ_V and κ_E and δ_V and δ_E, respectively:

$$A_{\mathsf{max}}(x) \leq \kappa_{\mathsf{max}} \cdot \delta_{\max} \cdot d_S(u) \text{ for all } x \in G_S \,. \tag{7.25}$$

Now, to derive bounds for $\hat{\mu}_V$ and $\hat{\mu}_E$, consider the following. According to Lemma 7.5, $\hat{\mu}_V$ and $\hat{\mu}_E$ were defined such that $\hat{\mu}_V \leq \Delta_V \cdot d_S(u)/A_{\mathsf{max}}(u)$ and $\hat{\mu}_E \leq \Delta_E \cdot d_S(u,v)/A_{\mathsf{max}}(u,v)$ hold for node resources $u \in V_S$ and edge resources $(u,v) \in E_S$, respectively, where Δ_V and Δ_E denote the (relative) resource augmentation factor of the fractional LP solution. As for all considered approximations $\Delta_V = \Delta_E = \Delta$ was set (cf. Corollary 7.6, Theorem 7.8, and Corollary 7.10), the following bounds hold with respect to the respective Δ values:

$$\Delta/(\kappa_V \cdot \delta_V) \leq \hat{\mu}_V \quad \text{and} \quad \Delta/(\kappa_E \cdot \delta_V) \leq \hat{\mu}_E \,. \tag{7.26}$$

Again, we note the following bound purely utilizing the respective maxima:

$$\Delta/(\kappa_{\mathsf{max}} \cdot \delta_{\max}) \leq \min\{\hat{\mu}_V, \hat{\mu}_E\} \,. \tag{7.27}$$

By casting the resource augmentation functions Λ_{cost} and $\Lambda^{\mathsf{scal}}_{\mathsf{prof}}$ as a function of this unified lower bound $\Delta/(\kappa_{\mathsf{max}} \cdot \delta_{\max})$, approximations only depending on the easily computable values κ_{max} and $\delta_{\max}$ can be obtained as shown in the theorem below. Furthermore, the following theorem also states the total runtime of the respective approximations by taking the results of Chapter 6 on the tractability of the fractional offline VNEP into account.

Theorem 7.22. Assume that $n = |V_S| \geq 3$, $\varepsilon_{\mathrm{L}} \in O(1)$, $\varepsilon_{\mathsf{scal}} \in [\delta_{\max}, 1]$, and that the maximal request treewidth is at least 1. By employing the results on the tractability of the fractional offline VNEP presented in Chapter 6, specifically Theorems 6.20, 6.39, 6.40, and 6.41, and the deterministic approximation results presented in Theorems 7.20 and 7.21, the deterministic XP-time approximations stated in Table 7.1 using the definitions of Tables 7.2 and 7.3 are obtained for the various different offline VNEP variants.

Objective/ Setting		Approximation Factors α	Approximation Factors $\max\{\beta,\gamma\}$	Runtime
Cost	$\langle$ **VE** $\vert$ **NR** $\rangle$	2	$\bar{\Lambda}_{\mathsf{cost}}(n,\kappa_{\mathsf{max}},\delta_{\max})$	$\mathsf{poly}\left(\lvert\mathcal{R}\rvert\cdot\kappa_{\mathsf{max}}\cdot n^{2\cdot\kappa_{\mathsf{ew}}}\right)$
				$\mathsf{poly}\left(\lvert\mathcal{R}\rvert\cdot\kappa_{\mathsf{max}}^2\cdot n^{2\cdot\kappa_{\mathsf{tw}}+3}\right)$
	$\langle$ **VE** $\vert$ **NRL** $\rangle$	$2\cdot(1+\varepsilon_{\mathrm{L}})$	$\bar{\Lambda}^{\mathsf{L}}_{\mathsf{cost}}(n,\kappa_{\mathsf{max}},\delta_{\max},\varepsilon_{\mathrm{L}})$	$\mathsf{poly}\left(\lvert\mathcal{R}\rvert\cdot\kappa_{\mathsf{max}}^3\cdot n^{2\cdot\kappa_{\mathsf{tw}}+2}\cdot\varepsilon_{\mathrm{L}}^{-1}\right)$
Profit	$\langle$ **VE** $\vert$ **NR** $\rangle$	$\underbrace{\varepsilon_{\mathsf{scal}}\cdot\left(1-\frac{2}{3}\sqrt{\frac{b_{\mathsf{frac}}}{b_{\mathsf{max}}}}\right)}_{\geq\varepsilon_{\mathsf{scal}}/3}$	$\bar{\Lambda}^{\mathsf{scal}}_{\mathsf{prof}}(n,\kappa_{\mathsf{max}},\delta_{\max},\varepsilon_{\mathsf{scal}})$	$\mathsf{poly}\left(\lvert\mathcal{R}\rvert\cdot\kappa_{\mathsf{max}}\cdot n^{2\cdot\kappa_{\mathsf{ew}}}\right)$
				$\mathsf{poly}\left(\lvert\mathcal{R}\rvert\cdot\kappa_{\mathsf{max}}^2\cdot n^{2\cdot\kappa_{\mathsf{tw}}+3}\right)$
	$\langle$ **VE** $\vert$ **NRL** $\rangle$	$\underbrace{\varepsilon_{\mathsf{scal}}\cdot\left(\frac{1-\frac{2}{3}\sqrt{\frac{b_{\mathsf{frac}}}{b_{\mathsf{max}}}}}{1+\varepsilon_{\mathrm{L}}}\right)}_{\geq(1+\varepsilon_{\mathrm{L}})^{-3/2}\cdot\varepsilon_{\mathsf{scal}}/3}$	$\bar{\Lambda}^{\mathsf{scal}}_{\mathsf{prof}}(n,\kappa_{\mathsf{max}},\delta_{\max},\varepsilon_{\mathsf{scal}})$	$\mathsf{poly}\left(\lvert\mathcal{R}\rvert\cdot\kappa_{\mathsf{max}}^3\cdot n^{2\cdot\kappa_{\mathsf{tw}}+2}\cdot\varepsilon_{\mathrm{L}}^{-1}\right)$

Table 7.1: Summary of deterministic approximation results for the offline VNEP.

$$\bar{\Lambda}_{\mathsf{cost}}(n,\kappa,\delta)=\begin{cases}6\cdot\kappa\cdot\delta\cdot\frac{\ln n}{\ln\ln n} & \text{if } \frac{2}{\kappa\cdot\delta}<\frac{\ln n}{\ln\ln n}\\ 12 & \text{if } \frac{\ln n}{\ln\ln n}\leq\frac{2}{\kappa\cdot\delta}<9\ln n\\ 2\left(1+\sqrt{9\ln n\cdot\frac{\kappa\cdot\delta}{2}}\right) & \text{if } \frac{2}{\kappa\cdot\delta}\geq 9\ln n\end{cases} \tag{7.28}$$

$$\bar{\Lambda}^{\mathsf{L}}_{\mathsf{cost}}(n,\kappa,\delta,\varepsilon_{\mathrm{L}})=\begin{cases}6\cdot\kappa\cdot\delta\cdot\frac{\ln n}{\ln\ln n} & \text{if } \frac{2\cdot(1+\varepsilon_{\mathrm{L}})}{\kappa\cdot\delta}<\frac{\ln n}{\ln\ln n}\\ 12\cdot(1+\varepsilon_{\mathrm{L}}) & \text{if } \frac{\ln n}{\ln\ln n}\leq\frac{2\cdot(1+\varepsilon_{\mathrm{L}})}{\kappa\cdot\delta}<9\ln n\\ 2(1+\varepsilon_{\mathrm{L}})\left(1+\sqrt{9\ln n\cdot\frac{\kappa\cdot\delta}{2\cdot(1+\varepsilon_{\mathrm{L}})}}\right) & \text{if } \frac{2\cdot(1+\varepsilon_{\mathrm{L}})}{\kappa\cdot\delta}\geq 9\ln n\end{cases} \tag{7.29}$$

$$\bar{\Lambda}^{\mathsf{scal}}_{\mathsf{prof}}(n,\kappa,\delta,\varepsilon_{\mathsf{scal}})=\begin{cases}8\cdot\kappa\cdot\delta\cdot\frac{\ln n}{\ln\ln n} & \text{if } \frac{\varepsilon_{\mathsf{scal}}}{\kappa\cdot\delta}<\frac{\ln n}{\ln\ln n}\\ 8\cdot\varepsilon_{\mathsf{scal}} & \text{if } \frac{\ln n}{\ln\ln n}\leq\frac{\varepsilon_{\mathsf{scal}}}{\kappa\cdot\delta}<12\ln n\\ (1+\sqrt{12\ln n\cdot\frac{\kappa\cdot\delta}{\varepsilon_{\mathsf{scal}}}})\cdot\varepsilon_{\mathsf{scal}} & \text{if } \frac{\varepsilon_{\mathsf{scal}}}{\kappa\cdot\delta}\geq 12\ln n\end{cases} \tag{7.30}$$

Table 7.2: Definition of functions used in Table 7.1 to bound capacity violations β and γ.

Symbol	Description
$n=\lvert V_S\rvert$	Number of substrate nodes.
$\mathcal{R}$	Set of requests with graphs $G_r=(V_r,E_r)$ for $r\in\mathcal{R}$.
$\kappa_{\mathsf{max}}=\max_{r\in\mathcal{R}}\{\lvert V_r\rvert,\lvert E_r\rvert\}$	Maximal request size with respect to nodes and edges.
$\kappa_{\mathsf{tw}}=\max_{r\in\mathcal{R}}\mathrm{tw}(G_r)$	Maximal treewidth of any request graph.
$\kappa_{\mathsf{ew}}=\max_{r\in\mathcal{R}}\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})$	Maximal extraction width of any of the *pre-computed* extraction orders $G_r^{\mathcal{X}}$ for $r\in\mathcal{R}$.
$\delta_{\max}=\max_{x\in G_S} d_{\mathsf{max}}(x)/d_S(x)$	Maximal demand to capacity ratio.
$\varepsilon_{\mathrm{L}}>0$	Approximation factor for the LCSP FPTAS.
$\varepsilon_{\mathsf{scal}}\in[\delta_{\max},1]$	A posteriori scaling factor for profit approximations.
b_{frac}	Profit of the fractional VNEP solution before scaling.
$b_{\mathsf{max}}=\max_{r\in\mathcal{R}} b_r$	Maximal profit of any of the requests.

Table 7.3: Overview of symbols used in Table 7.1.

Proof. We first turn to the runtime aspects of the respective *deterministic* approximations. According to the Theorems 7.20 and 7.21 the runtimes of the respective approximations are polynomially bounded in the time to compute the respective (approximate or optimal) fractional VNEP solutions. Accordingly, the runtimes stated in Table 7.1 are obtained by the runtimes stated in Theorems 6.20, 6.39, 6.40, and 6.41 as follows:

- The respective sums over the set of requests are substituted by considering only the largest summand and bounding the remaining summands accordingly, thereby introducing the factor $|\mathcal{R}|$.
- The number of substrate edges $|E_S|$ is bounded by $|V_S|^2$ while both the maximal number of virtual nodes and maximal virtual edges is bounded by κ_{max}.
- For the runtime of the approximations in the setting $\langle \mathbf{VE} \,|\, \mathbf{NRL} \rangle$, the bound $\mathcal{O}(n^4 \cdot \varepsilon_{\mathrm{L}}^{-1})$ is used for bounding the runtime of computing approximate LCSP solutions for $\varepsilon_{\mathrm{L}} \in o(1)$, as $\mathsf{time}_{\mathrm{LCSP}}(\varepsilon_{\mathrm{L}}) = \mathcal{O}\big(|E_S| \cdot |V_S| \cdot (\log\log|V_S| + 1/\varepsilon_{\mathrm{L}})\big)$ holds according to Theorem 6.35. Hence, the factor $\kappa_{\mathsf{max}} \cdot n^{2\cdot\kappa_{\mathsf{tw}}+2} + \mathsf{time}_{\mathrm{LCSP}}(\varepsilon_{\mathrm{L}})$ can be upper bounded by $\kappa_{\mathsf{max}} \cdot n^{2\cdot\kappa_{\mathsf{tw}}+2} \cdot \varepsilon_{\mathrm{L}}^{-1}$ under the reasonable assumption that $\kappa_{\mathsf{tw}} \geq 1$ holds.

With respect to the maximal resource augmentation, i.e., $\max\{\beta, \gamma\}$, we note the following. Firstly, the functions introduced in Table 7.2 are obtained from the functions Λ_{cost} (cf. Corollary 7.6) and $\Lambda^{\mathsf{scal}}_{\mathsf{prof}}$ (cf. Corollary 7.10) by considering the lower bounds stated in Equations 7.26 and 7.27. For example, considering the cost VNEP in the setting $\langle \mathbf{VE} \,|\, \mathbf{NR} \rangle$, the resource augmentation bounds $\beta = \bar{\Lambda}_{\mathsf{cost}}(n, \kappa_V, \delta_V)$ and $\gamma = \bar{\Lambda}_{\mathsf{cost}}(n, \kappa_E, \delta_E)$ are obtained from Equation 7.26 and Theorem 7.20 for $\Delta = 2$ by the observation that the function Λ_{cost} is monotonically decreasing in $\hat{\mu}$ (cf. Lemma 7.5). Using the lower bound stated in Equation 7.27, the maximal resource augmentation factor for the node and edge resources is bounded by $\bar{\Lambda}_{\mathsf{cost}}(n, \kappa_{\mathsf{max}}, \delta_{\max})$. Using the same arguments, the resource augmentation bounds for the cost VNEP in the setting $\langle \mathbf{VE} \,|\, \mathbf{NRL} \rangle$ and for the profit VNEP are obtained from Theorems 7.20 and 7.21, respectively.

Lastly, we discuss some peculiarities of the functions of Table 7.2. Firstly, the different constant factors of 6 and 12 in the first and second cases of $\bar{\Lambda}_{\mathsf{cost}}$ and $\bar{\Lambda}^{\mathsf{L}}_{\mathsf{cost}}$ arise as the respective first cases contain the factor $\Delta \cdot \hat{\mu}^{-1}$, such that the factor Δ is canceled out for $\hat{\mu} = \Delta/(\kappa_{\mathsf{max}} \cdot \delta_{\max})$. Secondly, by the same observation, the scaling factor $\varepsilon_{\mathsf{scal}}$ is canceled out for the first case of the function $\bar{\Lambda}^{\mathsf{scal}}_{\mathsf{prof}}$. Furthermore, we note that the functions $\bar{\Lambda}_{\mathsf{cost}}$, $\bar{\Lambda}^{\mathsf{L}}_{\mathsf{cost}}$, and $\bar{\Lambda}^{\mathsf{scal}}_{\mathsf{prof}}$ are monotonically increasing in κ_{max}, n, ε_{L}, and $\varepsilon_{\mathsf{scal}}$, while being monotonically decreasing in $\delta_{\max}$. ■

While the above set of results gives specific bounds on the maximal resource augmentations, in the following some general bounds are presented for different ranges of the value $\kappa_{\mathsf{max}} \cdot \delta_{\max}$.

Corollary 7.23. Assume that $\varepsilon_{\mathrm{L}} \in O(1)$ and $n = |V_S| \geq 3$. For any of the VNEP approximations the following holds.

(1) The maximal resource augmentation $\max\{\beta, \gamma\}$ is bounded by $\mathcal{O}(\delta_{\max} \cdot \kappa_{\mathsf{max}} \cdot \ln n / \ln\ln n)$.

(2) The maximal augmentation $\max\{\beta, \gamma\}$ lies in $\mathcal{O}(\ln n / \ln\ln n)$ for $(\kappa_{\mathsf{max}} \cdot \delta_{\max}) \in \mathcal{O}(1)$.

(3) The maximal augmentation $\max\{\beta, \gamma\}$ lies in $\mathcal{O}(1)$ for $(\kappa_{\mathsf{max}} \cdot \delta_{\max}) \in \mathcal{O}\left(\ln\ln n / \ln n\right)$.

Proof. Considering the first statement, we note that the resource augmentation functions are monotonically increasing in δ. Hence, setting $\delta_{\max}$ to the maximum value of 1, the result is obtained by considering the respective maximal augmentation factors.

For the second and third statements consider the following. Clearly, the resource augmentation functions $\bar{\Lambda}_{\mathsf{cost}}$, $\bar{\Lambda}^{\mathsf{L}}_{\mathsf{cost}}$, and $\bar{\Lambda}^{\mathsf{scal}}_{\mathsf{prof}}$ are monotonically increasing when increasing the product $\kappa \cdot \delta$. Hence, both results follow from plugging in the respective maximal values $\kappa_{\mathsf{max}} \cdot \delta_{\max} \leq c$ (for the second statement) and $\kappa_{\mathsf{max}} \cdot \delta_{\max} \leq c \cdot \ln\ln n / \ln n$ (for the third) for some sufficiently large value c. ■

Using the $\mathcal{O}$-notation, the above corollary can be considered to yield coarse-grained rules of thumb for three different cases, namely when the product $\kappa_{\mathsf{max}} \cdot \delta_{\max}$ is not bounded, when $\kappa_{\mathsf{max}} \cdot \delta_{\max}$ is a constant, and when $\kappa_{\mathsf{max}} \cdot \delta_{\max}$ is small compared to $\ln \ln n / \ln n$.

These resource augmentation factors were all obtained by the application of Chernoff bounds. However, one can also employ the Hoeffding bound to obtain similar results as was done in the publications [RS18a; RDS19; RS19b]. We now state the main results similar to Corollary 7.23, while referring the reader to the Appendix B for the analysis.

Corollary 7.24 (Augmentation Factors using the Hoeffding Inequality)**.** Assume that $\varepsilon_{\mathrm{L}} \in O(1)$ and $n = |V_S| \geq 3$. By employing Hoeffding bounds instead of Chernoff bounds to analyze the maximal resource augmentations, the following holds for any of the VNEP approximations.

(1) The maximal resource augmentation $\max\{\beta, \gamma\}$ is bounded by $\mathcal{O}(\delta_{\max} \cdot \kappa_{\mathsf{max}} \cdot \sqrt{|\mathcal{R}| \cdot \ln n})$.

(2) The maximal augmentation $\max\{\beta, \gamma\}$ lies in $\mathcal{O}(\sqrt{|\mathcal{R}| \cdot \ln n})$ for $(\kappa_{\mathsf{max}} \cdot \delta_{\max}) \in \mathcal{O}(1)$.

(3) The maximal augmentation $\max\{\beta, \gamma\}$ lies in $\mathcal{O}(\sqrt{|\mathcal{R}|})$ for $(\kappa_{\mathsf{max}} \cdot \delta_{\max}) \in \mathcal{O}\left(\ln \ln n / \ln n\right)$.

Proof. Observing that $\sqrt{\ln n} \in \mathcal{O}(\ln n / \ln \ln n)$ holds, the results follow from Theorem B.4. ■

Comparing these results obtained using the Hoeffding bound with the ones of the Chernoff bound stated in Corollary 7.23, we note that the additional factor $\sqrt{|\mathcal{R}|}$ is introduced while the term $\ln n / \ln \ln n$ is exchanged with $\sqrt{\ln n}$. As $\ln n / \ln \ln n$ grows only negligibly faster than $\sqrt{\ln n}$ and the ratio of these terms is less than 2 for values of n up to 10^{32}, the impact of these terms can be considered negligible for practical applications. Hence, the additional factor $\mathcal{O}(\sqrt{|\mathcal{R}|})$ is dominating in the comparison of both types of bounds. Nevertheless, it must be noted that the respective Hoeffding bounds include smaller constant terms (cf. Theorem B.4). Hence, for practical applications which use only a small number of requests, employing the Hoeffding bounds may be beneficial as they yield lower resource augmentation factors for up to 200 requests. For more details on the comparison of these bounds, the reader is referred to the discussion in Appendix B.2.

Besides the application of Hoeffding bounds, we will now conclude the discussion of approximation results by studying how resource augmentations for the profit VNEP can be reduced by employing the scaling factor $\varepsilon_{\mathsf{scal}}$. Interestingly, our below results show that the factor $\varepsilon_{\mathsf{scal}}$ can only be used to reduce resource augmentations when $\varepsilon_{\mathsf{scal}} / (\kappa_{\mathsf{max}} \cdot \delta_{\max}) > \ln n / \ln \ln n$ holds. Accordingly, as in this setting the resource augmentations are already constant (cf. Equation 7.30), in the following we turn our attention on how to obtain profit approximations *without* resource augmentations.

Theorem 7.25. Assume that $n = |V_S| \geq 3$ holds. When $\kappa_{\mathsf{max}} \cdot \delta_{\max} \cdot \ln n \leq 1/24$ holds, there exist deterministic XP-time approximations for the offline profit VNEP which do not violate any resource capacities. Specifically, letting $\epsilon^{\mathsf{opt}} = 3 \cdot \kappa_{\mathsf{max}} \cdot \delta_{\max} \cdot \ln n$ and $\varepsilon^{\mathsf{opt}}_{\mathsf{scal}} = \left(\sqrt{\epsilon^{\mathsf{opt}} + 1} - \sqrt{\epsilon^{\mathsf{opt}}}\right)^2 \geq 1/2$, by scaling the fractional solutions by the factor $\varepsilon^{\mathsf{opt}}_{\mathsf{scal}}$ the objective approximation factors shown in Table 7.4 are obtained, while the maximal augmentation factor is bounded by 1, i.e., $\max\{\beta, \gamma\} \leq 1$ holds.

Setting	$\boldsymbol{\alpha}$	**lower bound on** $\boldsymbol{\alpha}$
$\langle$ **VE** $\mid$ **NR** $\rangle$	$\varepsilon^{\mathsf{opt}}_{\mathsf{scal}} \cdot \left(1 - \frac{2}{3}\sqrt{\frac{b_{\mathsf{frac}}}{b_{\mathsf{max}}}}\right)$	$1/6$
$\langle$ **VE** $\mid$ **NRL** $\rangle$	$(1+\varepsilon_{\mathrm{L}})^{-1} \cdot \varepsilon^{\mathsf{opt}}_{\mathsf{scal}} \cdot \left(1 - \frac{2}{3}\sqrt{\frac{b_{\mathsf{frac}}}{b_{\mathsf{max}}}}\right)$	$(1+\varepsilon_{\mathrm{L}})^{-3/2} \cdot 1/6$

Table 7.4: Summary of deterministic approximation results for the profit VNEP under the small demand assumption using scaling.

Hence, assuming additionally that $\varepsilon_{\mathrm{L}} \in \mathcal{O}(1)$ holds when considering latencies, the approximations not only do not violate resource capacities but also always achieve a constant factor of the optimal profit, i.e., $\alpha \in \Omega(1)$ holds. Furthermore, we note that as ϵ^{opt} approaches zero, $\varepsilon^{\mathsf{opt}}_{\mathsf{scal}}$ approaches 1, i.e., the scaling factor becomes arbitrarily small.

Proof. We prove the theorem in the following three steps. We firstly show that by the choice of $\varepsilon^{\mathsf{opt}}_{\mathsf{scal}}$ the maximal resource augmentation is indeed upper bounded by 1, when $\varepsilon^{\mathsf{opt}}_{\mathsf{scal}}/(\kappa_{\mathsf{max}} \cdot \delta_{\max}) \geq 12 \ln n$ holds. In the next step, we prove that under the assumption $\kappa_{\mathsf{max}} \cdot \delta_{\max} \cdot \ln n \leq 1/24$ always $\varepsilon^{\mathsf{opt}}_{\mathsf{scal}}/(\kappa_{\mathsf{max}} \cdot \delta_{\max}) \geq 12 \ln n$ holds. In the last step, we argue that $\varepsilon^{\mathsf{opt}}_{\mathsf{scal}} \geq 1/2$ must always hold under the given assumptions.

Assuming that $\varepsilon^{\mathsf{opt}}_{\mathsf{scal}}/(\kappa_{\mathsf{max}} \cdot \delta_{\max}) \geq 12 \ln n$ holds, we prove that $\bar{\Lambda}^{\mathsf{scal}}_{\mathsf{prof}}(n, \kappa_{\mathsf{max}}, \delta_{\max}, \varepsilon^{\mathsf{opt}}_{\mathsf{scal}}) \leq 1$ holds, i.e., the maximal resource augmentation factor is bounded by 1 (cf. Theorem 7.22). To see this, we note that under the given assumptions always the third branch of the function $\bar{\Lambda}^{\mathsf{scal}}_{\mathsf{prof}}$ is chosen. To show $\bar{\Lambda}^{\mathsf{scal}}_{\mathsf{prof}}(n, \kappa_{\mathsf{max}}, \delta_{\max}, \varepsilon^{\mathsf{opt}}_{\mathsf{scal}}) \leq 1$, we simply evaluate $\bar{\Lambda}^{\mathsf{scal}}_{\mathsf{prof}}(n, \kappa_{\mathsf{max}}, \delta_{\max}, \varepsilon^{\mathsf{opt}}_{\mathsf{scal}})$:

$$
\begin{aligned}
&\bar{\Lambda}^{\mathsf{scal}}_{\mathsf{prof}}(n, \kappa_{\mathsf{max}}, \delta_{\max}, \varepsilon^{\mathsf{opt}}_{\mathsf{scal}}) \\
&= (1 + \sqrt{4 \cdot \epsilon^{\mathsf{opt}}/\varepsilon^{\mathsf{opt}}_{\mathsf{scal}}}) \cdot \varepsilon^{\mathsf{opt}}_{\mathsf{scal}} && \text{(by definition of } \bar{\Lambda}^{\mathsf{scal}}_{\mathsf{prof}} \text{ and } \epsilon^{\mathsf{opt}}) \\
&= (1 + \sqrt{4 \cdot \epsilon^{\mathsf{opt}}/\left(\sqrt{\epsilon^{\mathsf{opt}}+1} - \sqrt{\epsilon^{\mathsf{opt}}}\right)^2} \cdot \left(\sqrt{\epsilon^{\mathsf{opt}}+1} - \sqrt{\epsilon^{\mathsf{opt}}}\right)^2 && \text{(by definition of } \varepsilon^{\mathsf{opt}}_{\mathsf{scal}}) \\
&= \left(\sqrt{\epsilon^{\mathsf{opt}}+1} - \sqrt{\epsilon^{\mathsf{opt}}}\right)^2 + \sqrt{4 \cdot \epsilon^{\mathsf{opt}} \cdot \left(\sqrt{\epsilon^{\mathsf{opt}}+1} - \sqrt{\epsilon^{\mathsf{opt}}}\right)^2} && \text{(associativity)} \\
&= \left(\sqrt{\epsilon^{\mathsf{opt}}+1} - \sqrt{\epsilon^{\mathsf{opt}}}\right)^2 + 2 \cdot \sqrt{\epsilon^{\mathsf{opt}}} \cdot (\sqrt{\epsilon^{\mathsf{opt}}+1} - \sqrt{\epsilon^{\mathsf{opt}}}) = 1 && \text{(factorization of root)}
\end{aligned}
$$

Turning towards the second step, namely that indeed always the third branch of the function $\bar{\Lambda}^{\mathsf{scal}}_{\mathsf{prof}}$ is chosen, we have to prove that $\varepsilon^{\mathsf{opt}}_{\mathsf{scal}}/(\kappa_{\mathsf{max}} \cdot \delta_{\max}) \geq 12 \ln n$ holds. We first transform this inequality as follows:

$$
\begin{aligned}
& \varepsilon^{\mathsf{opt}}_{\mathsf{scal}}/(\kappa_{\mathsf{max}} \cdot \delta_{\max}) \geq 12 \ln n \\
\Leftrightarrow \quad & \varepsilon^{\mathsf{opt}}_{\mathsf{scal}} \geq 4 \cdot \epsilon^{\mathsf{opt}} && \text{(by definition of } \epsilon^{\mathsf{opt}}) \\
\Leftrightarrow \quad & \left(\sqrt{\epsilon^{\mathsf{opt}}+1} - \sqrt{\epsilon^{\mathsf{opt}}}\right)^2 \geq 4 \cdot \epsilon^{\mathsf{opt}} && \text{(by definition of } \varepsilon^{\mathsf{opt}}_{\mathsf{scal}}) \\
\Leftrightarrow \quad & \epsilon^{\mathsf{opt}} + \sqrt{\epsilon^{\mathsf{opt}}+1} \cdot \sqrt{\epsilon^{\mathsf{opt}}} - 1/2 \leq 0 && (7.31)
\end{aligned}
$$

Denoting by $f(\epsilon) = \epsilon + \sqrt{\epsilon+1} \cdot \sqrt{\epsilon} - 1/2$, the following is simple to verify: $f(0) = -1/2$ holds and as $f'(\epsilon) \geq 0$ holds for $\epsilon > 0$, f is monotonically increasing. Furthermore, f is continuous for $\epsilon > 0$ and $f(1/8) = 1/8 + \sqrt{9/64} - 1/2 = 0$ holds. By the above insights on f, Equation 7.31 holds as long as $\epsilon^{\mathsf{opt}} \leq 1/8$ holds. However, as ϵ^{opt} was defined to be $3 \cdot \kappa_{\mathsf{max}} \cdot \delta_{\max} \cdot \ln n$ and we assumed $\kappa_{\mathsf{max}} \cdot \delta_{\max} \cdot \ln n \leq 1/24$ to hold, this holds trivially.

For the last step, namely that $\varepsilon^{\mathsf{opt}}_{\mathsf{scal}} \geq 1/2$ holds as long as $\kappa_{\mathsf{max}} \cdot \delta_{\max} \cdot \ln n \leq 1/24$ holds, we note the following on the function $g(\epsilon) = (\sqrt{\epsilon+1} - \sqrt{\epsilon})^2$: $g(0) = 1$ holds and g is continuous for $\epsilon > 0$. Furthermore, $g'(\epsilon) = -(\sqrt{\epsilon} - \sqrt{\epsilon+1})^2/(\sqrt{\epsilon} \cdot \sqrt{\epsilon+1})$ holds for the derivative and as $g'(\epsilon) < 0$ holds for $\epsilon > 0$, the function g is monotonically decreasing. As $g(1/8) = (\sqrt{9/8} - \sqrt{1/8})^2 = (3/2 \cdot \sqrt{1/2} - 1/2 \cdot \sqrt{1/2})^2 = 1/2$ holds, we have $g(\epsilon) \geq 1/2$ for $\epsilon \in [0, 1/8]$. Now, as $\varepsilon^{\mathsf{opt}}_{\mathsf{scal}} = g(\epsilon^{\mathsf{opt}})$ holds, $\epsilon^{\mathsf{opt}} \leq 1/8$ follows by the assumption that $\kappa_{\mathsf{max}} \cdot \delta_{\max} \cdot \ln n \leq 1/24$ holds. Thus. $\varepsilon^{\mathsf{opt}}_{\mathsf{scal}} \geq 1/2$ is proven.

By the above arguments, $\bar{\Lambda}^{\mathsf{scal}}_{\mathsf{prof}}(n, \kappa_{\mathsf{max}}, \delta_{\max}, \varepsilon^{\mathsf{opt}}_{\mathsf{scal}}) \leq 1$ holds, proving that the maximal resource augmentation factor $\max\{\beta, \gamma\}$ is upper bounded by 1. Furthermore, as $\varepsilon^{\mathsf{opt}}_{\mathsf{scal}} \geq 1/2$ was shown,

the approximation factors for the objective are at most halved, hence still yielding constant factor approximations on the profit, while *not* violating resource capacities. Furthermore, we note that $\lim_{\epsilon^{\mathsf{opt}} \to 0} \varepsilon^{\mathsf{opt}}_{\mathsf{scal}} = \lim_{\epsilon^{\mathsf{opt}} \to 0} g(\epsilon^{\mathsf{opt}}) = 1$ holds by our above analysis. Hence, when ϵ^{opt} becomes arbitrarily small, the scaling factor $\varepsilon^{\mathsf{opt}}_{\mathsf{scal}}$ approaches 1 and becomes negligible. ■

The above theorem has constructively shown that there exist constant factor XP-time approximations for the offline profit VNEP as long as the maximal demand-to-capacity ratio is sufficiently small with respect to the substrate size and the maximal request size, i.e., as long as $\kappa_{\mathsf{max}} \cdot \delta_{\max} \cdot \ln n = \epsilon^{\mathsf{opt}}/3 \leq 1/24$ holds.

7.5 Derived Heuristics for the Offline Profit VNEP

In the following, heuristics based on the presented randomized rounding scheme are presented for the *profit* variant of the offline VNEP. The heuristics are proposed (i) to study the raw performance of the randomized rounding scheme and (ii) to obtain heuristics for the practical setting in which *resource capacities* are not to be exceeded at any cost. Besides these heuristics, we also study the best solution obtainable via randomized rounding while respecting resource capacities.

The first adaptation is to discard the consideration of respective approximation factors α, β, γ and simply returning *the best* solution found within a fixed number of rounding iterations. Concretely, we consider two

Algorithm 7.5: Heuristical Rounding (without LP Recomputations)

$\{\mathcal{D}_r\}_{r \in \mathcal{R}} \leftarrow$ **compute** solution to fractional offline VNEP
for $i \leftarrow 1$ **to** N **do**
  set $\mathsf{sol}_i \leftarrow \emptyset$
  foreach $r \in \mathcal{R}$ **do**
    choose $\hat{m}_r \leftarrow m_r^k$ **with probability** f_r^k
    if $\hat{m}_r \neq \emptyset$ ***and*** $\mathsf{sol}_i \cup \{\hat{m}_r\}$ *is still feasible* **then**
      set $\mathsf{sol}_i \leftarrow \mathsf{sol}_i \cup \{\hat{m}_r\}$ // accept request r
return the solution sol_i maximizing the profit

Algorithm 7.6: Heuristical Rounding with LP Recomputation

compute solution to LP 6.1 (using column generation)
for $i \leftarrow 1$ **to** N **do**
  set $\mathsf{sol}_i \leftarrow \emptyset$ **and** $\mathcal{R}' \leftarrow \mathcal{R}$
  foreach $r \in \mathcal{R}$ **do**
    foreach $r' \in \mathcal{R}'$ **and each** $m_{r'}^k \in \mathcal{M}_{r'}$ **do**
      if $\mathsf{sol}_i \cup \{m_r^k\}$ is infeasible **then set** $f_{r'}^k = 0$
    resolve Linear Program (without column generation)
    choose $\hat{m}_r \leftarrow m_r^k$ **with probability** f_r^k
    if $\hat{m}_r \neq \emptyset$ **then**
      set $\mathsf{sol}_i \leftarrow \mathsf{sol}_i \cup \{\hat{m}_r\}$ **and** $f_r^k = 1$ // accept request r
    else
      set $f_r^k = 0$ **for all** $m_r^k \in \mathcal{M}_r$ // reject request r
    $\mathcal{R}' \leftarrow \mathcal{R} \setminus \{r\}$
return the solution sol_i maximizing the profit

variants and refer to these *vanilla rounding*, as the rounding procedure itself is not changed, i.e., Lines 5-7 of Algorithm 7.2 are executed unchanged. The first variant, $\text{RR}_{\text{MinLoad}}$, returns the solution minimizing resource augmentations (and among those the one of the highest profit is selected). The second variant, $\text{RR}_{\text{MaxProfit}}$, returns the solution maximizing the profit (and among those the one of the least resource augmentations is selected).

Regarding heuristics that obey resource capacities, we first introduce the heuristical rounding presented in Algorithm 7.5. Here, Line 6 of Algorithm 7.2 is adapted in such a way that selected mappings are only accepted, when its incorporation into the solution does not exceed resource capacities. In other words, if a mapping m_r^k is selected for request $r \in \mathcal{R}$ whose addition would exceed any resource capacity, then this mapping is simply discarded and request r is not embedded.

As an improvement over this heuristic and facilitated by the column generation approach discussed in Section 6.4, we propose another rounding heuristic that a priori removes mappings whose addition would lead to resource violations and recomputes the LP before applying the rounding (see Algorithm 7.6). Therefore, the addition of any rounded mapping is *feasible* while also better guiding the rounding process by providing *currently optimal* rounding probabilities. Specifically, in Lines 5 and 6 first all infeasible mappings are 'removed' by setting the respective LP variables to 0. To reflect made rounding decisions in the LP, either the respective mapping variable is set to 1 (Line 10), or selecting any mappings of the rejected request is disabled (Line 12).

To distinguish between Algorithms 7.5 and 7.6, we often times refer to the former as heuristical rounding without LP recomputations and to the latter as heuristical rounding with LP recomputations. Also, note that the heuristical rounding *without* LP recomputations is referred to as $\text{RR}_{\text{Heuristic}}$, while for the rounding *with* LP recomputations no such identifier is used.

Besides the above discussed heuristics, we will also study the best solution obtainable via randomized rounding while respecting resource capacities. Specifically, similar to Jarray et al. [JK15], we propose to use the Multi-Dimensional Knapsack (MDK) Integer Program 7.7, as follows. Given the decomposed LP solution, i.e., a set of precomputed decomposed mappings $\tilde{\mathcal{D}}_r = \{(f_r^k, m_r^k)\}_k$ for each request, the MDK formulation introduces a single binary variable $x_r^k \in \{0,1\}$ per mapping m_r^k of request r while ensuring that at most one of the mappings of a request is selected and that capacities are not violated. As each decomposed mapping m_r^k has a probability of $f_r^k > 0$ to be chosen, the MDK allows to compute the best possible solution that may be attained by rounding the solution while having an *exponential* runtime. Note that compared to the enumerative LP Formulation 6.1, the MDK is restricted to the set of a priori computed potential mappings $\tilde{\mathcal{D}}_r$, while in the LP formulation *all* valid mappings $\mathcal{M}_r$ are considered.

Integer Program 7.7: Optimal Rounding of Solutions – Multi-Dimensional Knapsack

$$\max \sum_{r \in \mathcal{R}, (f_r^k, m_r^k) \in \tilde{\mathcal{D}}_r} b_r \cdot x_r^k \tag{7.32}$$

$$\sum_{(f_r^k, m_r^k) \in \tilde{\mathcal{D}}_r} x_r^k \leq 1 \qquad \forall r \in \mathcal{R} \tag{7.33}$$

$$\sum_{r \in \mathcal{R}, (f_r^k, m_r^k) \in \tilde{\mathcal{D}}_r} x_r^k \cdot A(m_r^k, x) \leq d_S(x) \qquad \forall x \in G_S \tag{7.34}$$

$$x_r^k \in \{0,1\} \qquad \forall (f_r^k, m_r^k) \in \tilde{\mathcal{D}}_r \tag{7.35}$$

7.6 Computational Evaluation

We now complement our formal approximation results with an extensive computational study. Specifically, the performance of the heuristics presented above is evaluated. As we are not aware of any systematic evaluation of the profit maximization in the offline settings, we present synthetic but extensive computational studies. The general methodology followed throughout this section is presented in Section 7.6.1.

In Section 7.6.2 the vanilla rounding heuristics as well as the heuristical rounding heuristic are evaluated based on the Linear Programming Formulation 6.5 for cactus request graphs. To establish a baseline for the experiments and validate the methodology, we also present results of the classic Multi-Commodity Flow Mixed-Integer Program.

In Section 7.6.3 we report on a study based on the column generation approach presented in Section 6.4 to solve the fractional offline VNEP. Specifically, focusing on heuristics that obey resource capacities, the heuristical rounding with and without LP recomputations are evaluated. Here, we also compare the solutions obtained by these randomized rounding heuristic to the solutions of a classic greedy heuristic.

We have implemented all presented algorithms in Python 2.7. To solve the Mixed-Integer Programs and Linear Programs Gurobi 8.1.1 (for Section 7.6.2) and Gurobi 8.0.0 (for Section 7.6.3) were employed. All of our source code and the results presented within this section are publicly available [Döh+19]. All experiments were executed on a server equipped with 4 Intel Xeon E5-4627v3 CPUs running at 2.6 GHz and reported runtimes are wall-clock times.

7.6.1 General Evaluation Methodology

Substrate Graphs. We use five different substrate networks obtained from the Internet Topology Zoo [Kni+11]. Specifically, wide-area network topologies of between 30 and 50 nodes and between 110 and 168 edges are considered. The topologies span single countries, a continent, or the whole world (see Table 7.5). We consider a single node type and set node and edge capacities uniformly to 100.

Mapping Restrictions. To force the virtual networks to span across the whole substrate network, we restrict the mapping of virtual nodes to one quarter of the substrate nodes. Hence, depending on the substrate topology, the number of suitable substrate nodes per virtual node lies between 7 and 12. We do not impose mapping restrictions for edges.

Name	Identifier	Type	Year	$\lvert V_S \rvert$	$\lvert E_S \rvert$
Deutsche Telekom	DT	Global	2010	30	110
NTT	NT	Global	2011	32	126
Geant	GE	Continent	2012	40	122
UUnet	UN	Country	2011	49	168
Surfnet	SN	Country	2010	50	136

Table 7.5: Summary of Substrate Networks Used in the Evaluation

Demand Generation. We control the demand-to-capacity ratio of node and edge resources using a node resource factor NRF and an edge resource factor ERF. The request's demands are drawn from an exponential distribution and afterwards normalized, such that the following holds:

$$\sum_{r\in\mathcal{R}}\sum_{i\in V_r} d_r(i) = \text{NRF}\cdot\sum_{u\in V_S} d_S(u) \tag{7.36}$$

$$\text{ERF}\cdot\sum_{r\in\mathcal{R}}\sum_{(i,j)\in E_r} d_r(i,j) = \sum_{(u,v)\in E_S} d_S(u,v) \tag{7.37}$$

The resource factors can be best understood under the assumption that all requests are embedded. Under this assumption, a resource factor NRF $= 0.6$ implies that the node load – averaged over all substrate nodes – equals exactly 60%. As virtual edges can be mapped on arbitrarily long paths (even of length 0), the edge resource factor should be understood as follows: the ERF equals 'the number of substrate edges that each virtual edge may use'. In particular, a factor ERF $= 0.5$ implies that if *each* virtual edge spans *exactly* 0.5 substrate edges (and all requests are embedded), then the (averaged) edge resource utilization equals exactly 100%. Hence, while increasing the NRF renders node resources more scarce, increasing the ERF reduces edge resource scarcity.

Profit Computation. To correlate the profit of a request with its size, its resource demands, and its mapping restrictions, we compute for each request its minimal embedding costs as follows. The cost $c(u,v)$ of using an edge $(u,v) \in V_S$ is set to be proportional to the geographical distance of its endpoints. The cost of nodes is set uniformly to $c(u) = \sum_{(u,v)\in E_S} c(u,v)/|V_S|$ for all $u \in V_S$. Hence, the total node cost equals the total edge cost. Defining the cost of a mapping m_r to be $\sum_{x\in G_S} A(m_r,x)\cdot d_S(x)$, we compute the minimum cost embedding for each request $r \in \mathcal{R}$ using an adaption of Mixed-Integer Program 6.3 and set b_r to the respective minimum embedding cost.

7.6.2 Validation and Evaluation based on LP for Cactus Requests

In this section, we report on results pertaining to the vanilla rounding heuristics and the heuristical rounding (without LP recomputations) that are obtained by solving the LP Formulation 6.5 for cactus request graphs. Accordingly, for this study, only cactus requests are considered, whose generation procedure is discussed in Section 7.6.2.1.

In Section 7.6.2.3, we report on the baseline results obtained by solving the Multi-Commodity Flow Mixed-Integer Programming Formulation 6.3. Afterwards, we discuss the performance of the rounding heuristics

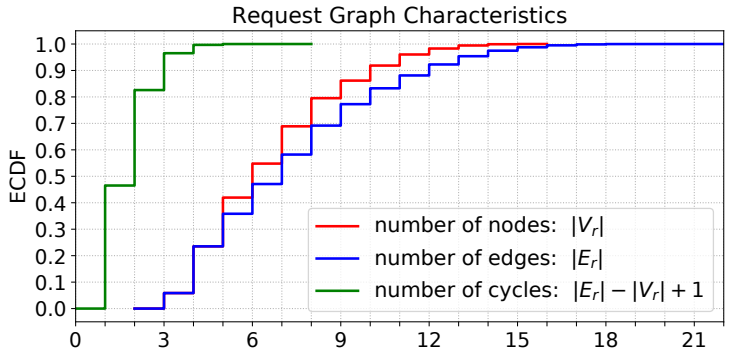


Figure 7.1: Characteristics of the generated cactus requests graphs, namely the number of nodes, edges and number of cycles. The depicted empirical cumulative distribution functions (ECDF) are based upon 100,000 sampled cactus requests graphs.

in Section 7.6.2.4 and of the best roundable MDK solutions in Section 7.6.2.5, and lastly, present in Section 7.6.2.6 an overview on the empirical formulation strength of the cactus LP formulation and the classic MCF formulation. Specifically, as shown in Theorem 6.5, the classic MCF formulation cannot be decomposed and may attain arbitrarily worse bounds on the profit compared to the solution of the fractional offline VNEP.

7.6.2.1 Cactus Request Graph Topology Generation

Cactus graph requests are generated by (i) sampling a random binary tree of maximum depth 3, (ii) adding additional edges randomly as long as they do not refute the cactus property *as long as such edges exist*, and (iii) orienting edges arbitrarily.

Concretely, the sampling process of binary trees works as follows: starting with at the root node, the number of children is drawn using the discrete distribution $\mathsf{Pr}(\#\text{children} = 0) = 0.15$, $\mathsf{Pr}(\#\text{children} = 1) = 0.5$, and $\mathsf{Pr}(\#\text{children} = 2) = 0.35$. For each (newly) generated node (of depth less than 3) further children are generated according to the same distribution. We discard graphs having less than 3 nodes. According to the above generation procedure, the expected number of nodes and edges is 6.54 and 7.28, respectively. On average, 61% of the edges lie on a cycle. Figure 7.1 offers a more in-depth view on the characteristics of the requests.

7.6.2.2 Instance Parameter Space

To generate instances, we consider the cartesian product of the following parameters:

- $|\mathcal{R}| \in \{40, 60, 80, 100\}$,
- NRF $\in \{0.2, 0.4, 0.6, 0.8, 1.0\}$,
- ERF $\in \{0.25,\ 0.5, 1.0, 2.0, 4.0\}$, and
- the 5 different substrate graph topologies as introduced in Table 7.5.

For each combination of these parameters, we generate 15 instances per parameter combination, yielding $7,500$ instances overall.

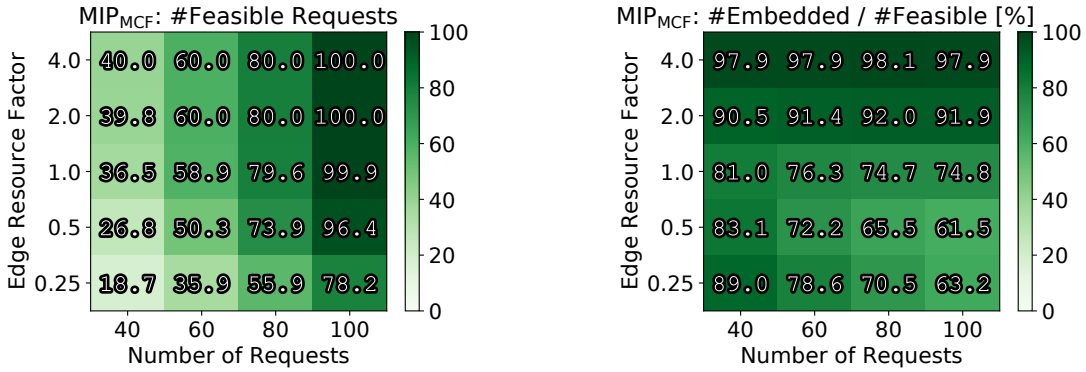


Figure 7.2: Overview of the feasibility of generated requests and the baseline's acceptance ratio. Each cell averages the result of 375 instances.
Left: The feasibility of requests is obtained from (cost-optimally) embedding the requests to compute the profit a priori. Note that absolute numbers are depicted.
Right: The acceptance ratio of the baseline MIP$_{\text{MCF}}$ with respect to the requests that were found to be feasible (infeasible requests are not considered).

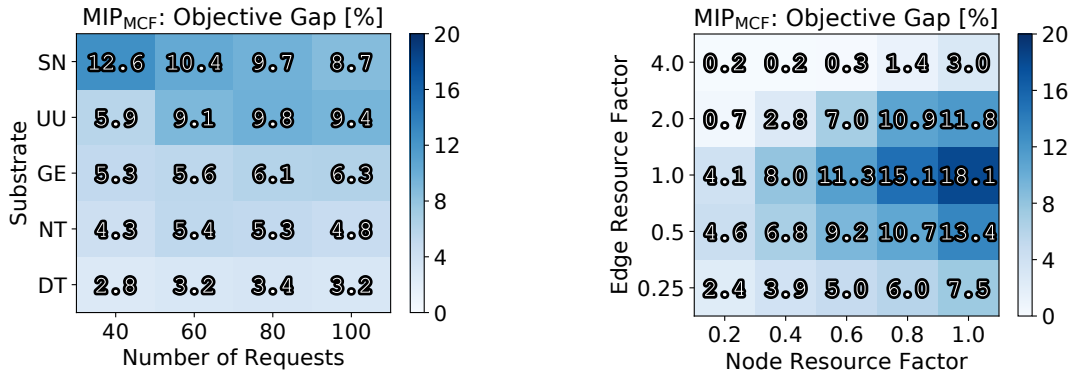


Figure 7.3: Overview of the objective gap achieved by the baseline algorithm MIP$_{\text{MCF}}$ after up to 2 hours of computation time. Note the different axes. The left plot averages the results of 375 and the right one the results of 300 instances per cell. The number of requests, i.e., the problem size, has a less distinct impact on the objective gap than the resource factors alone. Furthermore, the averaged objective gap increases with the substrate topology size.

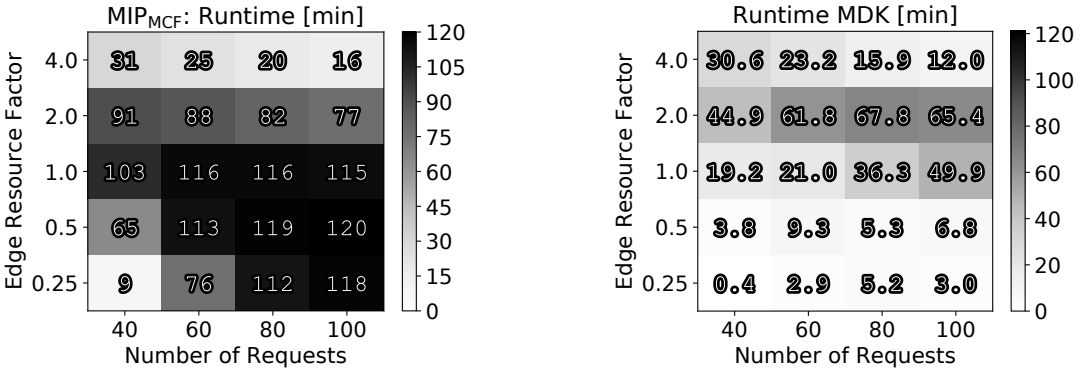


Figure 7.4: Overview of the runtimes of the baseline algorithm MIP$_{\text{MCF}}$ and the MDK formulation. Each cell averages 375 results. Both formulations were solved using Gurobi 8.1.1 and computations were terminated after 120 minutes.

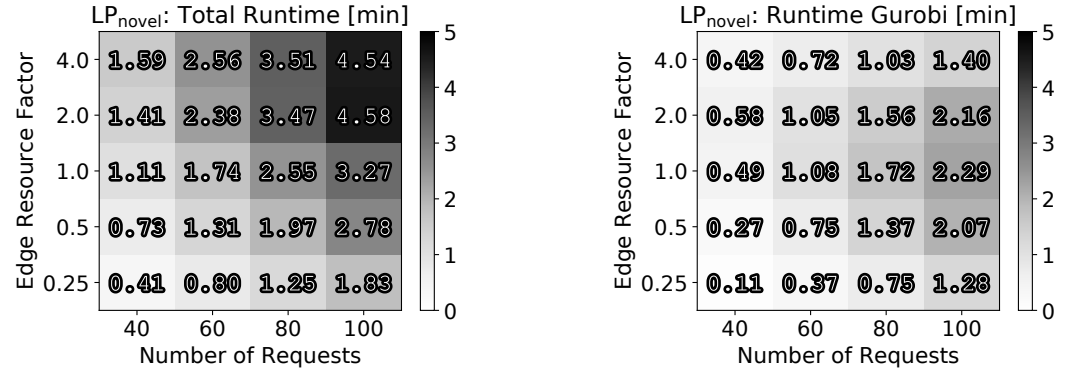


Figure 7.5: Overview of the runtimes of the novel LP Formulation 6.5 Each cell averages 375 results. On the left, the total runtime including the time to construct the LP and the time to decompose the solution is depicted. On the right only the time to solve the LP formulation is depicted.

7.6.2.3 Baseline Performance and Validation

To obtain a near-optimal baseline solution for each of the 7,500 instances, we employ Gurobi 8.1.1 to solve the Mixed-Integer Programming Formulation 6.3 using a single thread. We terminate the computation after 2 hours or when the *objective gap* falls below 1%, i.e. when the constructed solution is *provably* less than 1% off the optimum. On average the runtime per instance is 80.6 minutes (cf. Figure 7.4, left).

Figure 7.2 gives an initial overview of the number of requests for which feasible embeddings exist and the acceptance ratio of the best solution as a function of the number of requests and the edge resource factor. The number of feasible requests is determined during the a priori profit computation and may (on average) lie below 50% when edge resources are very scarce (ERF = 0.25) but otherwise consistently lies above 75%. Similarly, the acceptance ratio of the baseline solution highly depends on the edge resource factor, ranging from close to 62% to roughly 98% (on average).

Figure 7.3 depicts quality guarantees for the baseline solutions obtained during the solution process of IP Formulation 6.3. Concretely, the formulation is solved using Gurobi's branch-and-bound implementation, consistently yielding upper bounds on the attainable profit. Accordingly, the objective gap depicted in Figure 7.3 gives guarantees on how far the found solutions are at most off optimality (on average). While increasing the number of requests does not increase the objective gap per se, both the node and edge resource factors have a distinct impact. Particularly, for the *maximal* node resource factor of 1.0 and a

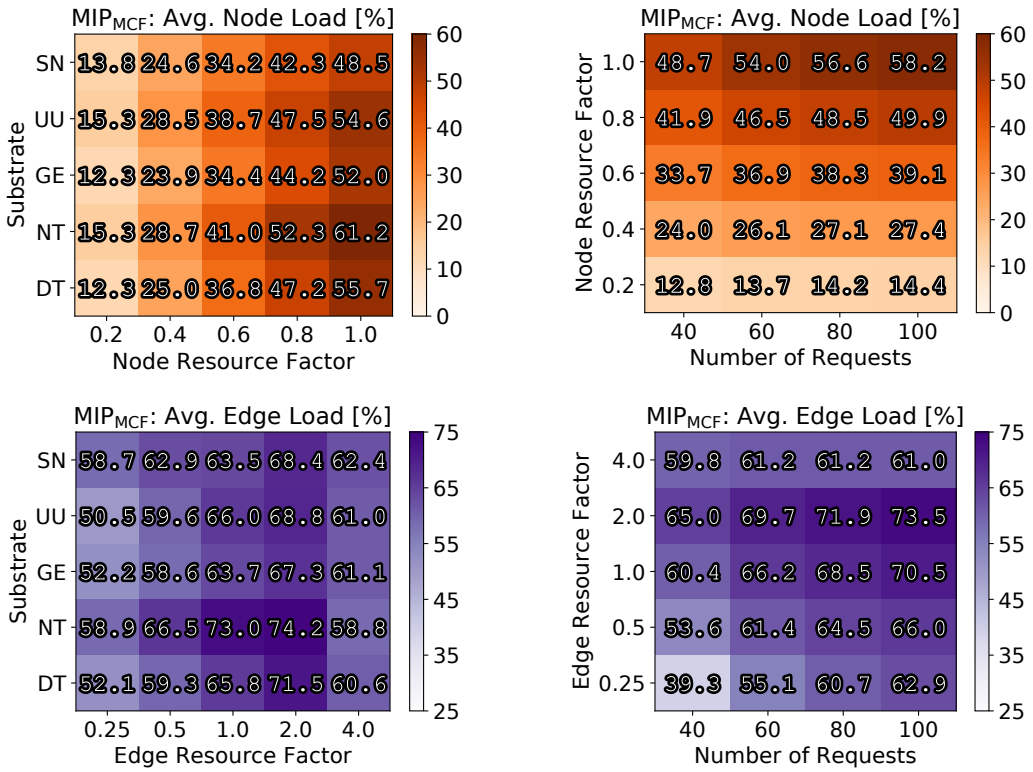


Figure 7.6: Overview of the resource loads of the solutions computed by the baseline algorithm MIP$_{\text{MCF}}$. Each cell averages the results of 300 or 375 solutions, respectively. Depicted are the averaged node/edge loads as a function of the node/edge resource factor and the number of requests and the substrate topology. While the node resource factor has a distinct impact on the average node load, the average edge load lie consistently between 50% and to 75%, allowing the conclusion that edge resources often represent the bottleneck resource. Furthermore, the substrate topology has only a minor impact.

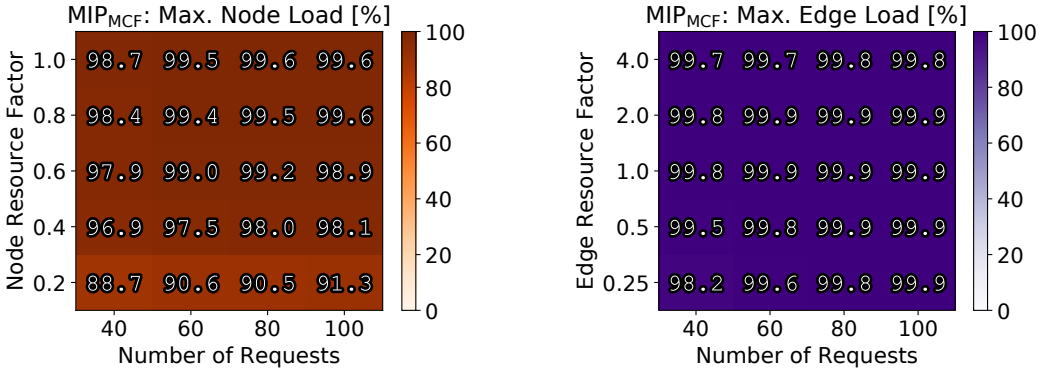


Figure 7.7: Overview of the maximal resource loads of the solutions computed by the baseline algorithm MIP_{MCF}. Each cell averages the results of 375 solutions. Depicted are the maximum node/edge loads as a function of the node/edge resource factor and the number of requests. Clearly, except for a node resource factor of 0.2, the maximum loads are always close to 100%.

medium edge resource factor of 1.0, the averaged objective gap lies slightly above 18%. The mean objective gap value is 6.6% and the maximum observed gap across all 7,500 instances is roughly 62%. Notably, the average objective gap increases with the substrate topology size (cf. Table 7.5).

Figures 7.6 and 7.7 validate the impact the resource factors have on the respective resource loads. Depicted are the averaged and maximal node and edge loads as a function of the respective resource factors, the number of requests, and the substrate topology. As can be clearly seen, increasing the node resource factor increases the average node loads, while the (average) maximum node resource load comes mostly close to 100%. For the edge resource loads, the picture is a different one. In particular, the average edge load consistently lies significantly above the average node load. Also, the maximum edge load approaches nearly 100% in all the cases. Overall, we conclude that the edge resource factor is the key limiting factor, as the utilization of edges is significantly higher. This observation is also supported by the impact the edge resource factor has on the acceptance ratios (cf. Figure 7.2).

7.6.2.4 Performance of Randomized Rounding Heuristics

To apply the rounding algorithms presented in Section 7.5, we solve the LP Formulation 6.5 for cactus requests by employing Gurobi 8.1.1, specifically its Barrier algorithm. Figure 7.5 (left) depicts the average runtime to solve the LP, including the time to construct the (potentially very large) LP. The latter is not negligible as the formulation contains up to 1,000k variables for some instances (cf. Figure 7.5, right). The runtime increases from around 1 minute for $|\mathcal{R}| = 40$ to around 3.5 minutes for $|\mathcal{R}| = 100$. The maximally observed runtime in our experiments amounted to roughly 15 minutes.

With respect to the results of our rounding heuristics, we first discuss the results of our vanilla rounding heuristics $\text{RR}_{\text{MinLoad}}$ and $\text{RR}_{\text{MaxProfit}}$. Concretely, we report on the best solution found within 1,000 rounding iterations. Figure 7.10 depicts the respective results as a scatter plot while Figure 7.9 depicts empirical cumulative distribution functions (ECDF) for both the achieved profit and the maximal resource loads. As can be seen, for $\text{RR}_{\text{MinLoad}}$ the algorithm achieves a profit between 50% and 140% compared to the best solution constructed by the MIP, while exceeding resource capacities mostly by 25% to 125% of the resource's capacity. For $\text{RR}_{\text{MaxProfit}}$, the achieved profit always exceeds the baseline's profit. This is to be expected based on our analysis in Section 7.2.1, as the expected benefit is at least as large as the optimal Mixed-Integer Programming formulation's value. The resource loads mostly lie below 500% with the maximum being 871%.

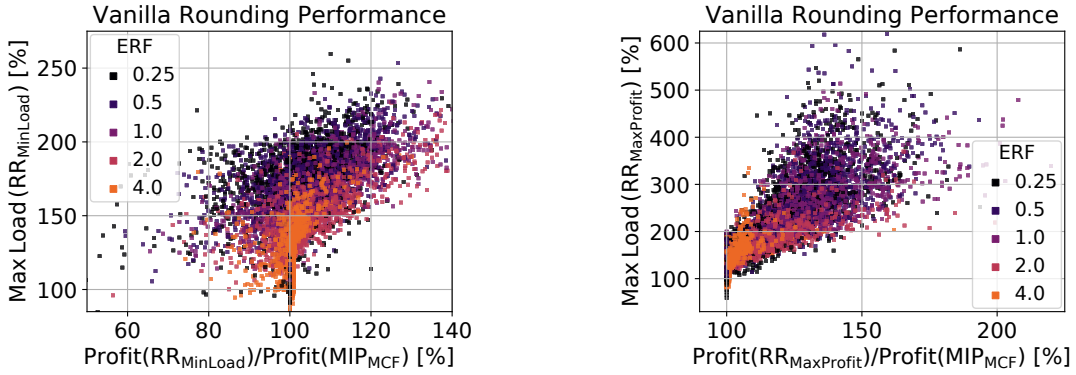


Figure 7.8: Depicted is the performance of the solutions obtained via the two *vanilla* rounding schemes $\mathrm{RR}_{\mathrm{MinLoad}}$ (left) and $\mathrm{RR}_{\mathrm{MaxProfit}}$ (right). Each point corresponds to a single instance and is colored according to the instance's edge resource factor. The left and right plots shows results for 7,458 and 7,499 of the 7,500 requests, respectively; the other results lie outside the depicted area.

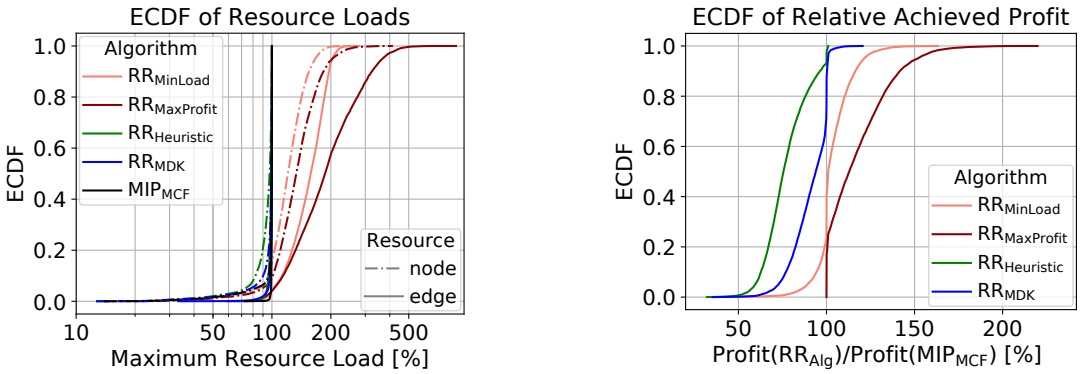


Figure 7.9: Comparison of the different algorithms in terms of (maximal) resource usage and the relative achieved profit for all 7,500 instances.

For both selection criteria, the edge resource factor has a distinct impact on the (overall) maximum load. This can be explained as follows. As each request edge may use any of the substrate edges (compared to the restricted node mappings), the chances of contentions, i.e., multiple request edges being mapped on the same substrate edge, is higher. Additionally, the contention for edge resources is increased by the fact that the number of virtual edges is (always) at least as large as the number of edges (cf. Figure 7.1) and that a single request edge may use *multiple* substrate edges. This observation is further substantiated by the ECDF presented in Figure 7.9: the maximal edge loads are consistently larger than the maximal node resource loads for all randomized rounding algorithms.

The results of the heuristical rounding, which does not exceed resource capacities, are presented in Figure 7.10 (heatmaps) and Figure 7.9 (ECDFs). Again, 1,000 rounding iterations were considered. While for low edge resource factors, i.e., scarce edge resources, the solutions achieve around 70% of the profit of the MIP baseline, for larger edge resource factors, the relative performance exceeds 83%. Also, the performance improves when increasing the number of requests. This can be explained as follows. For any combination of resource factors, the virtual resource demands are computed according to Equations 7.36 and 7.37 *independently* of how many requests are considered. Hence, when increasing the number of requests, the resource demands of each single request become smaller. Hence, the maximal demand-to-capacity ratio $\delta_{\max}$ decreases for *all* substrate resources $x \in G_S$ when increasing the number of requests. Accordingly, expected resource augmentations decrease (cf. Theorem 7.22). Thus, the heuristical rounding algorithm is able to more easily find solutions of high profit *not* augmenting resources.

Overall, the average relative performance with respect to the baseline solutions is 77.2%, with the minimal one being 32.1%. Furthermore, less than 2% of constructed solutions achieve less than 50% of the baseline's profit (cf. Figure 7.9).

7.6.2.5 Optimal Rounding Solution $\mathrm{RR}_{\mathrm{MDK}}$

We lastly discuss the results of executing the Multi-Dimensional Knapsack (MDK) Integer Program 7.7. As for the baseline IP, solutions were computed using Gurobi 8.1.1 and computations were terminated when an objective gap of less than 1% was reached or when Gurobi's runtime exceeded 2 hours. As shown in Figure 7.11, the optimal rounding solution improves upon the heuristical rounding solutions significantly: the average relative profit is 91.2%, yielding an average improvement over the heuristical rounding by 14%. Interestingly, for the maximal edge resource factors and 80 or 100 requests, the solutions found by the MDK slightly improve upon the baseline solutions (on average), with the maximal improvement over the baseline being 20.6% (cf. Figure 7.9). Since the runtime of the MDK (cf. Figure 7.4) consistently and at times significantly lies beneath the runtime for constructing the baseline solutions, the MDK may pose an interesting alternative to computing solutions using the MCF IP formulation.

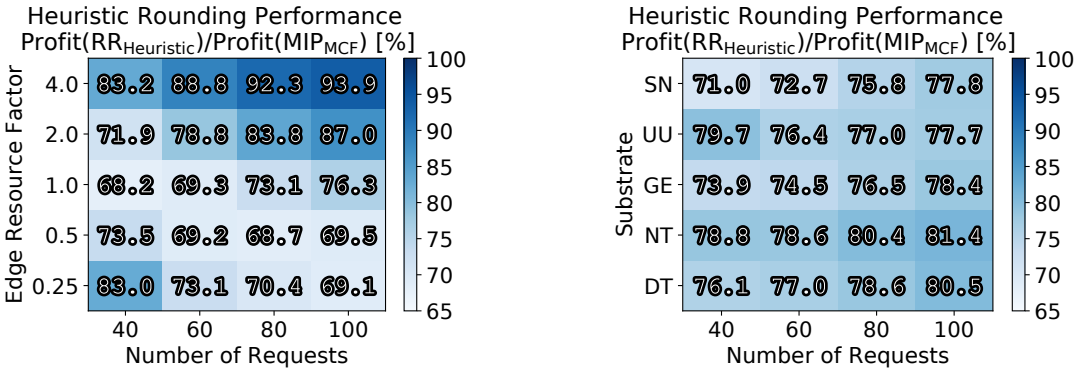


Figure 7.10: Overview of the averaged relative performance achieved by the heuristic rounding algorithm as a function of the edge resource factor and the number of requests (left) and the substrate topology (right). Each cell averages the result of 375 results.

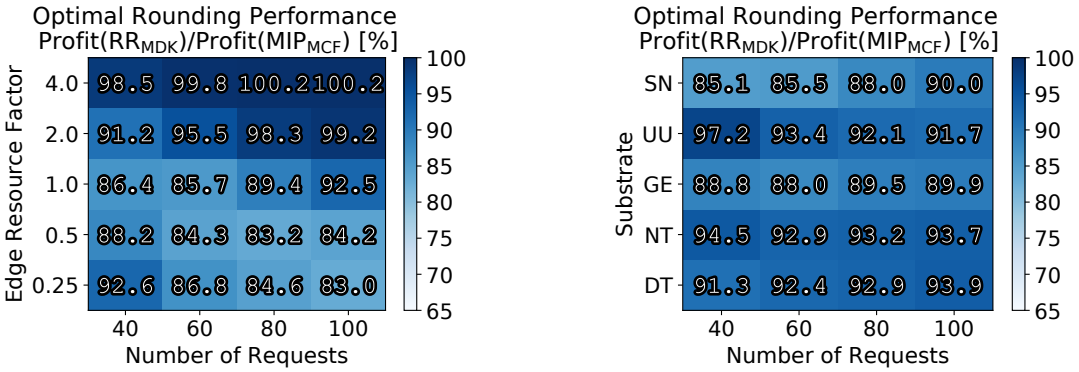


Figure 7.11: Overview of the averaged relative performance achieved by the MDK solutions as a function of the edge resource factor and the number of requests (left) and the substrate topology (right). Each cell averages the result of 375 results.

7.6.2.6 Comparison of Formulation Strengths

Lastly, we empirically study the strength of our novel LP Formulation 6.5 and compare it with the classic MCF Formulation 6.3. Concretely, as proven in Theorems 6.5, the classic MCF formulation has an unbounded or very large integrality gap in general, i.e., the bound on the profit returned by the solution can be arbitrarily far off the optimal attainable profit. Our novel LP formulation is provably stronger than the old formulation, as it only allows for (fractional) solutions, which can be decomposed into valid mappings. Thus, it will always yield (equal or) better bounds on the attainable profit.

Figure 7.12 presents the experimental comparison of both formulations. In particular, for each of the 7,5000 instances we compare the objective of the novel *Linear Programming* Formulation 6.5 to *two* bounds computed during the solution process of the baseline MIP_{MCF}: the initial bound, i.e., the objective of the *Linear Programming* Formulation 6.3, and the final (best) bound computed during the solution process of the (Mixed-)*Integer* Program.

As can be seen, the initial LP bounds of the classic formulation at times exceeds our novel formulation's objective by more than 300%, i.e., the classic formulation 'overestimates' the maximal attainable profit by a factor of more than 3. For roughly 22% of the instances, the novel LP improves the bounds by a factor of at least 1.5. Clearly, the more requests are considered, the less accurate the classic MCF formulation is. Considering the final bound computed during the execution of the (Mixed-)*Integer* Program, we see that these bounds always improve upon the novel LP's bound. However, for 80% of the instances, our LP's bound is only improved by roughly 15% with the maximal improvement being slightly more than 50%.

Concluding, we note that our novel LP formulation is much *stronger* than the classic formulation: it consistently yields significantly better bounds in practice compared to the classic LP formulation and comes close to the bounds obtained by solving the (Mixed-)*Integer* Program for up to 2 hours.

7.6.3 Evaluation of Column Generation Based Heuristics

In this section we present three types of evaluations to validate the treewidth based approach. Firstly, we present a study of the treewidth of random graphs to grasp for which graphs our approach may be reasonable. Secondly, we benchmark the column generation based Linear Programming Formulation 6.1

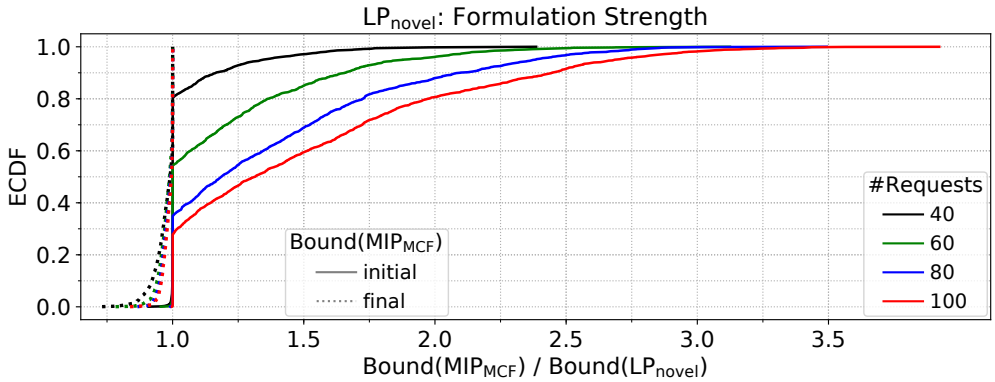


Figure 7.12: Comparison of the bounds on the profit computed by the novel LP Formulation 6.5 and the classic MCF Formulation 6.3. As objective bounds are continuously improved during the solution process of the Mixed-Integer Program MIP_{MCF}, we report on the initial (weakest) bound and the final (best) bound. The initial bound is essentially the objective value of the LP relaxation of Formulation 6.3, but might be improved by the solver Gurobi based on the introduction of cutting planes valid only for the integer variant.

against the LP Formulation 6.5 and show that the column generation approach clearly outperforms the latter formulation. Thirdly, we generate another set of offline VNEP instances according to different treewidths and compare the performance of the randomized rounding heuristics to the performance of the well-known ViNE heuristics (cf. Section 4.2.1).

7.6.3.1 Qualitative and Quantitative Analysis of the Treewidth

We have generated 1,200 undirected graphs with $\{5, \ldots, 45\}$ nodes and edge creation probabilities in the range of $\{0.05, 0.06, \ldots, 0.95\}$, yielding 4.47M graphs overall. We have then run the exact algorithm by Tamaki [Tam17] to compute the optimal treewidth. Our results are presented in Figure 7.13. Notably, the (average) treewidth is less than 6 for most graphs with fewer than 15 nodes and a connection probability of less than 50%. The runtime for computing the tree decompositions of width less than 10 lies vastly below two seconds with a median computation time of only 200ms, enabling the application of our approach in the first place.

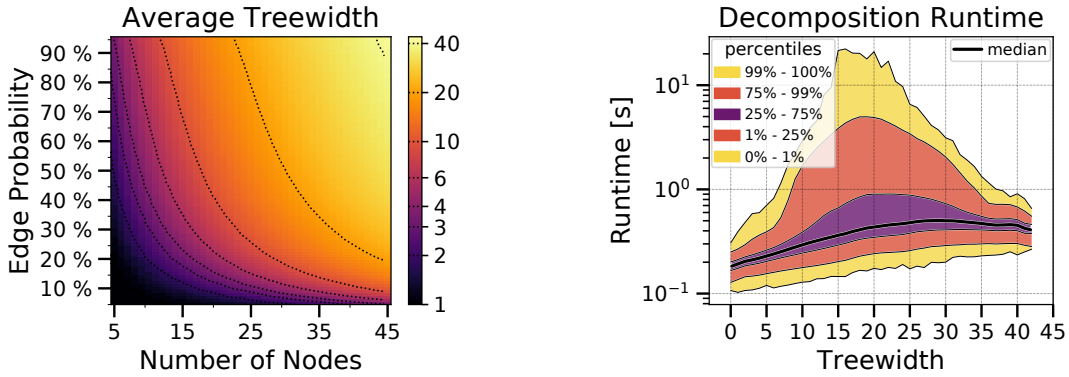


Figure 7.13: Study of the treewidth of random graphs using Tamaki's algorithm [Tam17].
Left: The average treewidth of randomly generated graphs. Depicted are also the contour lines for the values 2, 3, 4, 6, 10, 20, 40.
Right: Runtime of Tamaki's algorithm to compute the optimal tree decompositions.

7.6.3.2 Comparison of LP Runtimes

Our DynVMP implementation employs several optimizations. Most prominently, our implementation is based on small 'semi-nice' tree decompositions (cf. [Bod97]) to enable cost computations using matrix multiplications. Concretely, the used tree decomposition is a small one where for each edge a novel tree node is introduced, such that the newly created node's bag is the intersection of the two previously incident nodes. For more details, we refer the interested reader to our implementation [Döh+19].

As discussed in Section 6.4, the column generation approach allows to compute approximate solutions (cf. Lemma 6.37). In the experiments presented throughout this section, the separation process was stopped once a 1.001-optimal solution was constructed.

We first benchmark the performance of the column generation approach against the Linear Programming Formulation 6.5 for cactus request graphs (treewidth 2) on the instances discussed in Section 7.6.2. Figure 7.14 shows the runtime comparison in terms of speedup over the cactus LP formulation. Depicted are speedup factors when including the time to compute the tree decompositions and without. As for cactus graph requests the tree decompositions can be easily computed based on Lemma 6.6, in the following we mainly discuss the runtimes without the tree decomposition runtimes.

Firstly, note that the column generation approach performs always better, except for ≈ 3% of the scenarios with 40 and 60 requests. The average speedup lies around 6.0 with the speedup increasing with the number of requests. Moreover, for 15% to 20% of the instances the speedup factor lies above 10. Notably, the maximal speedup factor observed is 49.6×. When including the tree decomposition runtimes, the average speedup still lies above 1.5 with the maximum being 7.3. As the original LP runtime comes close to 900 seconds at times (cf. Figure 7.5), the potential runtime reduction is significant.

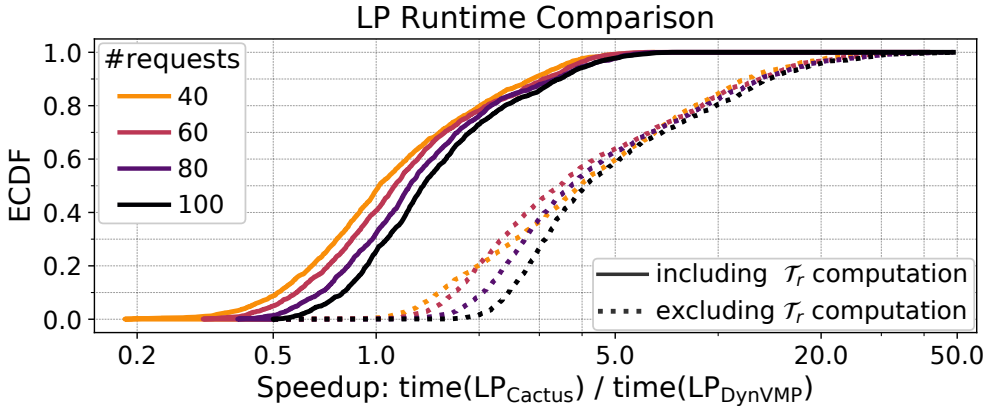


Figure 7.14: ECDF of the runtime speedup when using the novel column generation based LP over the LP Formulation 6.5 for cactus request graphs on the instances studied in Section 7.6.2. The solid lines include the time to compute the tree decompositions, while these are excluded for the dashed lines.

7.6.3.3 Performance of Heuristics

In the following we discuss the performance of the randomized rounding heuristics, obeying capacity constraints, and compare their performance to the well-known ViNE heuristics [CRB12]. To study the scalability of the column generation approach, we generate a novel set of 6,000 instances as follows.

Instance Generation. We generally follow the methodology presented in Section 7.6.1 to create instances. In particular, we consider the same varying number of requests $\{40, 60, 80, 100\}$, the same node and edge resource factors $\{0.2, 0.4, 0.6, 0.8, 1.0\}$ and $\{0.25, 0.5, 1, 2, 4\}$, the same mapping restrictions, as well as the same profit computation method. However, as substrate topology only the the medium-sized GÉANT network is used.

We generate requests, such that all have the same exact treewidth, lying in $\{1, 2, 3, 4\}$. The number of nodes per request is drawn uniformly from $\{5, \ldots, 15\}$. For treewidth 1, i.e., trees, the request graphs are generated randomly by adding edges until the graph is a tree (discarding edges creating cycles). For generating graphs of treewidth $2, 3, 4$, we employ the graphs generated to evaluate the performance of Tamaki's algorithm. To this end we have stored all generated undirected graphs and uniformly at random select graphs of the respective treewidth and number of nodes. As directed requests are considered, edge orientations are chosen uniformly at random.

Studied Algorithms and Implementation Details. We compare the performance of the randomized rounding heuristics and the ViNE heuristics for unsplittable VNEP embeddings (cf. Section 4.2.1 for an discussion). The ViNE algorithms use the Multi-Commodity Flow LP formulation to guide the embedding of single requests: node mappings are performed either randomly or deterministically according to the LP node mapping variables while request edges are embedded using shortest-paths. Two different LP

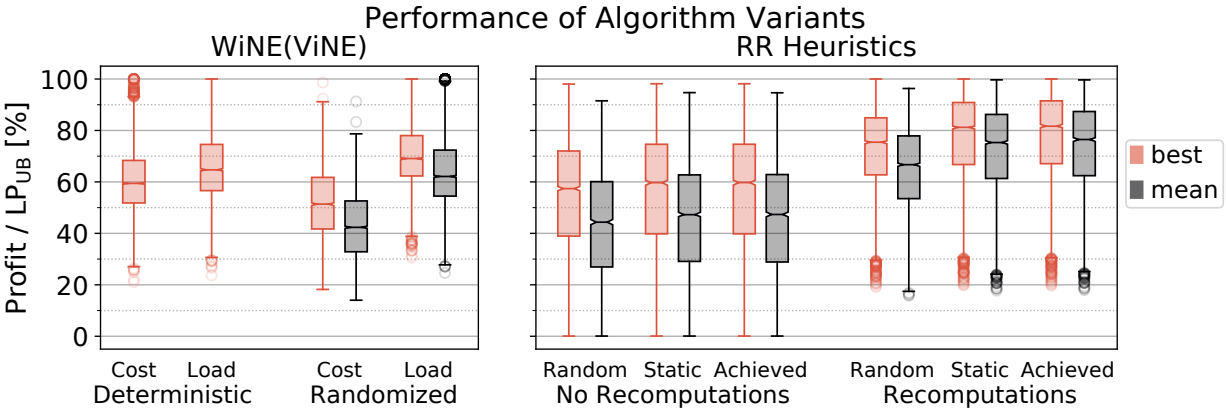


Figure 7.15: Performance of the four different WiNE algorithms the six different randomized rounding heuristics compared to the upper bound ($\mathbf{LP_{UB}}$).

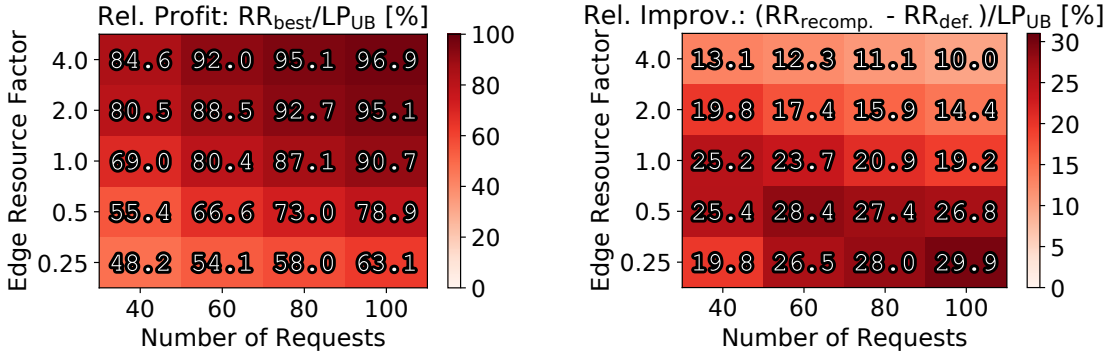


Figure 7.16: Left: Relative profit achieved by the best randomized rounding solution compared to the upper bound $\mathbf{LP_{UB}}$ (a cell averages 300 results). Right: Relative improvement with respect to the upper bound $\mathbf{LP_{UB}}$ that is achieved by the best solution rounded by the heuristic with LP recomputations versus the best solution rounded without LP recomputations.

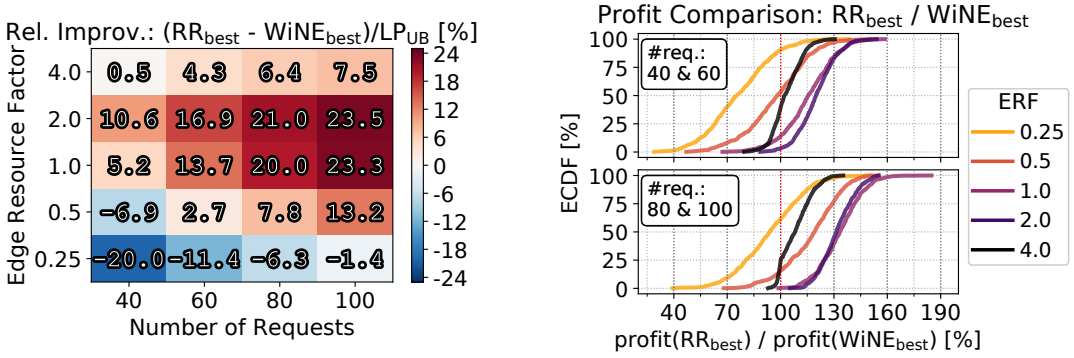


Figure 7.17: Left: Relative improvement of randomized rounding over ViNE (a cell averages 300 results). Right: Direct comparison of the best profits achieved (an ECDF represents 600 results).

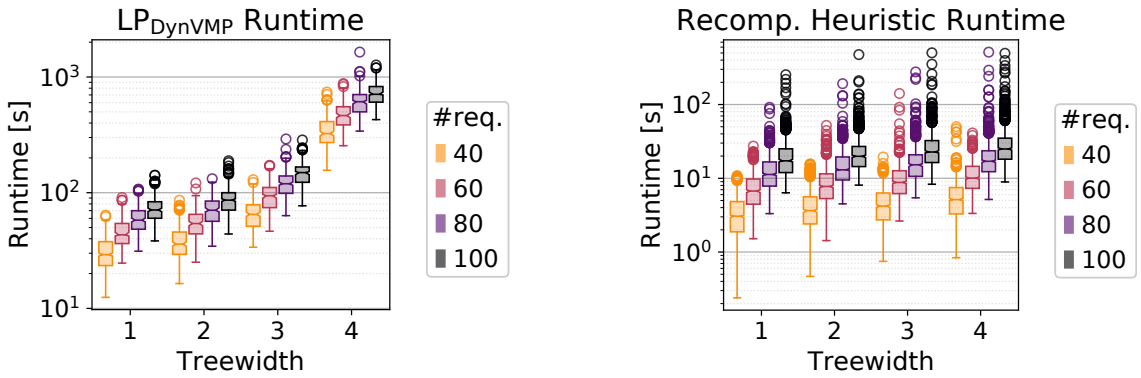


Figure 7.18: Runtime of the novel LP (left) and of the novel rounding heuristic (right) as a function of the treewidth and the number of requests. Note the logarithmic y-axes.

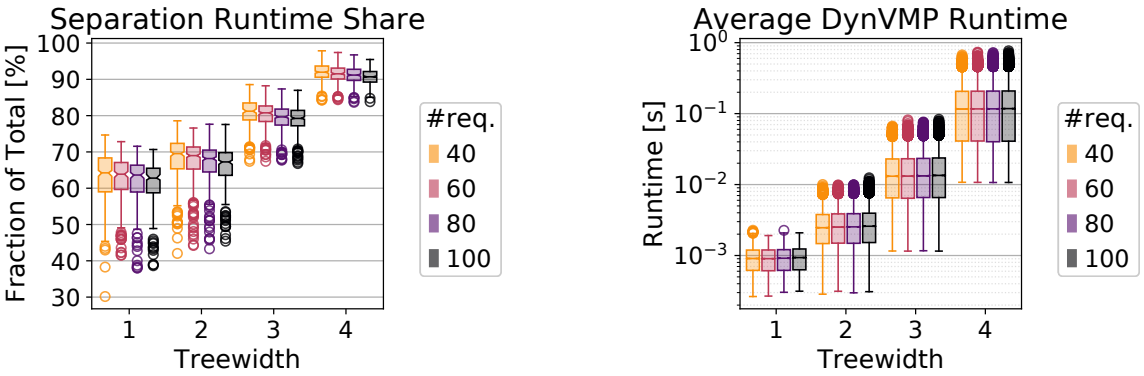


Figure 7.19: Share of the separation process in the total runtime of the LP and average runtime of the DynVMP algorithm to perform the separation (per request) (right).

objectives were proposed in [CRB12]: one minimizing resource usage and another also performing load-balancing. For the offline setting, the authors of [CRB12] have proposed the window-based heuristic (WiNE) that orders requests descendingly according to their profits and greedily embeds each request using ViNE.

Considering the randomized rounding heuristics, we again consider the heuristical rounding algorithm (cf. Algorithm 7.5 but also study the heuristical rounding with LP recomputations (cf. Algorithm 7.6). Furthermore, for this evaluation, we also study the impact of the order in which the requests are processed. In particular, we study the following orders: random (as in Section 7.6.2) and either sorting the requests in descending fashion by their static profits or their actual achieved profit in the LP. Specifically, the actually achieved profit in the LP formulation for request $r \in \mathcal{R}$ is given by $b_r \cdot \sum_k f_r^k$.

Results. We first report on the performance of the different algorithms. In Figure 7.15 the best and mean solution quality relative to the maximum attainable LP profit is depicted. For ViNE, the load-balancing objective outperforms the cost one. Considering the randomized rounding (RR) heuristics, the ones with recomputations significantly outperform the ones without. Ordering the requests according to the profit (static / achieved) yields the best solutions. This is also substantiated in Figure 7.16 (right): the best solution constructed using LP recomputations achieves on average 20.7% more profit than the best solution not applying recomputations.

Considering the performance of the best solution constructed by randomized rounding (cf. Figure 7.16 (left)), we note that on average 77.5% of the upper bound is achieved. This relative profit highly depends both

on the number of requests and the edge resource factor.

Figure 7.17 compares the performance of the randomized rounding and the WiNE heuristics. The mean relative improvement over WiNE again significantly depends on the number of requests and the edge resource factor: when edge resources are very scarce, i.e., for an ERF of 0.25, WiNE performs significantly better, while for ERFs 1.0, 2.0, and 4.0 randomized rounding consistently yields better solutions (86.7% of scenarios). Even more, for 80 and 100 requests and ERFs of 1.0 and 2.0, randomized rounding finds better solutions in 99.9% of the scenarios, improving the best ViNE solution by more than 30% in 57.5% of the scenarios. The performance drop for low ERFs may be due to fewer generated mappings being feasible.

The runtime of the column generation LP lies in the order of 100 seconds for treewidths below 3 and several hundred seconds for treewidth 4 (see Figure 7.18). The runtime of the recomputation heuristics mainly ranges between few seconds and 60 seconds (see Figure 7.18). The separation procedures takes up a substantial amount of the time to compute the LP solutions (see Figure 7.19). For a treewidth of 4, the separation runtime share lies around 90%. Considering the runtime of the DynVMP algorithm per request, we see that the runtime lies in the order of few milliseconds for a treewidth of 1 and is even for a treewidth of 4 upper bounded by 1 second, while the median lies around 100 milliseconds even in that case.

The runtimes of the rounding without recomputations and the ViNE heuristics are not depicted but their average consistently was 0.03s and 6.38s, respectively.

7.7 Summary and Novelty of Contributions

In this chapter the first tri-criteria approximations for the offline VNEP have been presented. All approximations are based on the XP-time algorithms to solve (or approximate) the respective fractional offline VNEP problems. The key to obtain these approximations has been to focus on the construction of valid mappings, which do not enforce feasibility with respect to the capacities, to allow for the computation of convex combinations of valid mappings using the LP formulations presented in Section 6. As a drawback of this approach, the obtained approximations may exceed resource capacities and it was shown that the resource augmentation factors are proportional to the maximal resource augmentation that a single valid mapping may impose. Hence, when the demands are small compared to the capacities, the respective resource augmentation factors are also small and one can even obtain approximations without resource augmentations for the profit variant when using scaling.

Disregarding the potentially large resource augmentations, our findings are significant: neither approximations for general request graphs nor approximations incorporating latencies were known in the realm of the VNEP. Compared to the results presented in [ERS16a] (cf. Theorem 4.4 for an overview), our approximation results do not depend on *any* assumptions on the demand-to-capacity ratios or the achieved benefit. We believe the presented approximation results to be of interest in their own right and we note that using the same techniques a novel approximation result for the *splittable* all-or-nothing multi-commodity flow problem was recently obtained [Liu+19].

While it may in general possible to strengthen the LP formulations using the technique proposed by Bansal et al. in [Ban+15], we note that the respective formulations would irrevocably lead to a numbers of variables exponential both in the (e.g.) treewidth *and* the diameter. Specifically, considering a tree decomposition $\mathcal{T}_r$ of diameter d, the size of the LP formulation would be (lower bounded) by $\mathcal{O}(|V_S|^{\mathrm{tw}(\mathcal{T}_r)\cdot d})$ (see Theorem 4.2). Nevertheless, this approach might be applicable for small requests of small treewidth and we leave the further study of the combination of these results to future work.

To study the practical impact of the obtained approximation results, we have also derived several heuristics for the profit variant of the VNEP. Specifically, vanilla rounding heuristics were studied to investigate the performance of the approximations and two heuristics that *obey capacities* were presented. Here,

even the simple and fast heuristical rounding approach, which discards mappings, whose addition would violate resource capacities, performs well and reaches 77.2% of the profit of the baseline MIP solutions. Even more, the heuristical rounding procedure with LP recomputations outperforms the simple rounding heuristic by on average 20.7% of the baseline's profit, while coming at the cost of longer but reasonable runtimes.

Notably, the performance of our heuristics is clearly influenced by the scarcity of edge resources: on the one hand, the greedy WiNE heuristic provides better solutions in the most cases when considering the lowest ERF of 0.25, while on the other hand, for ERFs of 1.0 and 2.0 our randomized rounding based heuristics yield up to 23.4% more (average) profit. Noting that our heuristics provide better solutions in 99.9% of the cases for ERFs of 1 and 2, this demonstrates the potential usefulness of our approach in practice.

Considering the runtime of the column generation approach, we note that it allows the computation of solutions even for 100 requests in less than 300 seconds for graphs of treewidth 3. This runtime is facilitated by the optimized implementation of the DynVMP algorithm, which has a runtime lying in the range of few to hundreds of milliseconds, which is comparable to other heuristic algorithms. Given the possibility to run the separation procedure in parallel, we believe that the respective algorithms can also scale to orders of magnitude more requests.

"Your assumptions are your windows on the world.
Scrub them off every once in a while or the light won't come in."
– Alan Alda

8

Optimal Virtual Cluster Embeddings and the Hose Based Model

While applications of the VNEP are mostly envisioned in the context of wide-area networks, the performance of cloud applications inside data centers is also sensitive to network conditions [MP12]. Specifically, cloud applications such as MapReduce and scale-out databases generate large amounts of network traffic, and a considerable fraction of their runtime is due to network activity [Cho+11].

Accordingly, over the last years, several systems have been developed for providing the cloud tenant with the illusion of a *Virtual Cluster* [Bal+11; Xie+12]: a star-shaped undirected request connecting virtual machines to a logical switch with uniform request demands.

In this chapter, we debunk some of the myths around the embedding of virtual cluster requests. First, we show that the online VNEP restricted to virtual cluster requests can be solved optimally in polynomial-time using our flow-based embedding algorithm VC-ACE for arbitrary data center topologies. Secondly, we argue that resources may be wasted by enforcing star-topology embeddings, and alternatively promote a hose embedding approach [Kum+02]. We discuss the computational complexity of hose embeddings and derive the HVC-ACE heuristic. Using simulations we substantiate the benefits of hose embeddings.

This chapter is structured as follows. In Section 8.1 virtual clusters are formally introduced and their application inside data centers is discussed. In Section 8.2 the optimal polynomial-time algorithm VC-ACE for virtual clusters is given and in Section 8.3 the hose interpretation of virtual clusters is introduced together with hardness results and a derived heuristic. In Section 8.4 we compare the performance of classic virtual cluster embeddings with the novel hose based interpretation. Section 8.5 concludes the chapter.

8.1 Overview of Virtual Clusters and Related Work

Several systems have been proposed over the last years which allow the application to specify network requirements and construct an inter-connecting virtual network with bandwidth guarantees [Bal+11; Xie+12; Pop+13]. Among those, the virtual cluster abstraction introduced by Ballani et al. [Bal+11] has received much attention: a Virtual Cluster request VC_r is defined as a tuple $\mathrm{VC}_r = (\mathcal{N}(r), d_b(r), d_c(r))$, specifying a *minimal* bandwidth $d_b(r) \in \mathbb{N}$ between the $\mathcal{N}(r) \in \mathbb{N}$ many virtual machines (VMs) of *size* $d_c(r) \in \mathbb{N}$ and a non-oversubscribed *logical switch*, independently of the VM locations in the data center topology (see Figure 8.1 for an example). A virtual cluster is attractive for its simplicity and flexibility: it describes a simple star topology which guarantees the feasibility of all communication patterns in which the aggregate ingress and egress bandwidth of each VM is bounded by $d_b(r)$.

The objective studied for virtual clusters embeddings is generally to minimize the resource usage of the embedding [Bal+11; Xie+12]. Accordingly, the embedding of virtual clusters is a classic instantiation of the online VNEP (cf. Definition 2.8) on undirected graphs. We use the following notation to model virtual clusters as regular requests and adapt the notation slightly to reflect that the requests are undirected.

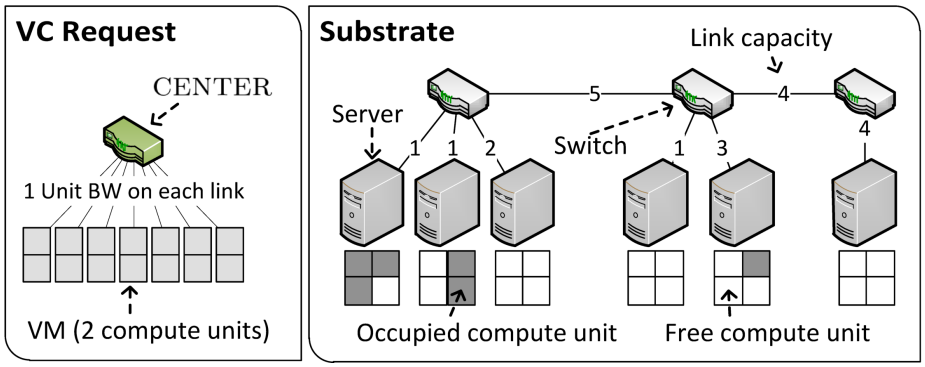


Figure 8.1: The VC_r with $\mathcal{N}(r) = 7$, $d_c(r) = 2$, $d_b(r) = 1$ on the *left* shall be embedded on the substrate on the *right*, such that node and link capacities are respected.

Formally, the virtual cluster $\mathrm{VC}_r = (\mathcal{N}(r), d_b(r), d_c(r))$ is modeled as an undirected graph $G_r = (V_r, E_r)$ with $V_r = \{1, 2, \ldots, \mathcal{N}(r), \textsc{center}\}$ and $E_r = \{\{i, \textsc{center}\} \mid i \in [\mathcal{N}(r)]\}$, where CENTER denotes the logical switch to which the VMs are connected. Accordingly, the demands are set to be $d_r(i) = d_c(r)$, $d_r(\textsc{center}) = 0$, and $d_r(\{i, \textsc{center}\}) = d_b(r)$ for $i \in [\mathcal{N}(r)]$. Notably, as the logical switch has no demand, (i) it can be mapped anywhere and (ii) no costs are considered for its mapping.

Despite the simplicity of the problem, only heuristic algorithms were considered so far for the embedding of virtual clusters [Bal+11; Xie+12]. Moreover, existing algorithms are usually tailored towards specific substrate networks, specifically *aggregated* fat tree data center topologies [ALV08] (see Figure 1.2b). Hence, the respective algorithms *cannot* be used for other modern data center topologies, such as BCube [Guo+09], Jellyfish [Sin+12], MDCube [Wu+09].

In fact, sometimes it is even claimed that the problem of "*allocating virtual cluster requests on graphs with bandwidth-constrained edges is $\mathcal{NP}$-hard*" [Bal+11], and accordingly, researchers have resorted to weaker quality measures, such as *spatial locality* [Xie+12].

8.2 Optimal VC Embeddings

This section presents the fast and optimal algorithm VC-ACE for embedding virtual clusters on general topologies. At the heart of VC-ACE (see Algorithm 8.1) lies the observation that the virtual embedding problem can be reduced to a series of flow problems on an extended substrate graph (see Figure 8.2). We exploit the following facts:

1) The required bandwidth $d_b(r)$ and the respective compute resources $d_c(r)$ of each VM in a virtual cluster is the same. As connections between the VMs and CENTER are embedded as *unsplittable* paths, the substrate's edge capacities (and costs) can be normalized and we may assume $d_b(r) = 1$ to hold.

2) Assuming that the VM mappings m_V as well as the location of the center are fixed, the cost-optimal link mapping can be computed in polynomial-time. Concretely, the minimum-cost unsplittable multi-commodity flow problem with commodities $\{(m_V(i), m_V(\textsc{center}), 1)\}$ can be transformed into an integral minimum-cost single-commodity flow problem by introducing a super source s^+ together along (possibly parallel) edges $e_i^+ = (\mathrm{s}^+, m_V(i))$ with $d_S(e_i^+) = 1$ for $1 \leq i \leq \mathcal{N}(r)$. It suffices then to ask for an integral minimum-cost flow of value $\mathcal{N}(r)$ from s^+ to $m_V(\textsc{center})$. The equivalence of these problems follows from construction, since $d_b(r) = 1$ holds and edge capacities are integral (cf. [KV18]).

3) Assume that the mapping of CENTER is fixed. The *mapping decision* for the VMs $\{1, \ldots, \mathcal{N}(r)\}$ can be incorporated into the integral minimum-cost flow problem in the following way. The super

Algorithm 8.1: VC-ACE Algorithm

Input: Substrate $S = (V_S, E_S)$, request $\mathrm{VC}_r = (\mathcal{N}(r), d_b(r), d_c(r))$
Output: Optimal VC mapping $\hat{m}_V, \hat{m}_E$ if feasible

1 $\hat{f} \leftarrow$ `NULL` **and** $\hat{v} \leftarrow$ `NULL`
2 $V_S^{\mathrm{ext}} = V_S \cup \{\mathrm{s}^+\}$ **and** $E_S^{\mathrm{ext}} = E_S \cup \{(\mathrm{s}^+, u) | u \in V_S\}$
3 $d_S^{\mathrm{ext}}(e) = \begin{cases} \lfloor d_S(e)/d_b(r) \rfloor & , \text{ if } e \in E_S \\ \lfloor d_S(u)/d_c(r) \rfloor & , \text{ if } e = (\mathrm{s}^+, u) \in E_S^{\mathrm{ext}} \end{cases}$
4 $c_S^{\mathrm{ext}}(e) = \begin{cases} c_S(e) \cdot d_b(r) & , \text{ if } e \in E_S \\ c_S(u) \cdot d_c(r) & , \text{ if } e = (\mathrm{s}^+, u) \in E_S^{\mathrm{ext}} \end{cases}$
5 **for** $v \in V_S$ **do**
6 $\quad f \leftarrow$ `MinCostFlow`$(\mathrm{s}^+, v, \mathcal{N}(r), V_S^{\mathrm{ext}}, E_S^{\mathrm{ext}}, d_S^{\mathrm{ext}}, c_S^{\mathrm{ext}})$
7 $\quad$ **if** f **is feasible** and $c_S^{\mathrm{ext}}(f) < c_S^{\mathrm{ext}}(\hat{f})$ **then**
8 $\quad\quad \hat{f} \leftarrow f$ **and** $\hat{v} \leftarrow v$
9 **if** $\hat{f} =$ `NULL` **then**
10 $\quad$ **return** `NULL`
11 **return** `DecomposeFlowIntoMapping`$(\hat{f}, \hat{v})$

source s^+ is connected to *all* substrate nodes $u \in V_S$ via $e_u^+ = (\mathrm{s}^+, u)$ with $d_S(e_u^+) = \lfloor d_S(u)/d_c(r) \rfloor$ and $c_S(e_u^+) = c_S(u) \cdot d_c(r)$. We now consider an integral minimum-cost flow from s^+ to the fixed location of the CENTER, i.e., $m_V(\text{CENTER})$. If such a flow $f : E_S \to \mathbb{N}$ of value $\mathcal{N}(r)$ exists, then $\sum_{u \in V_S} f(e_u^+) = \mathcal{N}(r)$ holds and $f(e_u^+)$ can be identified with the number of VMs that are placed on u. By construction, placing $f(e_u^+) \in \mathbb{N}$ many VMs onto u cannot violate node capacities and costs are correctly accounted for. Lastly, flows may only terminate at $m_V(\text{CENTER})$ and hence each node $u \in V_S$ establishes exactly $f(e_u^+)$ many unsplittable paths to $m_V(\text{CENTER})$.

The above insights are instrumental for designing VC-ACE (see Algorithm 8.1) and for understanding its correctness. Based on 3), if the virtual switch's mapping is fixed, then the optimal embedding can be computed by solving a single integral minimum-cost flow problem (see Line 6) on a specifically constructed graph (see Lines 2-4). For each possible location of CENTER the optimal flow together with the mapping of CENTER is stored (see Lines 7,8). Lastly, if a feasible flow existed, the cost optimal flow $\hat{f}$ is decomposed into paths $P = \langle \mathrm{s}^+, u_1, \ldots, u_n, m_V(\text{CENTER}) \rangle$, yielding a VM placement on node u_1 together with the substrate path $\langle u_1, \ldots, u_n, m_V(\text{CENTER}) \rangle$ towards $m_V(\text{CENTER})$.

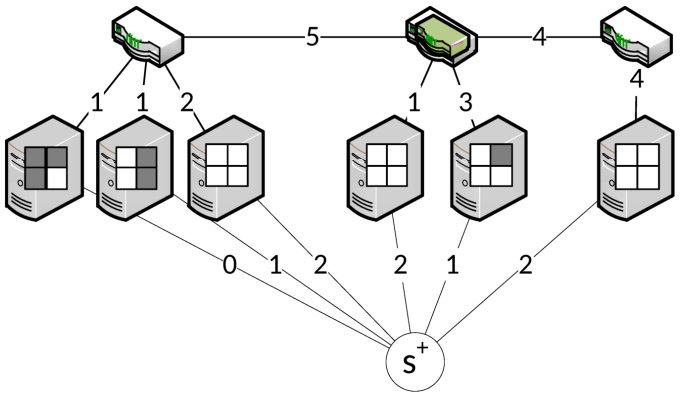


Figure 8.2: The flow problem for the situation from Figure 8.1, if CENTER is mapped to the middle switch.

VC-ACE has a polynomial runtime, which is dominated by solving exactly $|V_S|$ many minimum-cost flow problems. By employing the *Successive Shortest Paths Algorithm*, we obtain a runtime of $\mathcal{O}\left(\mathcal{N}(r) \cdot (n^2 \log n + n \cdot m)\right)$, where $n = |V_S|$ and $m = |E_S|$. On tree topologies like the fat tree, the runtime of VC-ACE can be reduced to $\mathcal{O}(n \cdot \mathcal{N}(r))$, which is on par with the best known heuristic approaches [Bal+11; Xie+12].

8.3 Hose-Based VC Embeddings

A virtual cluster essentially supports any *communication pattern* between the VMs for which the aggregate ingress and aggregate egress bandwidth at each VM is at most $d_b(r)$. If the only role of the logical switch in the VC_r abstraction is to facilitate these communication patterns, it is wasteful to enforce the explicit star embedding and the redirection via the unique center. Thus, one may consider to remove the switch and rather support the communication using *direct interconnections* between VM pairs in a hose fashion [Duf+99].

In the following, we introduce the respective *hose embedding* problem and study how to solve it. We define the hose virtual cluster embedding problem and highlight its potential benefits in Section 8.3.1. However, we also show in Section 8.3.2 that the unsplittable hose embedding is computationally hard, and present an optimal Mixed-Integer Programming (MIP) formulation in Section 8.3.3, which also forms the basis of our proposed splittable-hose cluster embedding algorithm HVC-ACE presented in Section 8.3.4.

8.3.1 Problem Definition and Motivation

Henceforth, we denote by $G_r^C = (V_r^C, E_r^C)$ the *clique* graph of request VC_r with $V_r^C = \{i \mid i \in [\mathcal{N}(r)]\}$ only containing the VMs and with $E_r^C = \{(i,j) \mid i,j \in V_r^C, i < j\}$ being the set of directed interconnections. Note that we employ directed edges only for notational purposes and that the sought after paths will still be undirected. A feasible solution to the hose-based virtual cluster embedding problem is characterized as follows:

1. The mapping of VMs must not violate node capacities.
2. Each *route* $(i,j) \in E_r^C$ is realized as *simple* path $m_E(\{i,j\}) \subseteq E_S$, connecting $m_V(i)$ and $m_V(j)$.
3. The (oblivious) routing according to m_E does not violate the substrate's edge capacities *under any communication pattern.* Concretely, there exists an integral bandwidth allocation $a^{u,v} \leq d_S(u,v)$ for $\{u,v\} \in E_S$, such that for all feasible traffic matrices $M \in \mathbb{R}^{\mathcal{N}(r) \times \mathcal{N}(r)}$, i.e., for each VM $i \in V_r^C$ it holds $\sum_{(j,i) \in E_r^C} M_{j,i} + M_{i,j} \leq d_b(r)$, and the bandwidth reservation is not exceeded:

$$\sum_{\{i,j\} \in E_r^C : \{u,v\} \in m_E(\{i,j\})} M_{i,j} \leq a^{u,v}\,.$$

Given bandwidth reservations $a^{u,v}$ the cost of a *hose-based* virtual cluster embedding is defined analogously to the definition of costs as

$$d_c(r) \cdot \sum_{i \in V_r^C} c_S(m_V(i)) + d_b(r) \cdot \sum_{e \in E_S} a^{u,v} \cdot c_S(e).$$

In the following, we will refer to this virtual cluster interpretation omitting the logical switch as the *hose-based virtual cluster*, short HVC.

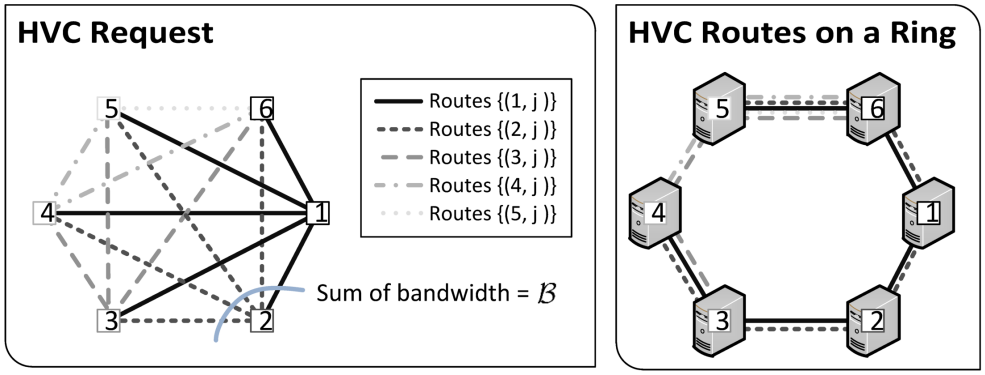


Figure 8.3: A hose-based virtual cluster HVC=**(6,1,1)** *(left)* and a feasible embedding of the HVC on a 6-node substrate ring *(right)* with 2 units of bandwidth on each link and one compute unit on each server.

In order to clarify and highlight the difference between the two virtual cluster interpretations, we consider the example in Figure 8.3: a ring substrate with 6 nodes, where nodes have a capacity of one unit and links have a capacity of two units, and assume the virtual cluster with $\mathcal{N}(r) = 6, d_b(r) = 1, d_c(r) = 1$.

First we observe that it is impossible to map a logical switch CENTER in this scenario: each ring node must host one VM and any placement of the CENTER therefore requires the establishment of $\mathcal{N}(r) - 1 = 5$ many independent paths to the respective substrate node onto which CENTER is mapped. This is impossible, since each node's accumulated bandwidth is 4; no 'classic' (star) VC_r embedding exists.

In the hose-based virtual cluster HVC however, a feasible embedding can be computed (depicted in Figure 8.3). To see that this is indeed a feasible solution, consider e.g., the substrate edge e connecting $m_V(5)$ and $m_V(6)$. The edge e lies on the routing paths of the VM pairs $R(e) = \{(1,5),(2,5),(3,6),(4,6),(5,6)\}$. Despite e's capacity being 2, this still is a feasible solution. The load on e amounts to $\sum_{(i,j)\in R(e)} M_{i,j}$ for a traffic matrix $M_{i,j}$. As $M_{i,j}$ is required to respect the *cumulative* bandwidth $d_b(r)$, this load is bounded by $M_{1,5} + M_{2,5} \leq 1$ and $M_{3,6} + M_{4,6} + M_{5,6} \leq 1$ and for the maximal allocations a^e that may be induced holds $2 \leq d_S(e)$.

This example highlights a *qualitative* disparity, in the sense that only HVCs can be embedded. Analogously, it is easy to construct examples where both VC_r and HVC can be embedded, but the corresponding optimal embeddings differ significantly in their costs. For instance, consider the above ring network with an additional node u whose compute capacity is 0, and which connects to all ring nodes with edges of bandwidth 1 at some cost $c > 0$. While the HVC solution still has the same cost, the cost of the unique VC_r solution is $\mathcal{N}(r) \cdot c$ which can be arbitrarily high. Our computational evaluation (see Section 8.4) shows that this qualitative disparity also arises in other topologies, e.g., in fat trees.

8.3.2 Computational Complexity

The above example has shown the potential benefit of using the hose model. In the following, we study the computational complexity of hose embeddings and show their hardness. In particular, we show that an optimal hose embedding is actually a star embedding, if edge capacities can be neglected. However, if edge capacities cannot be neglected, the respective embedding problem is hard and even inapproximable, unless $\mathcal{P} = \mathcal{NP}$ holds.

Case I: Bandwidth Requirements Are Negligible. We first present the rather intriguing result, that VC and HVC embeddings are the same, if sufficient bandwidth is available, concretely if $d_S(e) \geq \mathcal{N}(r) \cdot d_b(r)$ holds for all edges $e \in E_S$. This is a non-trivial result which follows from the famous VPN conjecture,

which was proven by Goyal et al. [GOS08]. In a nutshell, the VPN conjecture states that in *uncapacitated networks*, hose embedding problems with symmetric bandwidth bounds and no restrictions on routing (known as the `SymG` model), can be reduced to hose problem instances in which routing paths *must form a tree* (known as the `SymT` model). Based on this result, any optimal HVC embedding will have a central 'hub' (corresponding to the VC's logical switch) in the network, such that all traffic passes this node [GOS08].

Thus, when not considering link capacities any optimal HVC embedding contains a central 'hub'. Recall algorithm VC-ACE presented in Section 8.2 explicitly computes the *minimum cost* embedding towards such a node, called CENTER by us. By construction of the $\mathcal{N}(r)$ flows towards the center carrying exactly $d_b(r)$ much bandwidth, no VC solution will use more than $\mathcal{N}(r) \cdot d_b(r)$ bandwidth and thus by the VPN conjecture, the equality of VC and HVC solutions follows.

Case II: Bandwidth Requirements Are Not Negligible. We will now show that the HVC embedding problem is generally – on non-tree topologies – inapproximable unless $\mathcal{P} = \mathcal{NP}$ holds. This result again follows from the literature on hose embedding problems, where terminals are fixed. In particular, it was proven in [Gup+01] that computing optimal solutions in the capacitated hose model is $\mathcal{NP}$-hard. Furthermore, it was shown that even deciding whether a solution exists is $\mathcal{NP}$-complete, implying that – unless $\mathcal{P} = \mathcal{NP}$ holds – no polynomial-time approximation algorithm can exist.

A simple reduction shows that this result translates to the HVC embedding problem: Given is a symmetric hose problem on a graph (V_S, E_S) with the set of terminals $T \subseteq V_S$ and integral bandwidth bounds $b_t \in \mathbb{N}$ for each terminal $t \in T$. By setting $d_S(t) = b_t$ and $c_S(t) = 0$ for all $t \in T$, the original hose problem is equivalent to the respective HVC problem of embedding the virtual cluster $\mathrm{VC}_r(\sum_{t \in T} b_t, 1, 1)$ onto the substrate network with edge costs and capacities remaining the same.

8.3.3 Exact Unsplittable Hose Algorithm

The above inapproximability result rules out any type of efficient approximation algorithms. Hence we give the exact Mixed-Integer Programming formulation HVC-OSPE for obtaining *Optimal* Single-Path Embeddings (OSPE).

Our formulation (see MIP 8.2) builds upon the compact hose formulation by Altin et al. [Alt+07]. We employ the following additional notation. We use $\overleftrightarrow{E_S} = \{(u,v),(v,u) \mid \{u,v\} \in E_S\}$ to denote the set of bi-directed substrate edges. We use $\delta^+(W) = \{(u,v) \in \overleftrightarrow{E_S} \mid u \in W, v \notin W\}$. For breaking symmetries, we assume some arbitrary numbering of the VMs, given by the bijection $\sigma : V_S \to [|V_S|]$.

We introduce node mapping variables $y_i^u \in \{0,1\}$, with $y_i^u = 1$ iff. $i \in V_r^C$ is mapped onto the substrate node $u \in V_S$. Given these mapping variables, the symmetric hose formulation of [Alt+07] can be adapted: The routing variables $z_{i,j}^{u,v} \in \{0,1\}$ for $(i,j) \in E_r^C$ and $(u,v) \in \overleftrightarrow{E_S}$ determine the simple path between *the substrate nodes* onto which i and j have been mapped (see Constraint 8.6). The cumulative load variable of a substrate edge $\{u,v\}$ is introduced as $a^{u,v} \in \mathbb{N}$. The load variables are lower bounded according to the 'dual' variables $\omega_s^{u,v} \geq 0$ for all $s \in V_r^C, \{u,v\} \in E_S$ (see Constraints 8.11 and 8.8, and [Alt+07] for an in-depth explanation).

With respect to the node mapping, Constraint 8.2 enforces that each virtual cluster node is mapped onto exactly one substrate node. Constraints 8.4 and 8.10 bound the resource allocations in the substrate by the respective node and edge capacities. Lastly, we introduce Constraint 8.3 to *break symmetries*: as all nodes in V_r^C have identical resource requirements, a feasible node mapping induces up to $\mathcal{N}(r)!$ equivalent ones. Constraint 8.3 only allows for one of these permutations.

Despite breaking symmetries the formulation HVC-OSPE remains hardly solvable even for small networks. Our initial computational experiments have shown that the mean runtime of HVC-OSPE for

Mixed-Integer Program 8.2: HVC-OSPE

$$\min \sum_{i \in V_r^C, u \in V_S} c_S(u) \cdot y_i^u + \sum_{\{u,v\} \in E_S} c_S(u,v) \cdot a^{u,v} \tag{8.1}$$

$$\sum_{u \in V_S} y_i^u = 1 \qquad \forall i \in V_r^C \tag{8.2}$$

$$\sum_{u \in V_S} \sigma_u \cdot (y_i^u - y_{i+1}^u) \leq 0 \qquad \forall i \in V_r^C \setminus \{\mathcal{N}(r)\} \tag{8.3}$$

$$\sum_{i \in V_r^C} d_c(r) \cdot y_i^u \leq d_S(u) \qquad \forall u \in V_S \tag{8.4}$$

$$a^{u,v} \leq d_S(u,v) \qquad \forall \{u,v\} \in E_S \tag{8.5}$$

$$\sum_{(u,v) \in \delta^+(u)} z_{i,j}^{u,v} - \sum_{(v,u) \in \delta^-(u)} z_{i,j}^{v,u} = y_i^u - y_j^u \qquad \forall (i,j) \in E_r^C, u \in V_S \tag{8.6}$$

$$\sum_{i \in V_r^C} d_b(r) \cdot \omega_i^{u,v} \leq a^{u,v} \qquad \forall \{u,v\} \in E_S \tag{8.7}$$

$$z_{i,j}^{u,v} + z_{i,j}^{v,u} \leq \omega_i^{u,v} + \omega_j^{u,v} \qquad \forall (i,j) \in E_r^C, \{u,v\} \in E_S \tag{8.8}$$

optimally embedding a 10-node VC onto a 20-node substrate already exceeds 10 minutes. One reason for this runtime can be found in the number of integral flow variables which amounts to $\Theta(|E_S| \cdot \mathcal{N}(r)^2)$.

8.3.4 Algorithm for the Splittable Hose-Model

As discussed above, finding HVC embeddings is strongly $\mathcal{NP}$-hard in the unsplittable path model. Furthermore, using Mixed-Integer Program 8.2 to obtain (optimal) solutions in *reasonable* time seems out of reach.

To compute hose-based embeddings more efficiently, we have to drop (1) the flexible node mapping as well as (2) the unsplittable path routing model. Accordingly, this section presents the polynomial-time Linear Program 8.3 for computing optimal Hose Multi-Path Routings (HMPR) under fixed node mappings. In contrast to HVC-OSPE, the formulation HMPR can be solved within few minutes for much larger problems.[1] Together with an efficient node mapping heuristic based on algorithm VC-ACE, we obtain the effective stand-alone heuristic HVC-ACE for the (splittable) hose-based virtual cluster embedding.

Based on the Mixed-Integer Program 8.2 we obtain the *Linear Program* 8.3. First, if all VMs are feasibly mapped, i.e., without violating node capacities, via the function $m_V : V_r^C \to V_S$, Constraints 8.2 to 8.4 are not needed. However, more importantly, the computation of correct link loads can be significantly simplified, by actually discarding the $\Theta(|E_S| \cdot \mathcal{N}(r)^2)$ many flow variables. Dropping the constraint of unsplittable path routing, Constraints 8.6 and 8.8 can be equivalently stated using the cut-set inequalities expressed in Constraint 8.12 (cf. Altin et al. [Alt+07]): given any substrate node set W, to which VM i but not VM j has been mapped, there must exist a 'path' leaving W with value 1 such that the respective dual variables are bounded accordingly.

The *exponential* number of constraints can be *separated* efficiently using maximum-flow computations (cf. [KM98]) allowing to solve the formulation in polynomial-time [GLS88]. Lastly, the splittable routing

[1] In our computational evaluation (see Section 8.4) with hundreds of nodes and edges and up to 30 node requests, HMPR computed more than 95% of the solutions in less than 3 minutes. Furthermore, our prototypical implementation of the formulation HMPR uses the simple Edmonds-Karps maximum-flow algorithm to separate the Constraints 8.12 of Linear Program 8.3 using $|E_r| = \Theta(|V_r|^2)$ many flow computations. The runtime can be reduced significantly by using Hao and Orlin's algorithm requiring only $|V_r|$ many flow computations (see [KM98] for an explanation).

Linear Program 8.3: Hose Multi-Path Routing (HMPR)

$$\min \sum_{\{u,v\}\in E_S} c_S(u,v)\cdot a^{u,v} \tag{8.9}$$

$$a^{u,v} \leq d_S(u,v) \qquad \forall \{u,v\} \in E_S \tag{8.10}$$

$$\sum_{i\in V_r^C} d_b(r)\cdot \omega_i^{u,v} \leq a^{u,v} \qquad \forall \{u,v\} \in E_S \tag{8.11}$$

$$\sum_{(u,v)\in\delta^+(W)} (\omega_i^{u,v} + \omega_j^{u,v}) \geq 1 \qquad \forall (i,j) \in E_r \forall W \subset V_S : m_V(i) \in W, m_V(j) \notin W \tag{8.12}$$

Algorithm 8.4: HVC-ACE Embedding Algorithm

Input: Substrate $G_S = (V_S, E_S)$, request $\mathrm{VC}_r = (\mathcal{N}(r), d_b(r), d_c(r))$, cost factor $\lambda \geq 1$
Output: Splittable HVC-Embedding m_V, m_E

```
E_S^ext ← ∅
for e ∈ E_S do
    E_S^ext = E_S^ext ⊔ {e, e'}
    d_S^ext(e) = d_S(e) and d_S^ext(e') = ∞
    c_S^ext(e) = c_S(e) and c_S^ext(e') = c_S(e) · λ
m_V, m_E ← VC-ACE(V_S, E_S^ext, VC_r = (N(r), d_b(r), d_c(r)))
if m_V ≠ NULL then
    m_E ← HMPR(VC_r, m_V)
    if m_E ≠ NULL then
        return m_V, m_E
return NULL
```

for $(i,j) \in E_r$ can be recovered from the dual variables by computing a minimum-cost flow of value 1 from $m_V(i)$ to $m_V(j)$ with edge capacities $\omega_i^{u,v} + \omega_j^{u,v}$ for $\{u,v\} \in E_S$. Such a flow always exists as Constraint 8.12 enforces that any cut must have at least capacity 1 (cf. Constraint 8.8 of MIP 8.2).

Hence, the Linear Program 8.3 allows to compute optimal splittable hose-routings in polynomial-time for *fixed* node mappings. We now describe a heuristic to find a corresponding node mapping to obtain the efficient embedding algorithm HVC-ACE. Since optimal VC and HVC solutions coincide in uncapacitated networks (see Section 8.3.2), we employ an adaption of the VC mapping algorithm VC-ACE (see Algorithm 8.1). For each edge in the original graph, a copy of infinite capacity is added, having $\lambda \geq 1$ times the original cost. Clearly, on this adapted graph (V_S, E_S^{ext}) a feasible solution – given the existence of a feasible node mapping – always exists. By varying the factor λ, the cost of using an infinite-capacity edge can be controlled. Setting $\lambda = 1$, the cost structure of the original graph is not changed, while by setting $\lambda = \infty$ the usage of non-existent edges can be minimized. Thus, for $\lambda = \infty$, the node mapping equals the one found using VC-ACE, if one existed.

8.4 Evaluation

This section compares the performance of the optimal VC embedding and our heuristic hose-based embeddings. In particular we show that by using Algorithm HVC-ACE, the chances of being able to accept a single request can be up to 60% higher than in the classic VC model. We furthermore show that the hose-abstraction may reduce the resource footprint by up to 25% on fat tree topologies.

Our evaluation setup is as follows. We consider the (non-aggregated) fat tree (12-port switches, 432 servers) and the MDCube (4 BCubes with $n = 12$ and $k = 1$) data center topologies. We assume uniform

edge capacities and that each server offers 2 VM slots. Requests are embedded over time in an online fashion with exponentially distributed inter-arrival times and duration. The mean of the inter-arrival time is chosen to impose a system load of 75% (w.r.t. node utilization if all requests can be accepted). The size $\mathcal{N}(r)$ of the virtual clusters is chosen uniformly at random from the set $\{10, \ldots, 30\}$. The bandwidth $d_b(r)$ is chosen uniformly at random between 20% to 100% of the available bandwidth of a single (unused) substrate link. To impose an initial system load, requests are embedded within the first 45 time units (three generations of requests) using VC-ACE if possible. Then, a single data point is generated by embedding the next given request, using VC-ACE and HVC-ACE. Here HVC-ACE denotes the best solution found for the cost parameter $\lambda \in \{1, 5, 10, \infty\}$.

We evaluate two metrics: the acceptance ratio and the relative resource footprint. The *acceptance ratio* captures the ratio of requests that can be successfully embedded. The *relative* resource footprint is the quotient of the embedding costs of the solutions of HVC-ACE and VC-ACE, if VC-ACE found a solution. As we consider unit node and edge costs, the cost is proportional to the edge usage.

Figure 8.4 *(left)* shows a significant advantage of HVC-ACE in terms of acceptance ratio: starting from 13 nodes, VC-ACE can only embed roughly 40% of the requests. At 25 nodes, the acceptance ratio drops, to roughly 20%. The acceptance ratio of HVC-ACE remains close to 100% for up to 23 nodes. The drops of the acceptance ratio are related to the number of ports of the switches (12 port switches in the fat tree). Figure 8.4 *(right)* shows the impact of using HVC-ACE instead of VC-ACE on the footprint (when VC-ACE found a solution). The x-axis plots the relative footprint of HVC-ACE normalized by the footprint of VC-ACE. By adopting the hose model (i.e., when using HVC-ACE) the footprint can be reduced for roughly a quarter of all requests by up to 30%. Note that for 20% of the requests, the

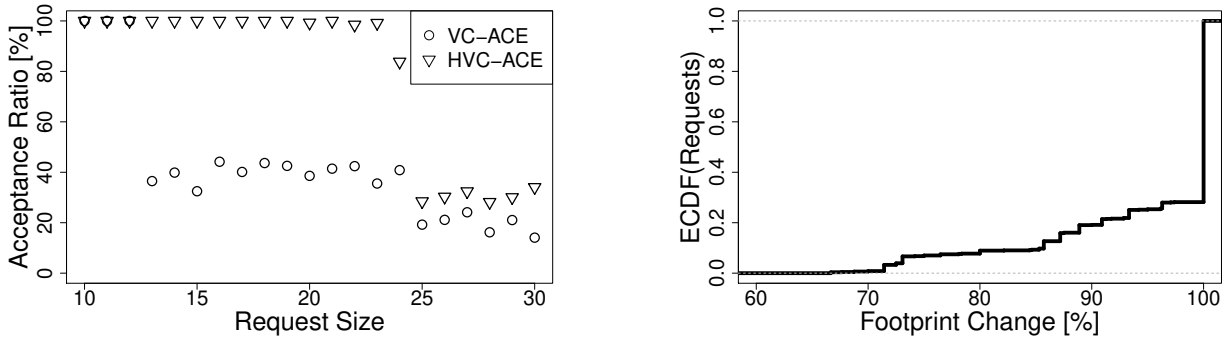


Figure 8.4: Experiments on a 432 server fat tree. Acceptance ratio of VC-ACE and HVC-ACE *(left)* and footprint benefits of HVC-ACE compared to VC-ACE *(right)*.

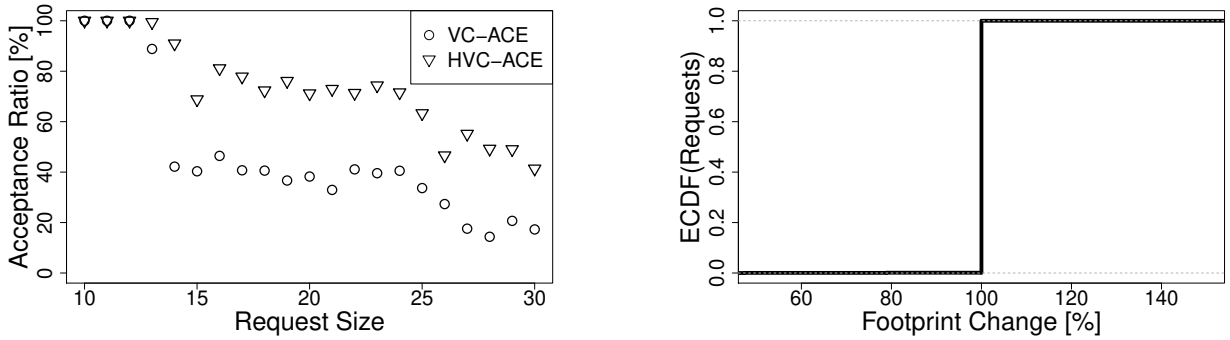


Figure 8.5: Experiments on a 576 server MDCube. Acceptance ratio of VC-ACE and HVC-ACE *(left)* and footprint benefits of HVC-ACE compared to VC-ACE *(right)*.

footprint was reduced by at least 10%.

The advantages of the HVC interpretation on hypercubic topologies are depicted in Figure 8.5: although HVC-ACE does not yield lower footprints than VC-ACE (if a solution was found), HVC-ACE can still provide an average improvement of 30% in the acceptance ratio, for $\mathcal{N}(r) \geq 14$.

8.5 Summary and Novelty of Contributions

In this chapter the virtual cluster embedding problem, i.e., a specific online VNEP instantiation, was considered. We showed that the problem is not $\mathcal{NP}$-hard, but can be solved in polynomial-time even on arbitrary topologies using our algorithm VC-ACE. Moreover, we have introduced the hose interpretation of virtual clusters and proposed the efficient embedding algorithm HVC-ACE. Our evaluation shows that the splittable hose abstraction can generally greatly improve the acceptance ratio and may yield better solutions in terms of resource utilization on fat trees.

Considering the novelty of our approach, we note that Bansal et al. independently and unbeknownst to of us gave an informal description of the VC-ACE algorithm in their paper [Ban+11]. Yet, the idea to frame the virtual cluster abstraction as a hose model is novel and independent of their result. Based on the results described in this paper, Fattohi derived a first competitive online algorithm for virtual cluster embeddings in his master thesis [Fat18].

"Industrial production, the flow of resources in the economy, the exertion of military effort in a war, the management of finances –all require the coordination of interrelated activities. What these complex undertakings share in common is the task of constructing a statement of actions to be performed, their timing and quantity (called a program or schedule) ..."

– George Dantzig

9 The Temporal Offline VNEP

As already noted in the above chapter, today's data center networks are often largely oversubscribed, rendering network bandwidth a scarce resource shared across many tenants. Additionally, many applications cycle through different phases, only some of which are network-intensive (e.g., the MapReduce shuffle phase). The traffic patterns for example measured in [Xie+12] indicate that popular cloud applications only generate substantial traffic during only 30%-60% of the entire execution. Accordingly, virtual networks should support temporally varying specifications.

Besides the changing requirements over time, applications (and hence virtual network requests) may also differ in their scheduling requirements: while some applications must be started immediately upon request, others may come with certain flexibilities on when they are executed, e.g., when combined with a corresponding price incentive [Hen+10; MKC13] or adaptive resource and spot market pricing schemes [AKK12; PH13].

In this chapter we initiate the study of the Temporal Virtual Network Embedding Problem (TVNEP) formally introduced in Definition 2.12: requests must be embedded for a given duration and within a specified time interval. Hence, the TVNEP conceptually consists of three tasks: (1) deciding which of the requests to embed, (2) finding good embeddings for the requests, and (3) scheduling the requests in such a fashion, according to their temporal specification, that no resource capacities are exceeded.

Furthermore note that the TVNEP is not only relevant for data-centers and high performance computing applications, but also for wide-are networks. For example, Google's *B4 network* [Jai+13] connects roughly a dozen data centers using a Software-Defined Networking (SDN) approach: the bandwidth-intensive data copies from one site to another, are planned from the logically centralized perspective of the SDN controller. This allows to run the network at higher utilizations and to prioritize interactive applications during periods of failure or resource constraints. According to the authors, no more than a few dozen data center deployments are anticipated in the near future, which renders such a central control of bandwidth feasible.

Overview. We present multiple mathematical programming formulations, and devise techniques to reduce the problem's complexity to enable solving the TVNEP on moderately sized instances to optimality. As we are interested mainly in the potential benefits of harnessing temporal flexibilities, a restricted model is considered: we assume splittable edge embeddings (cf. Definition 2.14) and will assume in the evaluation that the virtual nodes have been already mapped. On the one hand, these restrictions render the VNEP significantly easier to solve. On the other hand, embeddings of flows and corresponding scheduling decisions have to be made, still enabling the analysis of the benefit of temporal flexibilities.

With respect to the presented Mixed-Integer Programming formulations for the TVNEP, we make the following contributions:

1. We show that the TVNEP can be formulated as a continuous-time model, which in contrast to discrete models facilitates a compact and accurate representation of time. We present two different

Mixed Integer Program (MIP) formulations for this continuous model: the first, the so-called Δ-model, is based on *state change* representations at events and the second, the Σ-model, is based on explicit state representations. We discuss the (dis)advantages of either of these approaches, and argue that Σ formulations give much better relaxations, hence speeding up the branch-and-bound algorithm employed to solve the model.

2. Our main technical contribution is the cΣ-model: a compact and improved Σ-model which is based on rigorous state-space and symmetry reductions. The model clearly outperforms the other formulations, and enables us to solve moderately sized instances in the first place.
3. We report on our computational evaluation, and show that using the cΣ approach, solving the TVNEP to optimality is feasible for reasonable problem instances, and for different objectives, including access control, load balancing and makespan minimization. Furthermore, the cΣ-model significantly outperforms the Δ- and the simple Σ-model.

The remainder of this chapter is structured as follows. We discuss the two main approaches to formulate continuous-time programs in Section 9.1: an event based model and a state based model. Based on the state model, in Section 9.2 the compact state formulation is introduced and in Section 9.3 additional objectives under temporal flexibilities are discussed. We report on our computational experiments in Section 9.4 and conclude in Section 9.5.

9.1 The Continuous-Time Approach

As noted above, within this chapter *splittable* edge mappings are considered. We adapt the respective integer MCF Formulation 6.3 accordingly and present the complete formulation as parts of the formulation will be used throughout this section. In particular, we employ the MCF base formulation for the VNEP to compute time-invariant embeddings (see MIP 9.1). The splittable edge mappings are realized by relaxing the edge mapping variables $\vec{z}$: these are now real valued.

As stated above, we will show that the TVNEP problem can be modeled using a continuous-time approach. This is attractive as it avoids inaccuracies due to time discretizations and therefore allows us to solve the

Mixed-Integer Program 9.1: Base for Computing Time-Invariant Splittable Embeddings

$$\sum_{u \in V_S^{r,i}} y^u_{r,i} = x_r \qquad \forall r \in \mathcal{R}, i \in V_r \tag{9.1}$$

$$\sum_{u \in V_S \setminus V_S^{r,i}} y^u_{r,i} = 0 \qquad \forall r \in \mathcal{R}, i \in V_r \tag{9.2}$$

$$\sum_{(u,v) \in \delta^+(u)} z^{u,v}_{r,i,j} - \sum_{(v,u) \in \delta^-(u)} z^{v,u}_{r,i,j} = y^u_{r,i} - y^u_{r,j} \qquad \forall r \in \mathcal{R}, (i,j) \in E_r, u \in V_S \tag{9.3}$$

$$z^{u,v}_{r,i,j} = 0 \qquad \forall r \in \mathcal{R}, (i,j) \in E_r, (u,v) \in E_S \setminus E_S^{r,i,j} \tag{9.4}$$

$$\sum_{i \in V_r} d_r(i) \cdot y^u_{r,i} = a^u_r \qquad \forall r \in \mathcal{R}, u \in V_S \tag{9.5}$$

$$\sum_{(i,j) \in E_r} d_r(i,j) \cdot z^{u,v}_{r,i,j} = a^{u,v}_r \qquad \forall r \in \mathcal{R}, (u,v) \in E_S \tag{9.6}$$

$$x_r \in \{0,1\} \qquad \forall r \in \mathcal{R} \tag{9.7}$$

$$y^u_{r,i} \in \{0,1\} \qquad \forall r \in \mathcal{R}, i \in V_r, u \in V_S \tag{9.8}$$

$$z^{u,v}_{r,i,j} \in [0,1] \qquad \forall r \in \mathcal{R}, (i,j) \in E_r, (u,v) \in E_S \tag{9.9}$$

$$a^x_r \geq 0 \qquad \forall r \in \mathcal{R}, x \in G_S \tag{9.10}$$

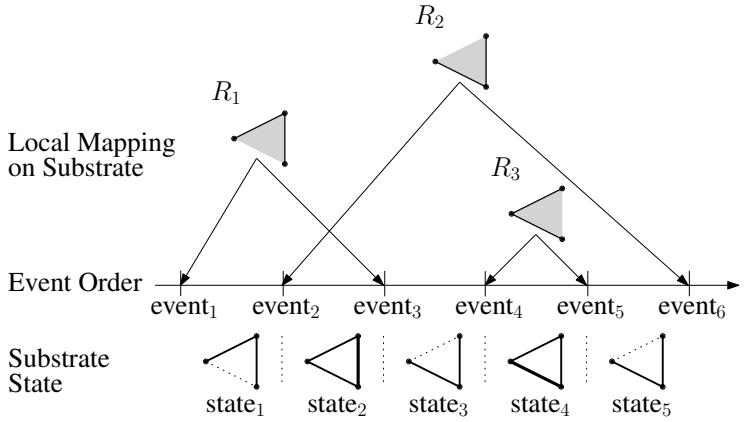


Figure 9.1: Shown are local substrate allocations of requests R_1, R_2, R_3 on a three node substrate. By assigning the start and end of requests to event points, states between events can be reconstructed to check the feasibility of allocations.

continuous VNEP as stated in Definition 2.12. We begin by discussing the conceptual model of abstract event points, and then derive two complementary ways to model the TVNEP in our continuous-time framework: the Δ-model and the Σ-model. While the Δ-model only represents state changes and therefore requires less variables, the Σ-model introduces explicit state variables to improve the LP-relaxations. We will argue that the latter is preferable, and it will also build the basis of our optimized cΣ-model presented in the subsequent section.

Our mathematical programming formulations rely on the event point model introduced in the following, which will allow to check the feasibility of possible solutions to the TVNEP.

9.1.1 The Abstract Event Point Model

To check whether resource capacities are obeyed at each point in time $t \in [0, \mathcal{T}]$, it suffices to consider the $2 \cdot |\mathcal{R}| - 1$ many intervals in which resource allocations are invariant. Denoting by t_r^+, and t_r^- the start and end times of request $r \in \mathcal{R}$ and by $T^{\pm} = \{t_r^+, t_r^- \mid r \in \mathcal{R}\}$ all points in time in which resource allocations change, the respective intervals in which resource allocations are invariant equal $\{[t^-, t^+] \mid t^-, t^+ \in T^{\pm}, (t^-, t^+) \cap T^{\pm} = \emptyset\}$, where (t^-, t^+) is to be read as time interval. Our models compute these time-invariant states using an event point model illustrated in Figure 9.1. Formally, we define the set of events $\mathcal{E} = \{e_1, \ldots, e_{2\cdot|\mathcal{R}|}\}$ and the set of states $\mathcal{S} = \{s_1, \ldots, s_{2\cdot|\mathcal{R}|-1}\}$.

In the following, we detail our core temporal model, which is based on this event point model, as Mixed-Integer Program 9.2. For modeling the assignment of the start of a request $r \in \mathcal{R}$ to an event point $e_i \in \mathcal{E}$ the binary variables $\chi^+_{r,e_i} \in \{0,1\}$ are used (cf. Constraint 9.21). Analogously, to map the end of a request r to an event point e_i the variables $\chi^-_{r,e_i} \in \{0,1\}$ are employed (cf. Constraint 9.22). Constraints 9.12 to 9.14 model the bijection between event points and the start and end of requests, such that only a single request event may be mapped to each event point $e_i \in \mathcal{E}$.

Furthermore, the time t_{e_i} of event $e_i \in \mathcal{E}$ and the starting t_r^+ and ending time t_r^- of requests $r \in \mathcal{R}$ are introduced (cf. Constraints 9.23 to 9.25). While Constraint 9.15 enforces the monotonicity of the event point times, Constraints 9.17 to 9.20 set the requests' start and end times according to the event mapping variables $\vec{\chi}^+$ and $\vec{\chi}^-$. Concretely, the start time t_r^+ and the end time t_r^- for request $r \in \mathcal{R}$ are lower and upper bounded by the respective event time t_{e_i} such that equality holds when $\chi^+_{r,e_i} = 1$ or $\chi^-_{r,e_i} = 1$ holds, as then the second summand equals 0. Lastly, Constraint 9.16 enforces that the duration of any request equals the request's duration.

Importantly, the Mixed-Integer Program 9.2 does only map the requests' start and end to events, assigns point in times to each event and links these with the start and end of the respectively mapped requests. Accordingly, MIP 9.2 does not enforce feasibility of the states between event points. In the following two complementary approaches to reconstruct the states and enforce the respective feasibility are introduced.

Mixed-Integer Program 9.2: General Temporal Extension Without Enforcing Feasibility

$$\text{Constraints and Variables (9.1) - (9.10)} \tag{9.11}$$

$$1 = \sum_{e_i \in \mathcal{E}} \chi^+_{r,e_i} \qquad \forall r \in \mathcal{R} \tag{9.12}$$

$$1 = \sum_{e_i \in \mathcal{E}} \chi^-_{r,e_i} \qquad \forall r \in \mathcal{R} \tag{9.13}$$

$$1 = \sum_{r \in \mathcal{R}} \chi^+_{r,e_i} + \chi^-_{r,e_i} \qquad \forall e_i \in \mathcal{E} \tag{9.14}$$

$$t_{e_i} \leq t_{e_{i+1}} \qquad \forall e_i \in \mathcal{E} : i < |\mathcal{E}| \tag{9.15}$$

$$t^d_r = t^-_r - t^+_r \qquad \forall r \in \mathcal{R} \tag{9.16}$$

$$t^+_r \leq t_{e_i} + (1 - \sum_{j=1,\dots,i} \chi^+_{r,e_j}) \cdot \mathcal{T} \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E} \tag{9.17}$$

$$t^+_r \geq t_{e_i} - (1 - \sum_{j=i,\dots,|\mathcal{E}|} \chi^+_{r,e_j}) \cdot \mathcal{T} \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E} \tag{9.18}$$

$$t^-_r \leq t_{e_i} + (1 - \sum_{j=1,\dots,i} \chi^-_{r,e_j}) \cdot \mathcal{T} \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E} \tag{9.19}$$

$$t^-_r \geq t_{e_i} - (1 - \sum_{j=i,\dots,|\mathcal{E}|} \chi^-_{r,e_j}) \cdot \mathcal{T} \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E} \tag{9.20}$$

$$\chi^+_{r,e_i} \in \{0,1\} \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E} \tag{9.21}$$

$$\chi^-_{r,e_i} \in \{0,1\} \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E} \tag{9.22}$$

$$t_{e_i} \in [0, \mathcal{T}] \qquad \forall e_i \in \mathcal{E} \tag{9.23}$$

$$t^+_r \in [0, \mathcal{T}] \qquad \forall r \in \mathcal{R} \tag{9.24}$$

$$t^-_r \in [0, \mathcal{T}] \qquad \forall r \in \mathcal{R} \tag{9.25}$$

9.1.2 Δ-Model: Representing only State Changes

A first and intuitive way to formulate the continuous-time program, is to represent state allocations (at states $\mathcal{S}$), by encoding only the state *differences* or *changes* $\Delta^x_{e_i} \in \mathbb{R}$ at event points $e_i \in \mathcal{E}$ for substrate resource $x \in G_S$. We will refer to this model as the Δ-model. Given these changes, the state allocations at state s_i for resource $x \in G_S$ compute to $\sum_{j=1}^{i} \Delta^x_{e_i}$. However, to compute the $\vec{\Delta}$ variables, conditional assignments of the following form are necessitated for all requests $r_j \in \mathcal{R}$ and all substrate resources $x \in V_S \cup E_S$:

$$\Delta^x_{e_i} = \begin{cases} +a^x_{r_j} & \text{, if } \chi^+_{r_j,e_i} = 1 \\ -a^x_{r_j} & \text{, if } \chi^-_{r_j,e_i} = 1 \end{cases}$$

These selective assignments can be modeled by the respective Constraints 9.27 to 9.30 of Mixed-Integer Program 9.3, yielding the Δ-model for the TVNEP. The feasibility of the allocations over all states is guaranteed by computing the respective allocations at each state (cf. Constraint 9.31) and enforcing that these allocations respect the resource capacities by the allocation variable bounds (cf. Constraint 9.33).

Mixed-Integer Program 9.3: Δ-Model for the TVNEP

$$\text{Core Model: Constraints and Variables (9.11) to (9.25)} \tag{9.26}$$
$$\Delta^x_{e_i} \leq +a^x_r + d_S(x) \cdot (1 - \chi^+_{r,e_i}) \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E}, x \in G_S \tag{9.27}$$
$$\Delta^x_{e_i} \geq +a^x_r - d_S(x) \cdot (1 - \chi^+_{r,e_i}) \cdot 2 \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E}, x \in G_S \tag{9.28}$$
$$\Delta^x_{e_i} \leq -a^x_r + d_S(x) \cdot (1 - \chi^-_{r,e_i}) \cdot 2 \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E}, x \in G_S \tag{9.29}$$
$$\Delta^x_{e_i} \geq -a^x_r - d_S(x) \cdot (1 - \chi^-_{r,e_i}) \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E}, x \in G_S \tag{9.30}$$
$$\sum_{e_j \in \mathcal{E}: j \leq i} \Delta^x_{e_j} = a^x_{s_i} \qquad \forall x \in G_S, s_i \in \mathcal{S} \tag{9.31}$$
$$\Delta^x_{e_i} \in \mathbb{R}_{\geq 0} \qquad \forall e_i \in \mathcal{E}, x \in G_S \tag{9.32}$$
$$a^x_{s_i} \in [0, d_S(x)] \qquad \forall s_i \in \mathcal{S}, x \in G_S \tag{9.33}$$

Mixed-Integer Program 9.4: Σ-Model for the TVNEP

$$\text{Core Model: Constraints and Variables (9.11) to (9.25)} \tag{9.34}$$
$$\chi^\Sigma_{r,e_i} = \sum_{j=1,\dots,i} \chi^+_{r,e_j} - \sum_{j=1,\dots,i} \chi^-_{r,e_j} \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E} \tag{9.35}$$
$$a^x_{r,s_i} \geq a^x_r - d_S(x) \cdot (1 - \chi^\Sigma_{r,e_i}) \qquad \forall r \in \mathcal{R}, s_i \in \mathcal{S}, x \in G_S \tag{9.36}$$
$$a^x_{s_i} = \sum_{r \in \mathcal{R}} a^x_{r,s_i} \qquad \forall s_i \in \mathcal{S}, x \in G_S \tag{9.37}$$
$$\chi^\Sigma_{r,e_i} \in [0,1] \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E} \tag{9.38}$$
$$a^x_{r,s_i} \in \mathbb{R}_{\geq 0} \qquad \forall r \in \mathcal{R}, s_i \in \mathcal{S}, x \in G_S \tag{9.39}$$
$$a^x_{s_i} \in [0, d_S(x)] \qquad \forall s_i \in \mathcal{S}, x \in G_S \tag{9.40}$$

As shown in the computational evaluation, the utilization of these constraint types yields very weak Linear Programming relaxations. In the following we discuss this weakness using an insightful example.

In the branch-and-bound process used to solve MIP formulations, non-fixed binary variables can attain any value in the range $[0, 1]$. Assume now that, besides other requests, two long lasting requests r_j, $j \in \{1, 2\}$ consisting of single nodes ($V_{r_j} = \{i_j\}$), are to be embedded on a single node substrate ($V_S = \{u\}$) and that both require all resources on this node ($d_{r_j}(i_j) = d_S(u)$). Clearly, such an embedding is not possible if r_1 and r_2 overlap temporally. The relaxation of the event mapping variables however allows for the following setting: $\chi^+_{r_1,e_l} = \chi^+_{r_2,e_l} = 0.5$ for $l \in \{1, 2\}$. Under this assignment, the above Constraints (9.27) and (9.28) reduce to $0 \leq \Delta^u_{e_l} \leq d_S(u)$ for $l \in \{1, 2\}$. Thus, in the relaxation the variables $\Delta^u_{e_j}$ can attain the value 0 and therefore no state allocations for the requests r_j, $j \in \{1, 2\}$, will ever become visible in the substrate's state. Furthermore, considering the mapping of the requests' ends, we note that when $\chi^-_{r_j,e_l} = \chi^-_{r_j,e_{l+1}} = 0.5$ holds for $j \in \{1, 2\}$ and some arbitrary l, then the constraints (9.29) and (9.30) reduce to $-d_S(u) \leq \Delta^u_{e_l} \leq 0$. Thus, all previous resource allocations accounted for at state s_l on node u can effectively be nullified.

The above consideration motivates our next approach, namely, the explicit state representation of the Σ-model.

9.1.3 Σ-Model: Representing States Explicitly

As discussed above, the linear relaxations of the Δ-model may fail to account for any resource allocations previously made. To ensure better relaxations, we will now present the Σ-model that explicitly represents states at the cost of introducing $\mathcal{O}(|\mathcal{S}| \cdot |\mathcal{R}|)$ additional variables to represent local state allocations made by each request. In Section 9.2 we will then show how the state-space can be effectively reduced.

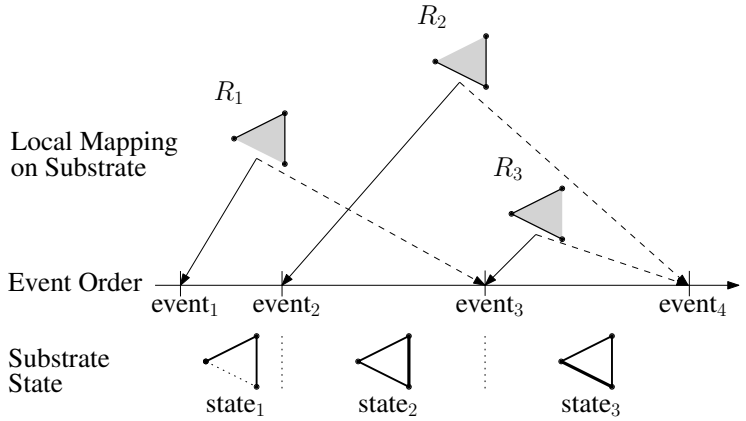


Figure 9.2: The example of Figure 9.1 revisited. By using only $|\mathcal{R}|+1$ many event points and allowing to map multiple requests' ends on events, the state model can be compactified.

To compute the local allocations, we introduce the variables $\vec{\chi}^{\Sigma}$ (cf. Constraints 9.35 and 9.38). Intuitively, χ^{Σ}_{r,e_i} denotes for each event point $e_i \in \mathcal{E}$ *to which extent* request $r \in \mathcal{R}$ is embedded at the respective point in time. This allows us to compute resource allocations *per request* as defined in Constraint 9.36: if request $r \in \mathcal{R}$ is not embedded at event e_i, i.e., $\chi^{\Sigma}_{r,e_i} = 0$, then the local state allocations $\vec{a}$ are not constrained and can be set to 0, while if $\chi^{\Sigma}_{r,e_i} = 1$ holds, the local state allocations are lower bounded by the actual resource usage. Finally, the Constraint 9.37 together with the variable bounds (cf. Constraint 9.40) guarantees feasibility of all state allocations.

Revisiting the single substrate example from the above section, we note that by the above model $\chi^{\Sigma}_{r_j,e_l} = 1$ will hold for $i = \{1, 2\}$ and $k = \{3, \ldots, 100\}$, implying that $a^x_{r_i,s_k} \geq 2 \cdot d_S(x)$ holds. Hence, the respective variable setting yields an infeasible relaxation. The Σ-model is therefore provably stronger than the Δ-model, as it excludes linear relaxation that are feasible for the Δ-model. Since the superiority of this model is based on the ability to derive state allocation directly using $\vec{\chi}^{\Sigma}$, we refer to this model as Σ-model.

9.2 The Compact State Model cΣ

Based on the Σ-model discussed in the previous section, we now present an optimized version of it, the cΣ-model (cf. Mixed-Integer Program 9.5). Concretely, the cΣ-model employs only $|\mathcal{R}| + 1$ event points compared to the $2 \cdot |\mathcal{R}|$ event points used in the Δ- and Σ-model, which not only reduces the state-space significantly but also acts as symmetry reduction. Furthermore, we introduce temporal dependency graph cuts to strengthen the cΣ-model.

9.2.1 Compactification

The major enhancement of the cΣ- over the Σ-model lies in the following observation: to check whether a state is feasible, it suffices to consider *only* the states originating from the start of a request. The truth of this is immediate, as the ending of a request may only reduce state allocations. Figure 9.2 visualizes this new approach using the same example as of Figure 9.1. In this new event point model each start of a request must be assigned uniquely to one event point. As the ordering of the ends of requests does not change the feasibility, we allow for the assignment of multiple requests' ends to the same event point. The semantic of our model will be the following: if the end of a request is mapped onto an event point e_i, then it must have ended between event points e_{i-1} and e_i.

Mixed-Integer Program 9.5: cΣ-Model for the TVNEP

$$\text{Constraints and Variables (9.1) - (9.10)} \tag{9.41}$$

$$1 = \sum_{e_i \in \mathcal{E}: i < |\mathcal{E}|} \chi^+_{r,e_i} \qquad \forall r \in \mathcal{R} \tag{9.42}$$

$$1 = \sum_{e_i \in \mathcal{E}: i > 1} \chi^-_{r,e_i} \qquad \forall r \in \mathcal{R} \tag{9.43}$$

$$1 = \sum_{r \in \mathcal{R}} \chi^+_{r,e_i} \qquad \forall e_i \in \mathcal{E} : i < |\mathcal{E}| \tag{9.44}$$

$$t^+_r \leq t_{e_i} + (1 - \sum_{j=1,\dots,i} \chi^+_{r,e_j}) \cdot \mathcal{T} \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E} \tag{9.45}$$

$$t^+_r \geq t_{e_i} - (1 - \sum_{j=i,\dots,|\mathcal{E}|} \chi^+_{r,e_j}) \cdot \mathcal{T} \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E} \tag{9.46}$$

$$t^-_r \leq t_{e_i} + (1 - \sum_{j=1,\dots,i} \chi^-_{r,e_j}) \cdot \mathcal{T} \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E} : i > 1 \tag{9.47}$$

$$t^-_r \geq t_{e_{i-1}} - (1 - \sum_{j=i,\dots,|\mathcal{E}|} \chi^-_{r,e_j}) \cdot \mathcal{T} \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E} : i > 1 \tag{9.48}$$

$$t_{e_i} \leq t_{e_{i+1}} \qquad \forall e_i \in \mathcal{E} : i < |\mathcal{E}| \tag{9.49}$$

$$t^d_r = t^-_r - t^+_r \qquad \forall r \in \mathcal{R} \tag{9.50}$$

$$\chi^\Sigma_{r,e_i} = \sum_{j=1,\dots,i} \chi^+_{r,e_j} - \sum_{j=1,\dots,i} \chi^-_{r,e_j} \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E} \tag{9.51}$$

$$a^x_{r,s_i} \geq a^x_r - d_S(x) \cdot (1 - \chi^\Sigma_{r,e_i}) \qquad \forall r \in \mathcal{R}, s_i \in \mathcal{S}, x \in G_S \tag{9.52}$$

$$a^x_{s_i} = \sum_{r \in \mathcal{R}} a^x_{r,s_i} \qquad \forall s_i \in \mathcal{S}, x \in G_S \tag{9.53}$$

$$1 = \sum_{i=|\mathsf{dist}^+_{\max}(v)|+1}^{|\mathcal{R}|+1-|\mathsf{dist}^-_{\max}(v)|} \chi(e_i, v) \qquad \forall v \in V_{\mathsf{dep}} \tag{9.54}^\star$$

$$0 \leq \sum_{j=1}^{i-\mathsf{dist}_{\max}(v,w)} \chi(e_j, v) - \sum_{j=1}^{i} \chi(e_j, w) \qquad \begin{array}{l} \forall v \in V_{\mathsf{dep}}, w \in \mathsf{dist}^+_{\max}(v), e_i \in \mathcal{E} : \\ \quad \mathsf{dist}_{\max}(v,w) + 1 \leq i \leq |\mathcal{E}| \end{array} \tag{9.55}^\star$$

$$\chi^+_{r,e_i} \in \{0,1\} \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E} \tag{9.56}$$

$$\chi^-_{r,e_i} \in \{0,1\} \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E} \tag{9.57}$$

$$t_{e_i} \in [0, \mathcal{T}] \qquad \forall e_i \in \mathcal{E} \tag{9.58}$$

$$t^+_r \in [0, \mathcal{T}] \qquad \forall r \in \mathcal{R} \tag{9.59}$$

$$t^-_r \in [0, \mathcal{T}] \qquad \forall r \in \mathcal{R} \tag{9.60}$$

$$\chi^\Sigma_{r,e_i} \in [0,1] \qquad \forall r \in \mathcal{R}, e_i \in \mathcal{E} \tag{9.61}$$

$$a^x_{r,s_i} \in \mathbb{R}_{\geq 0} \qquad \forall r \in \mathcal{R}, s_i \in \mathcal{S}, x \in G_S \tag{9.62}$$

$$a^x_{s_i} \in [0, d_S(x)] \qquad s_i \in \mathcal{S}, x \in G_S \tag{9.63}$$

Accordingly, for the cΣ-model, the event set $\mathcal{E} = \{e_1, \dots, e_{|\mathcal{R}|+1}\}$ together with the state set $\mathcal{S} = \{s_1, \dots, s_{|\mathcal{R}|}\}$ is employed and the mapping to event points is changed accordingly (cf. Constraints 9.42 to 9.44). The respective constraints now state that the request starts must be bijectively mapped on all but the last event point (Constraint 9.42 and 9.44), while the requests' end must be mapped on any of the event points except the first one (cf. Constraint 9.43).

Due to the changed event point semantics, slight modifications for the computation of the request end times are necessitated. While Constraints 9.45 to 9.47 are essentially identical to the ones of the Σ-model, the Constraint 9.48 now lower bounds the end time of a request by the time of the *previous* event point.

The Constraints 9.49 to 9.53 are the same as in the Σ-model (cf. MIP 9.4), while the Constraints 9.54 and 9.55 are *valid inequalities*: while not necessary for the correctness of the model, they strengthen it as shown in the next section.

9.2.2 Temporal Dependency Graph Cuts

Considering the cΣ-model presented above, one finds that both temporal as well as state allocation variables are only constrained using the event mapping variables $\vec{\chi}^+, \vec{\chi}^-$. To yield good LP relaxations, the event mapping variables should therefore be as less *smeared* as possible, i.e., that requests' starts and ends should be assigned fractionally only to as few event points as possible. In this section, we therefore introduce *Temporal Dependency Graph Cuts*: these cuts are valid constraints that can be a priori derived from the requests' temporal specification and strengthen the model.

We introduce the directed temporal dependency graph $G_{\mathsf{dep}}(\mathcal{R}) = (V_{\mathsf{dep}}, E_{\mathsf{dep}})$ that will reflect *temporal* dependencies. We define the graph as follows. The set of nodes represents the set of *abstract* start and end points for each request: $V_{\mathsf{dep}} = \mathcal{R} \times \{\mathsf{start}, \mathsf{end}\}$. We define the following functions to help in calculating the earliest possible start and the latest possible end time.

$$\mathsf{earliest}((r, \mathsf{t}) \in V_{\mathsf{dep}}) = \begin{cases} t_R^s & , \text{if } \mathsf{t} = \mathsf{start} \\ t_r^s + t_r^d & , \text{if } \mathsf{t} = \mathsf{end} \end{cases}$$

$$\mathsf{latest}((r, \mathsf{t}) \in V_{\mathsf{dep}}) = \begin{cases} t_r^e - t_r^d & , \text{if } \mathsf{t} = \mathsf{start} \\ t_r^e & , \text{if } \mathsf{t} = \mathsf{end} \end{cases}$$

A directed edge $(v, w) \in V_{\mathsf{dep}} \times V_{\mathsf{dep}}$ will be contained in E_{dep} iff. v *must* take place *before* w:

$$E_{\mathsf{dep}} = \{(v, w) \in V_{\mathsf{dep}} \times V_{\mathsf{dep}} \mid \mathsf{latest}(v) < \mathsf{earliest}(w)\}.$$

Having defined the dependency graph, we can now start to derive cuts, i.e., valid inequalities. First convince yourself that $G_{\mathsf{dep}}(\mathcal{R})$ is acyclic. For $(v, w) \in E_{\mathsf{dep}}$, we define the weight of (v, w) to be 1, if v represents the start of a request and 0 otherwise. As $G_{\mathsf{dep}}(\mathcal{R})$ is acyclic, we can compute maximal distances by negating the weights and applying the Floyd-Warshall algorithm [Cor+09]. Let $\mathsf{dist}_{\max} : V_{\mathsf{dep}} \times V_{\mathsf{dep}} \to \mathbb{N}$ denote the maximal distances between any two nodes. We set $\mathsf{dist}_{\max}(v, w) = 0$ if w is not reachable from v in $G_{\mathsf{dep}}(\mathcal{R})$. We make the following observations:

1. If a node $v \in V_{\mathsf{dep}}$ *is reachable* from n many nodes of the signature (r_i, start), then v cannot be mapped on either one of the first n event points.
2. Similarly, if a node $v \in V_{\mathsf{dep}}$ *reaches* n other nodes of the signature (r_i, start), then all these must occur *after* v. Furthermore, if v itself is a start event, then its corresponding end event must be mapped after the start. Therefore, if $v = (r, \mathsf{start})$, then v cannot be mapped on the last $n+1$ events and otherwise, if $v = (r, \mathsf{end})$ then v cannot be mapped on the last n events.

3. Let $v, w \in V_{\mathsf{dep}}$, such that $0 < \mathsf{dist}_{\max}(v, w) = d$ holds. Assuming that w is mapped on e_i then v must be mapped on $\{e_1, ..., e_{i-d}\}$.

Note that the first two above observations can be formulated in the following graph theoretical way: given a node $v \in V_{\mathsf{dep}}$ consider the maximal subgraph (with respect to nodes and edges) of G_{dep} such that all nodes within this subgraph reach v or can be reached by v respectively. Counting the number of edges attributed with 1 then directly gives the number of leading and trailing event points on which the request of v cannot be embedded. We denote these values by $\mathsf{dist}^+_{\max}(v)$ and $\mathsf{dist}^-_{\max}(v)$, respectively. To formalize the temporal dependency cuts, we additionally introduce the following function to obtain the respective event mapping variable for a temporal dependency node.

$$\chi(e_i \in \mathcal{E}, (r, \mathsf{t}) \in V_{\mathsf{dep}}) = \begin{cases} \chi^+_{r,e_i} & \text{if } \mathsf{t} = \mathsf{start} \\ \chi^-_{r,e_i} & \text{if } \mathsf{t} = \mathsf{end} \end{cases}$$

Constraint 9.54 effectively restricts the mapping of the start and end of requests to a limited range given the number of event points that must precede and follow it, respectively. The constraint can also effect further state-space reductions as follows. Let $\mathcal{E}^+_r = \{e_i, ..., e_j\}$ and $\mathcal{E}^-_r = \{e_{j+k}, ..., e_l\}$ denote the event ranges for the start and end of r respectively with $k > 0$. Clearly, $\chi^{\Sigma}_{r,e_n} = 1$ holds for $i \leq n < j + k$ (cf. Constraint 9.51). Thus, the state allocations $\vec{a}_{r,s_n}$ must equal the respective time-invariant allocations $\vec{a}_r$ as computed by Constraints 9.5 and 9.6 and the respective state variables can be removed.

Constraint 9.55 effectively reduces variable smearings by implementing the intuition of point (3) above. Concretely, it considers a node $w \in V_{\mathsf{dep}}$ that must occur at least $\mathsf{dist}_{\max}(v, w)$ event points before $v \in V_{\mathsf{dep}}$. Considering an event point e_i, the subtrahend computes the extent to which w was mapped until (and including) it. As v must precede w by at least $\mathsf{dist}_{\max}(v, w)$ many event points, v must be mapped accordingly at least to the same extent on the events $\{e_1, \ldots, e_{i-\mathsf{dist}_{\max}(v,w)}\}$.

9.2.3 Symmetry Reductions

We will now show that the compactification of the Σ-model into the cΣ-model also represents a symmetry reduction. Consider the scenario of k requests $\mathcal{R} = \{r_1, \ldots, r_k\}$ of duration $t^d_i = 1 + 1/2^i$ for $1 \leq i \leq k$ which are to be embedded in the interval $[0, 2]$. Clearly, a priori, the only possible event order is to first start all requests and then end them (this is indeed implied by the temporal dependency graph cuts). We will now show that the number of possible solutions can be reduced by a factor of 2^k using the cΣ-model in comparison to the Δ- and Σ-model. Assume that the requests' starts are assigned in the order of ascending durations: $\chi^+_{r_i,e_i} = 1$. Based on the chosen durations, request r_{i+1} can either be scheduled to complete before r_i by setting $t^-_{r_{i+1}} = t^-_{r_i} - 1/2^{k+1}$ or after r_i by setting $t^-_{r_{i+1}} = t^-_{r_i} + 1/2^{k+1}$. Importantly, setting these times does not interfere with the assumption that the start of request r_i is to be mapped on event point e_i. Therefore, having fixed the requests' start variables, there might be as much as 2^k many possible combinations to order the requests' ends. In contrast, the cΣ-model allows only for a single solution, in which all requests' ends are mapped on the last event point, hence showing that the cΣ-model indeed reduces symmetries.

9.3 Objective Functions

We will now present the objective functions used in the evaluation of our approach.

Access Control. The task is to maximize the revenue of the substrate's provider. We choose the revenue to be the sum of requested virtual node resources over time:

$$\max \sum_{r \in \mathcal{R}} x_r \cdot t_r^d \cdot \sum_{i \in V_r} d_r(i)$$

Maximize Earliness. Given a fixed set of requests to embed, the provider may charge an additional fee depending on how soon the request is embedded. We assume that this fee is proportional to the duration of the request, such that if the request is started at the earliest possible time, the fee is t_r^d and if the request starts at the latest possible time, the fee is 0.

$$\max \sum_{r \in \mathcal{R}} t_r^d \cdot (1 - \frac{t_r^+ - t_r^s}{t_r^e - t_r^s - t_r^d})$$

Balance Node Load over Time. Given a fixed set of requests to embed, the provider may try to schedule and embed the given requests in such a way, that during the span of time under consideration, the number of nodes never being used more than some fraction of their capacity is maximized. To this end, we introduce additional binary variables $F_u \in \{0,1\}$ for $u \in V_S$, deciding whether a substrate node is never used more than $\mathsf{f}\,\%$ of its capacity, for $\mathsf{f} \geq 0$.

$$\max \sum_{u \in V_S} F_u \,, \text{subject to}$$
$$(1 - F_u) \cdot (1 - \mathrm{f}) \cdot d_S(u) \geq a_{s_i}^u - \mathsf{f} \cdot d_S(u) \quad \forall u \in V_S, s_i \in \mathcal{S}$$

Disable Links for Energy Savings. Given a fixed set of requests to embed, another goal one may pursue with is the conservation of energy. Concretely, one may want to turn off as many links (i.e., switch or router ports) as possible for as long as possible [See10]. To model this objective function, we introduce decision variables $D_{u,v} \in \{0,1\}$ for $(u,v) \in E_S$, such that $D_{u,v} = 1$ iff. (u,v) can be disabled over the whole time span $[0, \mathcal{T}]$.

$$\max \sum_{(u,v) \in E_S} D_{u,v} \,, \text{subject to}$$
$$a_{s_i}^{u,v} \leq d_S(u,v) \cdot (1 - D_{u,v}) \quad \forall (u,v) \in E_S, s_i \in \mathcal{S}$$

9.4 Computational Evaluation

This section presents our computational evaluation of the algorithms to solve TVNEP, and focuses on the following questions: (1) How do the Δ, Σ and cΣ formulations perform in comparison? (2) To which extent can we compute optimal solutions to the TVNEP using the cΣ formulation? (3) What are the *benefits* of allowing for temporal flexibility?

As underlying scenario we have chosen a synthetic workload on a data-center topology. The task is to solve the TVNEP for a day of work, represented by twenty requests spread over the day. We consider twenty-four of such workloads independently to allow for qualitative conclusions. As our main focus is how well our algorithms can cope with temporal flexibilities, we increment the temporal flexibility (initially there are none) in steps of 30 minutes until each request is equipped with a temporal flexibility of 300 minutes (5 hours). For each of the resulting $24 \times 11 = 264$ scenarios, we compute solutions using the Δ-, Σ-, and the cΣ-model with the access control objective.

After having presented the results under the access control objective, the performance of the cΣ-model under the three other objectives presented above is discussed.

9.4.1 Methodology

As substrate for our experiments we have chosen a directed 4×5 grid graph, with $|V_S| = 20$ nodes and $|E_S| = 62$ directed edges. Capacities on the substrate are set to be 3.5 for nodes and to 5 for links. As request topologies, we use five node stars consisting of a single center and four surrounding nodes, such that either all links are directed towards the center or away from the center. This topology may represent a classical master-slave relationship, or a Virtual Cluster. The requests' required resources are chosen uniformly at random from the interval $[1, 2]$, such that with high probability only two nodes can be mapped on the same substrate node. We generate 20 requests from a Poisson arrival process with exponentially distributed inter-arrival time of 1 hour. The duration of requests is sampled from the (heavy-tailed) Weibull distribution with shape parameter two and scale parameter four, giving an expected duration of approximately 3.5 hours. As our focus in this work is to compare different continuous-time models, we fix node mappings a priori: for each virtual node we select uniformly at random a substrate node on which this node is to be embedded. Importantly however, the virtual links are not fixed beforehand; our algorithms therefore do not only need to find a subset of requests to embed, but furthermore need to decide when these requests are to be scheduled, compute the embeddings of the virtual links and guarantee that no substrate capacities are exceeded.

All experiments were conducted on Intel Xeon L5420@2,5 Ghz servers with 8 GB of RAM using Gurobi 5.60. Experiments were terminated after one hour of execution. All model and data files as well as the logs are available at [RS13].

9.4.2 Results

Access Control Objective. We start by comparing the different MIP formulations. Figure 9.3 depicts the runtime of the Δ-, Σ-, and cΣ-model as a function of the time flexibility. As we terminate experiments after one hour of execution, Figure 9.4 shows the *objective gap* (the relative difference between upper and lower bound in the branch-and-bound process) after one hour. First, note that the Δ-model is unable to generate solutions already for 90 minutes of flexibility, as only for 1 of the 24 scenarios, a solution is found. Considering the Σ- and cΣ-model, both always yield feasible solutions. However, the runtime as well as the gaps of the cΣ-model are on average one magnitude lower than for the Σ-model. We therefore conclude that our optimizations presented in Section 9.2 significantly reduce the runtime and improve the quality of solutions: the runtime of the cΣ-model lies one order of magnitude below the runtime of the Σ-model until 210 minutes of flexibility, where the runtime of the cΣ-model also approaches the time limit, and the objective gap is approximately one order of magnitude less when using the cΣ-model.

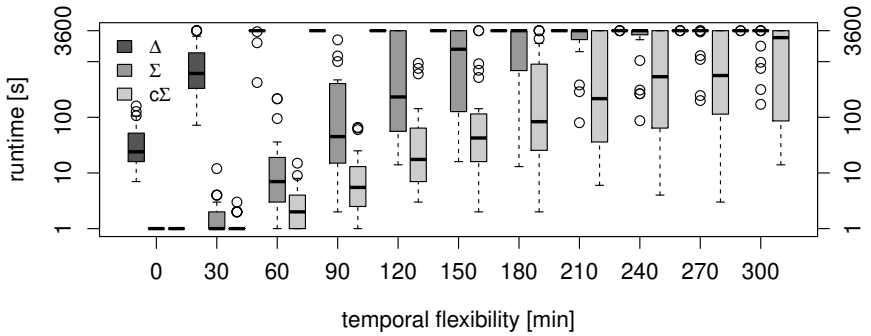


Figure 9.3: Runtime of MIP formulations as a function of the time flexibility. As computations were terminated after one hour, a runtime of 3600 implies, that no optimal solution has been found. Note the logarithmic y-axis.

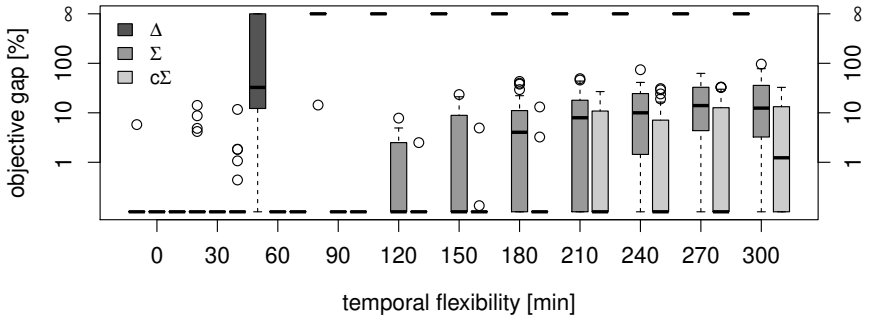


Figure 9.4: Objective gap of the formulations as a function of the time flexibility. The value ∞ denotes that not a single solution was found. Note the logarithmic y-axis.

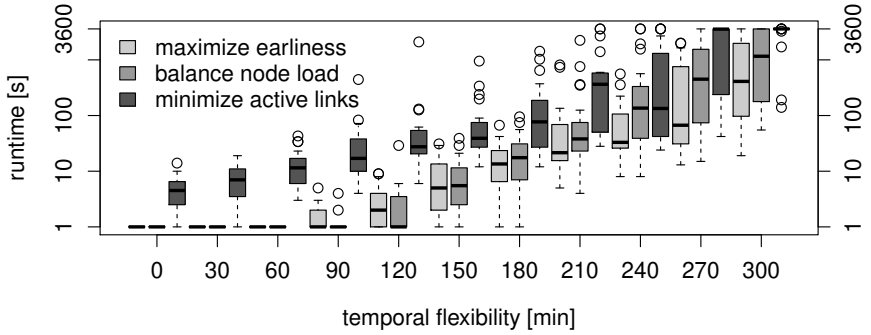


Figure 9.5: Runtime of the cΣ-model under the objectives presented in Section 9.3. Note the log. y-axis.

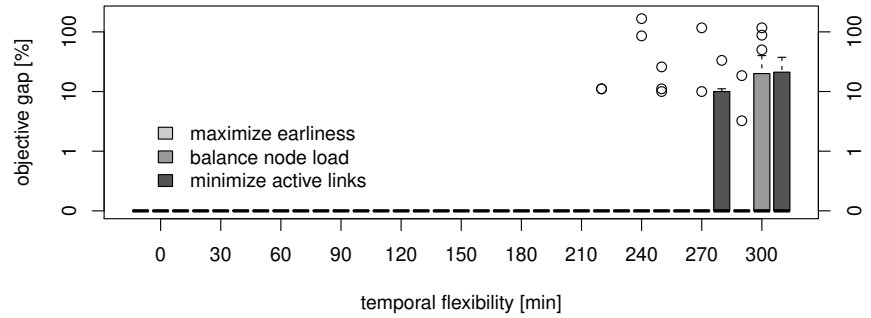


Figure 9.6: Gap of the cΣ-model under the objectives presented in Section 9.3.

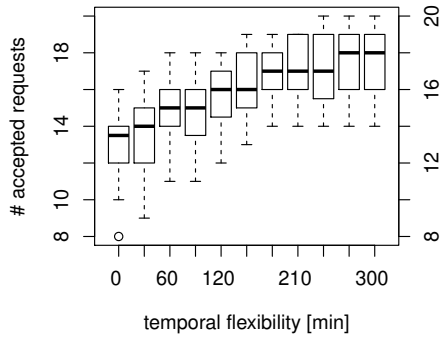


Figure 9.7: Number of requests embedded by the cΣ-model.

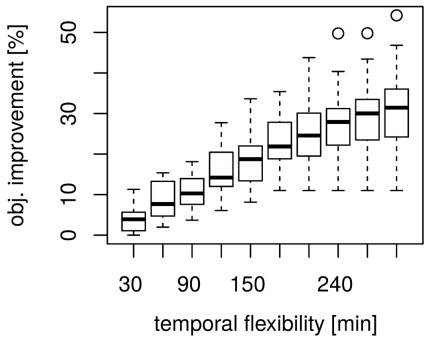


Figure 9.8: Relative improvement of the access control objective compared with the objective at flexibility 0.

Performance of cΣ-Model under Different Objectives. We will now consider the performance of the cΣ-model under objectives not containing access control, but for maximizing earliness, the number of free nodes and the number of disabled links. To this end, we reuse the results from the access control computation (from the cΣ-model) and remove all requests that could not be embedded while forcing the other requests to be embedded. Figure 9.7 depicts a boxplot of the number of embedded requests for the different temporal flexibilities.

Figures 9.5 and 9.6 present the runtime and objective gap respectively. While the objective to maximize energy savings by disabling links seems to be the hardest of these, note that only at temporal flexibilities of 270 and 300 minutes, many scenarios did not yield optimal solutions within one hour of execution time. Furthermore, considering the runtime, optimal solutions for all three objectives can be computed within close to 3 minutes up to a temporal flexibility of 210 minutes, which equals the expected duration of the requests. Note that for the objective of balancing the node load the parameter f was set to $2/3.5 = 4/7$, such that a node was considered 'balanced' when the resource allocations do not exceed 2 at any point in time. As for virtual nodes the demands are chosen uniformly at random from the range $[1, 2]$, this means, that at any point in time at most one virtual node is mapped onto any substrate node.

9.5 Summary and Novelty of Contributions

As today's networks are becoming more virtualized, enabling a flexible service allocation and resource sharing, it is important that a predictable performance is ensured by resource isolation. Virtual networks can provide such guarantees. However, for an optimal resource allocation, virtual network requests should not only be mapped in a flexible and efficient manner, but temporal flexibilities should be harnessed.

Accordingly, within this chapter several Mixed-Integer Programs to solve the TVNEP to (near-)optimality were presented. In particular, the cΣ-model was presented that incorporates state-space and symmetry reductions. Furthermore, valid inequalities called temporal dependency graph cuts were introduced that additionally strengthen the model. The presented approach is universal in the sense that it can be easily adapted to cater for different objectives or additional constraints. In fact, in our technical report [RS13] we also introduce time-dependent virtual networks whose resource allocation rate does depend on the request's duration. This sort of abstraction might be especially useful to realize delay-tolerant background data transfers [Lao+09].

With respect to the novelty of our approach, we note that this work is the first one to study scheduling aspects for the VNEP. Furthermore, our formulations differ vastly from related work. Specifically, also in other fields with time-critical applications, mathematical programming solutions have been proposed to solve these. In particular, the spatio-temporal composition of distributed multimedia objects is considered in [Shi+98], and more remotely similar tasks arise in chemical production planning [LF01]. Indeed, several concepts introduced in the chemical production literature also apply to the VNEP [FL04]. However, the focus in chemical production planning lies on the *sequential* planning of production *processes*, where tasks can only be performed after the completion of predecessor tasks whose output is needed as input; time and specification flexibilities play only a secondary role. We are not aware of any results on more complex objective functions with time-dependent variables, whose state (e.g., resource allocations) and state differences needs to be tracked over time, as it is achieved with our approach.

"It is not enough to be in the right place at the right time.
You should also have an open mind at the right time."
– Pál Erdős

10 Concluding Remarks

The advent of (network) virtualization has allowed to share both computational and network resources. To provide performance guarantees for abstractions as virtual networks, several new resource allocation problems arose. Among these, the Virtual Network Embedding Problem plays a central role. However, given both the flexibility in mapping nodes and routing flows, solving the VNEP (and its siblings) is highly challenging. This thesis has provided a broad theoretical foundation for the VNEP while also bridging the gap towards practical applications. In the following we first summarize our major findings and then briefly discuss future work.

10.1 Summary

We have firstly given a comprehensive set of hardness results for the VNEP, showing that the VNEP is $\mathcal{NP}$-complete and inapproximable (unless $\mathcal{P}=\mathcal{NP}$) under any combination of node and edge mapping restrictions. Furthermore, we have shown that these hardness results also pertain to computing approximate embeddings and still hold when restricting request graphs to be planar. The latter result paved the way for an important insight: to theoretically capture the mapping flexibility introduced by the VNEP the structure of the request graphs has to be taken into account. This lead us to the parametrized Linear Programming formulations presented in Chapter 6.

Specifically, in Chapter 6 several LP formulations to solve the fractional offline VNEP were presented. Starting by analyzing the classical MCF formulation, it was shown to yield optimal solutions for the fractional VNEP only for tree request graphs. In contrast, for cyclic requests, the MCF formulation generally yields no meaningful results. This, in turn, led to novel Linear Programming formulations to capture the inherent structure of the request graphs: one formulation based on the novel notion of extraction width and one based on the well-known notion of treewidth, with the latter being more promising. Specifically, based on dynamic programming on tree decompositions, the efficient XP-time algorithm DynVMP to solve the Valid Mapping Problem was proposed. Applying the parametrized DynVMP algorithm as separation oracle, the first XP-time algorithms for solving the fractional offline VNEP were obtained. Even more, extending the DynVMP algorithm to perform the separation approximately when considering latencies, also the first XP-approximations for the fractional offline VNEP in the presence of latency constraints were obtained.

The ability to solve the fractional offline VNEP has then been proven in Chapter 7 to be sufficient to obtain the first XP-time approximation algorithms for the profit and cost offline VNEP under any combination of mapping restrictions. Our approximations provide strict *quality* guarantees while their runtime is parametrized by either the extraction width or the treewidth. The approximations are obtained by employing the well-known technique of randomized rounding of LP solutions. While the theoretic analysis has shown that the resource violations depend on the maximal allocation to capacity ratio of any valid mapping and might be high in the worst-case, our computational study has shown that employing our heuristical rounding approaches provide a valuable basis for XP-time *heuristics*. Concretely, our

extensive evaluations have shown that heuristical rounding performs well compared to optimal baseline solutions and can outperform greedy heuristics like ViNE consistently and significantly when the demand to capacity ratio is sufficiently small.

In addition to studying the algorithmic foundations of the VNEP, this thesis also presented results on the embedding of virtual clusters and a temporal extension of the VNEP. In particular, for the specific incarnation of virtual clusters, the polynomial-time solvability of the online embedding problem was shown. Beyond this important result, we also proposed a hose-based interpretation of virtual cluster embeddings which bears the potential to significantly reduce bandwidth usage while coming at the cost of being computationally hard to solve. Hence, focusing on the hose model with splittable edge mappings, a polynomial-time heuristic for the splittable hose model was presented and its benefits were thoroughly evaluated.

Lastly, for the scheduling of virtual networks under temporal flexibilities several continuous-time Mixed-Integer Programming models were proposed. Analyzing the shortcomings of the initial model, which only represents state changes, a compact state-based model with additional state-space and symmetry reductions was derived. Our computational evaluation has shown that the latter model is sufficiently strong to compute near-optimal solutions for various objectives, while clearly outperforming the preliminary mathematical programming models.

10.2 Future Work

We believe that this thesis opens up many interesting directions for both theoretical and practical future research.

Considering the computational complexity of the VNEP, we note that our results on approximate embeddings only consider the relaxation of a single type of mapping restriction. no hardness results when relaxing several mapping restrictions at the same time were given. Furthermore, given our parametrized approximation algorithms, a parametrized complexity analysis of the VNEP would be of interest to study to which extent our results can be improved.

The XP-time algorithms to solve or approximate the fractional offline VNEP rely either on the notions of the extraction width or the treewidth. As discussed in Section 6.3.6 the extraction width approach can be extended and may eventually lead back to the notion of treewidth and seems hence less promising. For the column generation oriented treewidth approach, we are confident that the theoretical results can – in general – not be improved significantly. However, refinements of the implementation should be studied to further speed up computations. For example, it would be interesting to study the impact of the number of valid mappings added during the separation process per request. Furthermore, the separation process could be parallelized to scale the separation to significantly more requests.

Our XP-time approximation results are comprehensive as they also pertain to the most general setting $\langle \mathbf{VE} \,|\, \mathbf{NRL} \rangle$ and we believe that the resource augmentation analysis is optimal up to constant factors. Nevertheless, the potentially large resource augmentation factors might impede its practical application. In fact, based on the intriguing results of Bansal et al. [Ban+15], who use a Sherali-Adams inspired hierarchy, the resource augmentation factors could potentially be improved significantly. While directly applying this approach will likely prove impractical due to the resulting formulation sizes, iterative rounding schemes or iterative refinements of an initially small LP formulation might yield similar results for practical applications. Such improved approximations could also pave the way towards competitive online algorithms for the VNEP, which could then rely on much smaller resource augmentation factors for single requests.

Independently of the potential development of such improved approximations, we believe to have already provided an interesting starting point for competitive online algorithms. Specifically, our DynVMP

algorithm is the first algorithm to optimally solve the VMP. As the VNEP reduces to the VMP when the resource demands are small compared to the substrate capacities, the framework for competitive online algorithms of Even et al. [Eve+13] can readily be applied in this case, yielding the first competitive online algorithms for the VNEP in this case. Secondly, even when demands are not small, the DynVMP algorithm always provides lower bounds on the cost of optimal VNEP solutions and this might be used to *heuristically* reject requests according to some cost measure. In addition to the above, we also believe that one might derive well-performing heuristics for the online VNEP from the DynVMP algorithm. For example, one may iteratively compute optimal valid mappings under varying cost measures which penalize resource augmentations, thereby potentially yielding feasible embeddings.

Lastly, we also note that the ability to *approximate* valid mappings under latency constraints will hopefully lead to novel results and applications in the realm of service chaining, where request graphs are naturally of limited complexity while requiring tight latency bounds in many use cases. Here, one interesting extension of our model would be the consideration of latency constraints which span over several virtual edges.

"In review, I feel that I lacked some of the abilities that are important for an applied statistician who has to handle problems on a daily basis. I lacked the library of rough and ready techniques to produce usable results. However, I found that dealing with real applied problems, no matter how unimportant, without this library, required serious consideration of the issues and was often a source of theoretical insight and innovation."

– Herman Chernoff

Derivation of Chernoff Bounds

In the following, probabilistic tail bounds known as Chernoff bounds are derived. Specifically, the following probabilistic upper and lower bounds are derived for the sum $X = \sum_{i\in[N]} X_i$ of $N \in \mathbb{N}$ *independent* random variables with $X_i \in [0,1]$ for $i \in [N]$ and $\mu_i = \mathsf{E}[X_i]$ denoting the expected value.

Theorem A.1. Let X be the sum of N random variables $X_1, \ldots, X_n$ with $X_i \in [0,1]$ for $i \in [N]$. Denoting by $\hat{\mu}_i \geq \mu_i = \mathsf{E}[X_i]$ upper bounds on the expected value of random variable X_i, $i \in [N]$, the following holds for any $\delta > 0$ and $\hat{\mu} = \sum_{i\in[N]} \hat{\mu}_i$:

$$\mathsf{Pr}[X \geq (1+\delta)\cdot\hat{\mu}] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\hat{\mu}} \leq e^{-\delta^2\cdot\hat{\mu}/3}\,. \tag{A.1}$$

Theorem A.2. Let X be the sum of N random variables $X_1, \ldots, X_n$ with $X_i \in [0,1]$ for $i \in [N]$. Denoting by $\tilde{\mu}_i \leq \mu_i = \mathsf{E}[X_i]$ lower bounds on the expected value of random variable X_i, $i \in [N]$, the following holds for any $\delta \in (0,1)$ and $\tilde{\mu} = \sum_{i\in[N]} \tilde{\mu}_i$:

$$\mathsf{Pr}[X \leq (1-\delta)\cdot\tilde{\mu}] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\tilde{\mu}} \leq e^{-\delta^2\cdot\tilde{\mu}/2}\,. \tag{A.2}$$

Theorem A.3. Let $X = \sum_{i\in[N]} X_i$ be the sum of $N \in \mathbb{N}$ *independent* random variables with $X_i \in [0,1]$ and let $\mu_i = \mathsf{E}[X_i]$ denote their respective expected value for $i \in [N]$. Let $\hat{\mu}_i \geq \mu_i$ be an upper bound on the expected value of $X_i \in [0,1]$, $i \in [N]$, and let $\hat{\mu} = \sum_{i\in[N]} \hat{\mu}_i$ such that,

$$\hat{\mu} = \sum_{i\in[N]} \hat{\mu}_i \geq 1\,. \tag{A.3}$$

Denote by c, n, λ, ρ, ξ constants, such that,

$$\begin{aligned} c \geq& 2 && \text{(A.4)}\\ n \geq& 3 && \text{(A.5)}\\ \rho \geq& 2 && \text{(A.6)}\\ \rho\cdot c \geq& e && \text{(A.7)}\\ \xi \geq& \ln\ln n/\ln n && \text{(A.8)}\\ \lambda =& \rho\cdot c\cdot\xi\cdot\ln n/\ln\ln n\,. && \text{(A.9)} \end{aligned}$$

The following holds:

$$\mathsf{Pr}[X \geq \lambda\cdot\hat{\mu}] \leq 1/n^{c\cdot\hat{\mu}\cdot\xi}\,. \tag{A.10}$$

While similar results are contained in any textbook on randomized algorithms [MR95; DP09; WS11; AS16; MU17], most textbooks either do not consider upper and lower bounds of the expectation, leave parts of the proofs to the reader, or state the results only for binary random variables $X_i \in \{0,1\}$, $i \in [N]$.

In Section A.1 some prerequisites for the proofs are given, while the proofs for the above stated Theorems A.1 and A.2 are given in Section A.2 and in Section A.3 Theorem A.3 is proven. Henceforth, our presentation is mostly based on the textbook of Williamson and Shmoys [WS11], while lecture notes of Nick Harvey and Chandra Chekuri were of equal importance to fill in details in the respective proofs.

A.1 Prerequisites

In the following elementary lemmas are proven, which often rely on the analysis of convex functions.

Lemma A.4. The following holds for any $t \in \mathbb{R}$ and $x \in [0,1]$:

$$\exp(t \cdot x) \leq 1 + (\exp(t) - 1)$$

Proof. The function $f(x) = c^x$ is convex as $f''(x) = \ln^2(c) \cdot c^x > 0$ holds for any $c > 0$ and any $x \in \mathbb{R}$. As f is convex, the following holds for any $z \in [x_1, x_2]$

$$f(z) \leq \frac{f(x_2) - f(x_1)}{x_2 - x_1} \cdot (z - x_1) + f(x_1)$$

Setting $c = \exp(t)$, $x_1 = 0$, and $x_2 = 1$, the result is obtained for all $z \in [0,1]$:

$$\exp(t \cdot z) \leq (\exp(t) - 1) \cdot z + 1\,.$$ ■

Lemma A.5. The following holds for any $t \in \mathbb{R}$ and $x \in [0,1]$:

$$1 + (\exp(t) - 1) \cdot x \leq \exp((\exp(t) - 1) \cdot x)$$

Proof. The inequality follows from the observation that $1 + z \leq \exp(z)$ holds for $z \in \mathbb{R}$, which follows from the convexity of $f(z) = \exp(z)$. By substituting z with $(\exp(t) - 1) \cdot x$ the result is obtained for any $t \in \mathbb{R}$ and $x \in [0,1]$. ■

Lemma A.6. Let $X_i \in [0,1]$ denote a single random variable of expectation $\mu_i = \mathsf{E}[X_i]$. For any $t \in \mathbb{R}$, the following holds for the random variable $Y_i = \exp(t \cdot X_i)$:

$$\mathsf{E}[Y_i] \leq \exp((\exp(t) - 1) \cdot \mu_i) \tag{A.11}$$

Proof. As the function $f(x) = \exp(t \cdot x)$ is convex for any $t \in \mathbb{R}$, the following holds:

$$\begin{aligned}
\mathsf{E}[Y_i] =& \mathsf{E}\left[\exp(t \cdot X_i)\right] \\
\leq& \mathsf{E}\left[1 + (\exp(t) - 1) \cdot X_i\right] && \text{(by Lemma A.4)} \\
=& 1 + (\exp(t) - 1) \cdot \mathsf{E}[X_i] && \text{(by linearity of expectation)} \\
\leq& \exp((\exp(t) - 1) \cdot \mathsf{E}[X_i]) && \text{(by Lemma A.5)} \\
=& \exp((\exp(t) - 1) \cdot \mu_i) && \text{(by definition of } \mathsf{E}[X_i] = \mu_i)
\end{aligned}$$

■

Lemma A.7. The following inequality holds for any $x \in [0,1]$:

$$(1 + x)\ln(1 + x) - x \geq x^2/3\,. \tag{A.12}$$

Proof. To start off, we note that the above inequality is satisfied for $x = 0$ and that the inequality is preserved as long as the derivative of the left-hand side is greater than or equal to the derivative of the right-hand side. Considering the derivatives $\frac{d}{dx}\left((1+x)\ln(1+x) - x\right) = \ln(1+x)$ and $\frac{d}{dx}\left(x^2/3\right) = 2/3 \cdot x$, we note again that for $x = 0$ the respective derivatives have the same value. However, as the function $f(x) = \ln(1+x)$ is concave and we have $\ln(2) > 2/3$, the function $f(x)$ must always lie above the line $g(x) = 2/3 \cdot x$ in the interval $[0, 1]$. Hence, as the derivative of the left-hand side is always greater than or equal to the derivative of the right-hand side, the statement follows. ■

Lemma A.8. The following inequality holds for any $x \in [0, 1)$:

$$-x - (1-x)\ln(1-x) \leq -x^2/2\,. \tag{A.13}$$

Proof. We apply an analogous argument as in the proof of Lemma A.7. To start off, we note that the above inequality is satisfied for $x = 0$ and that the inequality is preserved as long as the derivative of the left-hand side is less than the derivative of the right-hand side. Considering the derivatives $\frac{d}{dx}\left(-x - (1-x)\ln(1-x)\right) = \ln(1-x)$ and $\frac{d}{dx}\left(-x^2/2\right) = -x$, we note again that for $x = 0$ the respective derivatives have the same value. Repeating the argument, the derivative of the left-hand side is smaller than the derivative of the right-hand side if the second derivative of the left-hand side is always smaller than the second derivative of the right-hand side. Hence, as $\frac{d^2}{dx}\left(-x - (1-x)\ln(1-x)\right) = -1/(1-x)$, $\frac{d^2}{dx}\left(-x^2/2\right) = -1$, and $-1/(1-x) \leq -1$ holds for $x \in (0, 1)$, the second and first derivatives of the left-hand side are smaller than the respective derivatives of the right-hand side. Hence, Inequality A.13 holds for any $x \in [0, 1)$. ■

Lemma A.9. The following inequality holds for any $x > 0$:

$$\ln\ln x - \ln\ln\ln x \geq 0.5 \cdot \ln\ln x\,. \tag{A.14}$$

Proof. To prove the above inequality, we consider the function $f(z) = 0.5 \cdot z - \ln z$ with $f'(z) = 0.5 - 1/z$ for $z > 0$. Setting $f'(z_0) = 0$, the unique root $z_0 = 2$ is obtained. As $f'(z) \geq 0$ for $z \geq 2$ and $f(z_0) = 1 - \ln(2) > 0$ holds, we obtain that $f(z) > 0$ holds for $x \geq 2$. Furthermore, it is easy to check that $f(z) \geq 0$ holds for $z \in (0, 1)$, as in this case the natural logarithm is negative. Hence, as the function f is continuously differentiable on $(0, \infty)$, and as the function is positive on the intervals $(0, 1)$ and $[2, \infty)$ while f' has a single root at $z_0 = 2$, the function must be always positive on $(0, \infty)$. Thus, $f(z) \geq 0$ holds for $z > 0$.

Substituting $z = \ln\ln x$ then yields $0.5 \cdot \ln\ln x - \ln\ln\ln x \geq 0 \Leftrightarrow \ln\ln x - \ln\ln\ln x \geq 0.5 \cdot \ln\ln x$. ■

A.2 Proofs of Chernoff Bounds

We now prove the correctness of the two Chernoff bounds.

Theorem A.1. Let X be the sum of N random variables $X_1, \ldots, X_n$ with $X_i \in [0, 1]$ for $i \in [N]$. Denoting by $\hat{\mu}_i \geq \mu_i = \mathsf{E}[X_i]$ upper bounds on the expected value of random variable X_i, $i \in [N]$, the following holds for any $\delta > 0$ and $\hat{\mu} = \sum_{i \in [N]} \hat{\mu}_i$:

$$\mathsf{Pr}[X \geq (1+\delta)\cdot\hat{\mu}] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{\hat{\mu}} \leq e^{-\delta^2\cdot\hat{\mu}/3}\,. \tag{A.1}$$

Proof. Let $Y_i = \exp(t \cdot X_i)$ for some value $t \in \mathbb{R}$, $i \in [N]$. As the variables $X_1, \ldots, X_N$ are pairwise independent, the variables Y_i are also pairwise independent. Considering $Y = \exp(t \cdot X)$, the following holds due to the pairwise independence of the variables $Y_1, \ldots, Y_N$:

$$\mathsf{E}[Y] = \mathsf{E}[\exp(t \cdot X)] = \mathsf{E}\left[\exp(t \cdot \sum\nolimits_i X_i)\right] = \mathsf{E}\left[\prod\nolimits_i \exp(t \cdot X_i)\right] = \prod\nolimits_i \mathsf{E}[Y_i]\,. \tag{A.15}$$

By Lemma A.6 the Inequality A.11 holds. Assuming that $t > 0$ holds, the exponential function $f(z) = \exp((\exp(t) - 1) \cdot z)$ is monotonically increasing. Using the upper bound $\hat{\mu}_i \geq \mu_i = \mathsf{E}[X_i]$ on the expectation of X_i, $i \in [N]$, with $\hat{\mu} = \sum_{i \in [N]} \hat{\mu}_i$, the following is obtained:

$$\mathsf{E}[Y_i] \leq \exp((\exp(t) - 1) \cdot \hat{\mu}_i)\,. \tag{A.16}$$

Using Equations A.15 and A.16 together with Markov's inequality, the left inequality of the Chernoff bound A.1 is obtained for any $\delta > 0$ by setting $t = \ln(\delta + 1) > 0$:

$$\begin{aligned}
&\mathsf{Pr}[X \geq (1+\delta) \cdot \hat{\mu}] \\
&\quad = \mathsf{Pr}[\exp(t \cdot X) \geq \exp(t \cdot (1+\delta) \cdot \hat{\mu})] \\
&\quad \leq \frac{\mathsf{E}[\exp(t \cdot X)]}{\exp(t \cdot (1+\delta) \cdot \hat{\mu})} && \text{(by Markov's inequality)} \\
&\quad = \frac{\prod_{i \in [N]} \mathsf{E}[Y_i]}{\exp(t \cdot (1+\delta) \cdot \hat{\mu})} && \text{(by Equation A.15)} \\
&\quad \leq \frac{\prod_{i \in [N]} \exp((\exp(t) - 1) \cdot \hat{\mu}_i)}{\exp(t \cdot (1+\delta) \cdot \hat{\mu})} && \text{(by Equation A.16)} \\
&\quad = \exp\left(\Big(\sum_{i \in [N]} (\exp(t) - 1) \cdot \hat{\mu}_i \Big) - \big(t \cdot (1+\delta) \cdot \hat{\mu} \big) \right) && \text{(one exponent)} \\
&\quad = \exp\left(\Big(\sum_{i \in [N]} ((1+\delta) - 1) \cdot \hat{\mu}_i \Big) - \big(\ln(1+\delta) \cdot (1+\delta) \cdot \hat{\mu} \big) \right) && (t = \ln(\delta+1)) \\
&\quad = \exp\left(\big(\delta \cdot \hat{\mu} \big) - \big(\ln(1+\delta) \cdot (1+\delta) \cdot \hat{\mu} \big) \right) && \text{(definition of } \hat{\mu} = \textstyle\sum_{i \in [N]} \hat{\mu}_i) \\
&\quad = \frac{\exp\left(\delta \cdot \hat{\mu} \right)}{\exp\left(\ln(1+\delta) \cdot (1+\delta) \cdot \hat{\mu} \right)} \\
&\quad = \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}} \right)^{\hat{\mu}}
\end{aligned}$$

This completes the proof of the first inequality. The second inequality, namely $\left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}} \right)^{\tilde{\mu}} \leq e^{\delta^2 \cdot \tilde{\mu}/3}$, is a corollary of Lemma A.7. Specifically, by Lemma A.7 the following holds for $\delta \in [0, 1]$

$$x - (1+\delta)\ln(1+\delta) \leq -\delta^2/3\,. \tag{A.17}$$

Multiplying both sides with $\tilde{\mu}$ and exponentiating both sides yields the desired result. ■

Theorem A.2. Let X be the sum of N random variables $X_1, \ldots, X_n$ with $X_i \in [0, 1]$ for $i \in [N]$. Denoting by $\tilde{\mu}_i \leq \mu_i = \mathsf{E}[X_i]$ lower bounds on the expected value of random variable X_i, $i \in [N]$, the following holds for any $\delta \in (0, 1)$ and $\tilde{\mu} = \sum_{i \in [N]} \tilde{\mu}_i$:

$$\mathsf{Pr}[X \leq (1-\delta) \cdot \tilde{\mu}] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}} \right)^{\tilde{\mu}} \leq e^{-\delta^2 \cdot \tilde{\mu}/2}\,. \tag{A.2}$$

Proof. We first prove the inequality $\mathsf{Pr}[X \leq (1-\delta) \cdot \tilde{\mu}] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}} \right)^{\tilde{\mu}}$ for $\delta \in (0, 1)$. The proof is analogous to the one of Equation A.1 of Theorem A.1. Specifically, we again let $Y_i = \exp(t \cdot X_i)$, $i \in [N]$, for some t. Specifically, for any $\delta \in (0, 1)$, we now choose $t = \ln(1 - \delta)$, such that $t < 0$ holds.

Note that Inequality A.11 of Lemma A.6 still holds. As $\exp(t)-1 = -\delta < 0$ holds for $t < 0$, the exponential function $f(z) = \exp((\exp(t)-1)\cdot z)$ is monotonically decreasing. Using the lower bound $\tilde{\mu}_i \leq \mu_i = \mathsf{E}[X_i]$ on the expectation of X_i, $i \in [N]$, and $\tilde{\mu} = \sum_{i\in[N]} \tilde{\mu}_i$ the following is obtained:

$$\mathsf{E}[Y_i] \leq \exp((\exp(t)-1)\cdot\tilde{\mu}_i) \tag{A.18}$$

As Equation A.15 holds independently of the choice of t, the result is obtained analogously to Equation A.1:

$$\begin{aligned}
&\mathsf{Pr}[X \leq (1-\delta)\cdot\tilde{\mu}] \\
&= \mathsf{Pr}[\exp(t\cdot X) \geq \exp(t\cdot(1-\delta)\cdot\tilde{\mu})] && \text{(as } t<0\text{)} \\
&\leq \frac{\mathsf{E}[\exp(t\cdot X)]}{\exp(t\cdot(1-\delta)\cdot\tilde{\mu})} && \text{(by Markov's inequality)} \\
&= \frac{\prod_{i\in[N]}\mathsf{E}[Y_i]}{\exp(t\cdot(1-\delta)\cdot\tilde{\mu})} && \text{(by Equation A.15)} \\
&\leq \frac{\prod_{i\in[N]}\exp((\exp(t)-1)\cdot\tilde{\mu}_i)}{\exp(t\cdot(1-\delta)\cdot\tilde{\mu})} && \text{(by Equation A.18)} \\
&= \exp\left(\Big(\sum_{i\in[N]}(\exp(t)-1)\cdot\tilde{\mu}_i\Big) - \big(t\cdot(1-\delta)\cdot\tilde{\mu}\big)\right) && \text{(one exponent)} \\
&= \exp\left(\Big(\sum_{i\in[N]}((1-\delta)-1)\cdot\tilde{\mu}_i\Big) - \big(\ln(1-\delta)\cdot(1-\delta)\cdot\tilde{\mu}\big)\right) && \text{(using } t=\ln(1-\delta)\text{)} \\
&= \exp\left(\big(-\delta\cdot\tilde{\mu}\big) - \big(\ln(1-\delta)\cdot(1-\delta)\cdot\tilde{\mu}\big)\right) && \text{(definition of } \tilde{\mu}=\textstyle\sum_{i\in[N]}\tilde{\mu}_i\text{)} \\
&= \frac{\exp\left(-\delta\cdot\tilde{\mu}\right)}{\exp\left(\ln(1-\delta)\cdot(1-\delta)\cdot\tilde{\mu}\right)} \\
&= \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\tilde{\mu}}
\end{aligned}$$

This completes the proof of the first inequality. The second inequality, namely $\left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\tilde{\mu}} \leq e^{-\delta^2\cdot\tilde{\mu}/2}$ is a corollary of Lemma A.8. Specifically, by Lemma A.8 the following holds for $\delta \in (0,1)$

$$-\delta - (1-\delta)\ln(1-\delta) \leq -\delta^2/2\,. \tag{A.19}$$

Multiplying both sides with $\tilde{\mu}$ and exponentiating both sides yields the desired result. ■

A.3 A Generalized Balls-and-Bins Bound

In this section an often used corollary of the Chernoff bound of Theorem A.1 is proven. Specifically, the corollary can be used to show that for any sufficiently large values $n, c \in \mathbb{N}$ the probability that the sum $X = X_1 + \ldots, X_N$ exceeds the expected value $\mathsf{E}[X]$ by a factor of $O(\ln n/\ln\ln n)$ is upper bounded by $1/n^c$. This result is commonly used when (lower and upper) bounding the number of balls randomly placed in bins as well as for bounding congestion in networks. However, concise proofs are either left to the reader or only more specific cases are studied by, e.g., considering binary random variables $X_i \in \{0,1\}$, $i \in [N]$, not considering upper bounds, or not proving specific values for the constants used in the construction [RT85; MR95; DP09; WS11; AS16; MU17]. Furthermore, we slightly generalize the result to allow to obtain better bounds when $\mathsf{E}[X] \approx \ln n/\ln\ln n$ holds.

Theorem A.3. Let $X = \sum_{i\in[N]} X_i$ be the sum of $N \in \mathbb{N}$ *independent* random variables with $X_i \in [0,1]$ and let $\mu_i = \mathsf{E}[X_i]$ denote their respective expected value for $i \in [N]$. Let $\hat{\mu}_i \geq \mu_i$ be an upper bound on the expected value of $X_i \in [0,1]$, $i \in [N]$, and let $\hat{\mu} = \sum_{i\in[N]} \hat{\mu}_i$ such that,

$$\hat{\mu} = \sum_{i\in[N]} \hat{\mu}_i \geq 1\,. \tag{A.3}$$

Denote by c, n, λ, ρ, ξ constants, such that,

$$\begin{aligned} c \geq& 2 && \text{(A.4)}\\ n \geq& 3 && \text{(A.5)}\\ \rho \geq& 2 && \text{(A.6)}\\ \rho \cdot c \geq& e && \text{(A.7)}\\ \xi \geq& \ln\ln n / \ln n && \text{(A.8)}\\ \lambda =& \rho \cdot c \cdot \xi \cdot \ln n / \ln\ln n\,. && \text{(A.9)} \end{aligned}$$

The following holds:

$$\mathsf{Pr}[X \geq \lambda \cdot \hat{\mu}] \leq 1/n^{c\cdot\hat{\mu}\cdot\xi}\,. \tag{A.10}$$

Proof. First note that $\ln n / \ln\ln n \geq 1$ holds for all $n \geq 3$ (by Equation A.5). Accordingly, $\lambda \geq e$ holds by Equations A.7 to A.9. Setting $\delta = \lambda - 1$ and applying Theorem A.1, the following is obtained:

$$\begin{aligned} &\leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\hat{\mu}} && \text{(by Theorem A.1)}\\ &= \left(\frac{e^{\lambda-1}}{\lambda^{\lambda}}\right)^{\hat{\mu}} && \text{(substituting } \lambda = \delta + 1\text{)}\\ &\leq \left(\frac{e}{\lambda}\right)^{\lambda\cdot\hat{\mu}} && \text{(increasing numerator of fraction } < 1\text{)}\\ &= \left(\frac{e\cdot\ln\ln n}{\rho\cdot c\cdot\xi\cdot\ln n}\right)^{(\rho\cdot c\cdot\xi\cdot\ln n/\ln\ln n)\cdot\hat{\mu}} && \text{(plugging in definition of } \lambda\text{)}\\ &= \left(\frac{\rho\cdot c}{e}\cdot\xi^{-1}\cdot\frac{\ln n}{\ln\ln n}\right)^{-(\rho\cdot c\cdot\xi\ln n/\ln\ln n)\cdot\hat{\mu}} && \text{(}(a/b)^1 = (b/a)^{-1}\text{)}\\ &= \left(\exp\left(\ln\left(\frac{\rho\cdot c}{e}\cdot\xi^{-1}\cdot\frac{\ln n}{\ln\ln n}\right)\right)\right)^{-(\rho\cdot c\cdot\xi\cdot\ln n/\ln\ln n)\cdot\hat{\mu}}\\ &= \Big(\exp\big(\ln(\rho\cdot c/e) + \ln\xi^{-1} + \ln\ln n - \ln\ln\ln n)\big)\Big)^{-(\rho\cdot c\cdot\xi\cdot\ln n/\ln\ln n)\cdot\hat{\mu}}\\ &\leq \Big(\exp\big(\ln(\rho\cdot c/e) + \ln\ln n - \ln\ln\ln n\big)\Big)^{-(\rho\cdot c\cdot\xi\cdot\ln n/\ln\ln n)\cdot\hat{\mu}} && \text{(as } \xi^{-1}\geq 1\text{)}\\ &= \Big(\exp\big(-\rho\cdot c\cdot\ln n/\ln\ln n\cdot\big(\ln(\rho\cdot c/e) + (\ln\ln n - \ln\ln\ln n)\big)\big)\Big)^{\xi\cdot\hat{\mu}}\\ &\leq \Big(\exp\big(-\rho\cdot c\cdot\ln n/\ln\ln n\cdot\big(\ln(\rho\cdot c/e) + (0.5\cdot\ln\ln n)\big)\big)\Big)^{\xi\cdot\hat{\mu}} && \text{(by Lemma A.9)}\\ &\leq \Big(\exp\big(-\rho\cdot c\cdot\ln n/\ln\ln n\cdot(0.5\cdot\ln\ln n)\big)\Big)^{\xi\cdot\hat{\mu}} && \text{(by Equation A.7)}\\ &\leq \Big(\exp\big(-c\cdot\ln n\big)\Big)^{\xi\cdot\hat{\mu}} = 1/n^{c\cdot\xi\cdot\hat{\mu}} && \text{(by Equation A.6)} \end{aligned}$$

■

"Ever since I switched from economics to probability and statistics in my early student days, this area has continued to absorb my interests. The very idea that the seeming chaos of chance obeys mathematical laws is immensely attractive."

– Wassily Hoeffding

B

Randomized Rounding Analysis using Hoeffding Bounds

In Sections 7.1 and 7.2 the performance of the different VNEP approximations was analyzed. Specifically, Chernoff bounds were used for bounding the resource allocations. Within this Chapter, we now derive resource bounds by employing the following Hoeffding bound.

Theorem B.1 (Hoeffding's inequality [DP09])**.** Let $X = \sum_{i=1}^{N} X_i$, $X_i \in [a_i, b_i]$, be a sum of N independent random variables. The following holds for any $t \geq 0$:

$$\mathsf{Pr}[X - \mathsf{E}(X) \geq t] \leq \exp\left(-2t^2 / \left(\sum_{i\in[n]} (b_i - a_i)^2\right)\right)$$

In Section B.1 we first derive a general lemma similar to Lemma 7.5 and then apply this result to the different approximations. In Section B.2 we compare the obtained results with the ones based on the Chernoff bounds.

B.1 Bounding Resource Allocations using Hoeffding

In the following, we analyze the probability that a rounded solution exceeds substrate capacities by a certain factor. We first recapitulate the notation used in Chapter 7. Allocations on resource $x \in G_S$ by request $r \in \mathcal{R}$ are modeled as random variable $A_{r,x} \in [0, A_{\mathsf{max}}(r,x)]$. By definition, we have $\mathsf{Pr}[A_{r,x} = A(m_r^k, x)] = f_r^k$ and $\mathsf{Pr}[A_{r,x} = 0] = 1 - \sum_k f_r^k$. Furthermore, we denote by $A_x = \sum_{r\in\mathcal{R}} A_{r,x}$ the random variable capturing the overall allocations on resource $x \in G_S$. By definition $\mathsf{E}[A_x] = \sum_{r\in\mathcal{R}} \sum_k f_r^k \cdot A(m_r^k, x)$ holds. We employ Hoeffding's inequality to upper bound A_x as follows.

Lemma B.2. Consider a resource $x \in G_S$ and let $\Delta_x > 0$ be chosen in such a way that $\Delta_x \cdot d_S(x)$ is an upper bound on the expected allocations on x, i.e., $\Delta_x \cdot d_S(x) \geq \mathsf{E}(A_x)$ holds. Furthermore, let $\hat{\mu}_{r,x} = A_{\mathsf{max}}(r,x)/d_S(x)$, $\Gamma(x) = \sum_{r\in\mathcal{R}} \hat{\mu}_{r,x}^2$, and $\delta(x,\lambda) = \sqrt{c \cdot \Gamma(x) \cdot \ln\lambda}$. Then the following holds for any $\lambda, c \geq 1$:

$$\mathsf{Pr}(A_x \geq (\Delta_x + \delta(x,\lambda)) \cdot d_S(x)) \leq \lambda^{-2\cdot c}\,. \tag{B.1}$$

Proof. We apply the Hoeffding bound (cf. Theorem B.1) with $t = \delta(x,\lambda) \cdot d_S(x)$:

$$\mathsf{Pr}\Big(A_x - \mathsf{E}(A_x) \geq \delta(x,\lambda) \cdot d_S(x)\Big) \leq \exp\left(\frac{-2 \cdot c \cdot \Gamma(x) \cdot \ln\lambda \cdot d_S^2(x)}{\sum_{r\in\mathcal{R}} (A_{\mathsf{max}}(r,x))^2}\right)$$

$$= \exp\left(\frac{-2 \cdot c \cdot \Gamma(x) \cdot \ln \lambda}{\sum_{r \in \mathcal{R}} (A_{\mathsf{max}}(r,x)/d_S(x))^2}\right)$$
$$= \exp\left(\frac{-2 \cdot c \cdot \Gamma(x) \cdot \ln \lambda}{\Gamma(x)}\right) \qquad = \lambda^{-2 \cdot c} \tag{B.2}$$

As we assume $\mathsf{E}(A_x) \leq \Delta_x \cdot d_S(x)$ to hold, Equation B.1 follows directly from Equation B.2. ■

Given Lemma B.2, we obtain the following corollary.

Corollary B.3. Let $\Delta_V, \Delta_E > 0$ be chosen minimally s.t. $\Delta_V \cdot d_S(u) \geq \mathsf{E}[A_u]$ and $\Delta_E \cdot d_S(u,v) \geq \mathsf{E}[A_{u,v}]$ hold for $u \in V_S$ and $(u,v) \in E_S$, respectively. Furthermore, let $\Gamma(X) = \max_{x \in X} \Gamma(x) = \max_{x \in X} \sum_{r \in \mathcal{R}} \hat{\mu}_{r,x}^2$ for a set of substrate resources $X \subseteq G_S$ with $\hat{\mu}_{r,x} = A_{\mathsf{max}}(x)/d_S(x)$. Considering a substrate graph of n nodes, i.e., $n = |V_S|$, the following holds for any $c \geq 1$:

$$\mathsf{Pr}\left[\begin{array}{c} \exists u \in V_S : \quad A_u \geq (\Delta_V + \sqrt{c \cdot \Gamma(V_S) \cdot \ln n}) \cdot d_S(u) \\ \vee\, \exists (u,v) \in E_S : A_{u,v} \geq (\Delta_E + \sqrt{c \cdot \Gamma(E_S) \cdot \ln n}) \cdot d_S(u,v) \end{array}\right] \leq n^{2-2 \cdot c}. \tag{B.3}$$

Proof. First, note that $\Gamma(V_S) \geq \Gamma(u)$ and $\Gamma(E_S) \geq \Gamma(u,v)$ hold for $u \in V_S$ and $(u,v) \in E_S$, respectively. By applying Lemma B.2 for each of the n node resources and each of the at most $n \cdot (n-1)$ edge resources while setting $\lambda = n$, the result is obtained via a union bound. ■

The performance analysis of the (randomized) approximation algorithms can be easily adapted by using the above corollary instead of the probabilistic resource bounds stated in Corollaries 7.6 (for the cost VNEP $\langle \mathbf{VE} \,|\, \mathbf{NR} \rangle$ approximation) and 7.10 (for the profit VNEP $\langle \mathbf{VE} \,|\, \mathbf{NR} \rangle$). Accordingly, the following approximation results are obtained when using Hoeffding:

Theorem B.4. By using the above derived Hoeffding bounds, Theorems 7.7, 7.8, 7.11, and 7.13 can be adapted to obtain the resource allocation bounds stated in Tables B.1 and B.2.

Objective/ Setting		**Resource Approximation Factors**	
		$\boldsymbol{\beta - \beta' = \gamma - \gamma'}$	$\mathbf{max\{\beta, \gamma\}}$
Cost	$\langle \mathbf{VE} \,|\, \mathbf{NR} \rangle$	2	$2 + \delta_{\max} \cdot \kappa_{\mathsf{max}} \cdot \sqrt{3/2 \cdot \lvert\mathcal{R}\rvert \cdot \ln n}$
	$\langle \mathbf{VE} \,|\, \mathbf{NRL} \rangle$	$2 \cdot (1 + \varepsilon_{\mathrm{L}})$	$2 \cdot (1 + \varepsilon_{\mathrm{L}}) + \delta_{\max} \cdot \kappa_{\mathsf{max}} \cdot \sqrt{3/2 \cdot \lvert\mathcal{R}\rvert \cdot \ln n}$

Table B.1: Resource approximation factors of Theorems 7.7 and 7.8 for the cost offline VNEP when using the Hoeffding inequality; $\beta' = \sqrt{3/2 \cdot \Gamma(V_S) \cdot \ln n}$ and $\gamma' = \sqrt{3/2 \cdot \Gamma(E_S) \cdot \ln n}$.

Objective/ Setting		**Resource Approximation Factors**	
		$\boldsymbol{\beta - \beta' = \gamma - \gamma'}$	$\mathbf{max\{\beta, \gamma\}}$
Profit	$\langle \mathbf{VE} \,|\, \mathbf{NR} \rangle$	1	$1 + \delta_{\max} \cdot \kappa_{\mathsf{max}} \cdot \sqrt{2 \cdot \lvert\mathcal{R}\rvert \cdot \ln n}$
	$\langle \mathbf{VE} \,|\, \mathbf{NRL} \rangle$	$1 + \varepsilon_{\mathrm{L}}$	$1 \cdot (1 + \varepsilon_{\mathrm{L}}) + \delta_{\max} \cdot \kappa_{\mathsf{max}} \cdot \sqrt{2 \cdot \lvert\mathcal{R}\rvert \cdot \ln n}$

Table B.2: Resource approximation factors of Theorems 7.11 and 7.13 for the profit offline VNEP when using the Hoeffding inequality; $\beta' = \sqrt{2 \cdot \Gamma(V_S) \cdot \ln n}$ and $\gamma' = \sqrt{2 \cdot \Gamma(E_S) \cdot \ln n}$.

Within the tables, again the following notation is used. $\delta_{\max}$ denotes the maximal demand to capacity ratio, i.e., $\delta_{\max} = \max_{x \in G_S} d_{\mathsf{max}}(x)/d_S(x)$, κ_{max} denotes the maximal request size, i.e., $\kappa_{\mathsf{max}} =$

$\max_{r\in\mathcal{R}}\{|V_r|,|E_r|\}$, $\varepsilon_{\mathrm{L}} > 0$ denotes the approximation factor for the LCSP FPTAS, $|\mathcal{R}|$ denotes the number of requests, and $n = |V_S|$ denotes the number of substrate nodes.

Proof. To obtain the results, note the following. The probability to violate capacities in the proofs of Theorems 7.7 and 7.8 was upper bounded by $1/n$ for $n \geq 3$. Accordingly, setting $c = 3/2$ in Corollary B.3, the respective results in Table B.1 are obtained by setting Δ_V and Δ_E to 2 (for the approximation without latencies) and Δ_V and Δ_E to $2 \cdot (1 + \varepsilon_{\mathrm{L}})$ (for the approximation with latencies). Similarly, the resource bounds β and γ for the profit approximation are obtained by setting $c = 2$ and Δ_V and Δ_E to 1 (for the approximation without latencies) and Δ_V and Δ_E to $1 + \varepsilon_{\mathrm{L}}$ (for the approximation with latencies) in Corollary B.3.

To obtain the respective bounds on $\max\{\beta,\gamma\}$, note the following. By definition of $\Gamma(X) = \sum_{r\in\mathcal{R}} \hat{\mu}_{r,x}^2$ and using $A_{\mathsf{max}}(x) \leq \delta_{\max} \cdot \kappa_{\mathsf{max}} \cdot d_S(x)$, the following bounds for $\Gamma(V_S)$ and $\Gamma(E_S)$ are obtained:

$$\Gamma(V_S) \leq \max_{x\in V_S} \sum_{r\in\mathcal{R}} (A_{\mathsf{max}}(x)/d_S(x))^2 \leq |\mathcal{R}| \cdot \delta_{\max}^2 \cdot \kappa_{\mathsf{max}}^2 \tag{B.4}$$

$$\Gamma(E_S) \leq \max_{x\in E_S} \sum_{r\in\mathcal{R}} (A_{\mathsf{max}}(x)/d_S(x))^2 \leq |\mathcal{R}| \cdot \delta_{\max}^2 \cdot \kappa_{\mathsf{max}}^2 \tag{B.5}$$

Plugging these bounds in, the maximal resource augmentation factors are readily obtained. ■

B.2 Comparison of Chernoff and Hoeffding Bounds

Given the alternative resource bounds derived above, the question arises which of these bounds yield the lowest resource augmentations. In the following we focus on the comparison of the resource bounds for the profit approximation without latencies. Given the bounds stated in Tables 7.1 and B.2, the resource augmentations according to the Chernoff and Hoeffding bounds are bounded by $\mathcal{O}(\kappa_{\mathsf{max}} \cdot \delta_{\max} \cdot \ln n / \ln\ln n)$ and $\mathcal{O}(\kappa_{\mathsf{max}} \cdot \delta_{\max} \cdot \sqrt{|\mathcal{R}| \cdot \ln n})$, respectively. As $\ln n / \ln\ln n$ only grows negligibly faster than $\sqrt{\ln n}$, the bounds mainly differ by the Hoeffding bound including the factor $\sqrt{|\mathcal{R}|}$.

While the above suggests that the Chernoff bounds are much tighter than the Hoeffding bound (disregarding the factor $\sqrt{\ln n}/\ln\ln n$ which lies below 2 for $n \leq 10^{32}$), the Chernoff bound contains larger constant factors. As accurate bounds on the resource augmentations are of importance to obtain the least resource violations, e.g., when employing derandomization, Figure B.1 depicts the actual quotient of the respective bounds. For small number of requests, the resource bounds obtained by the Hoeffding inequality may improve over the Chernoff bounds by a factor of roughly 10, while starting with roughly 200 requests and for values $\hat{\mu} \geq 0.25$, the Chernoff bound improves over the Hoeffding one. The Chernoff bound improves over the Hoeffding bound by a factor of 10^3 for $5 \cdot 10^8$ many requests. Considering small resource demands, i.e., $\hat{\mu} \leq 2 \cdot 10^{-2}$ for $n = 10^{-2}$ and $\hat{\mu} \leq 4 \cdot 10^{-3}$ for $n = 10^9$, the Chernoff and the Hoeffding bounds yield nearly identical results up to roughly $10k$ requests.

Summarizing, the resource bounds based on the Hoeffding inequality are generally more accurate for up to 200 requests, while afterwards the additional factor $\mathcal{O}(\sqrt{|\mathcal{R}|})$ renders the bounds based on the Chernoff inequality more accurate.

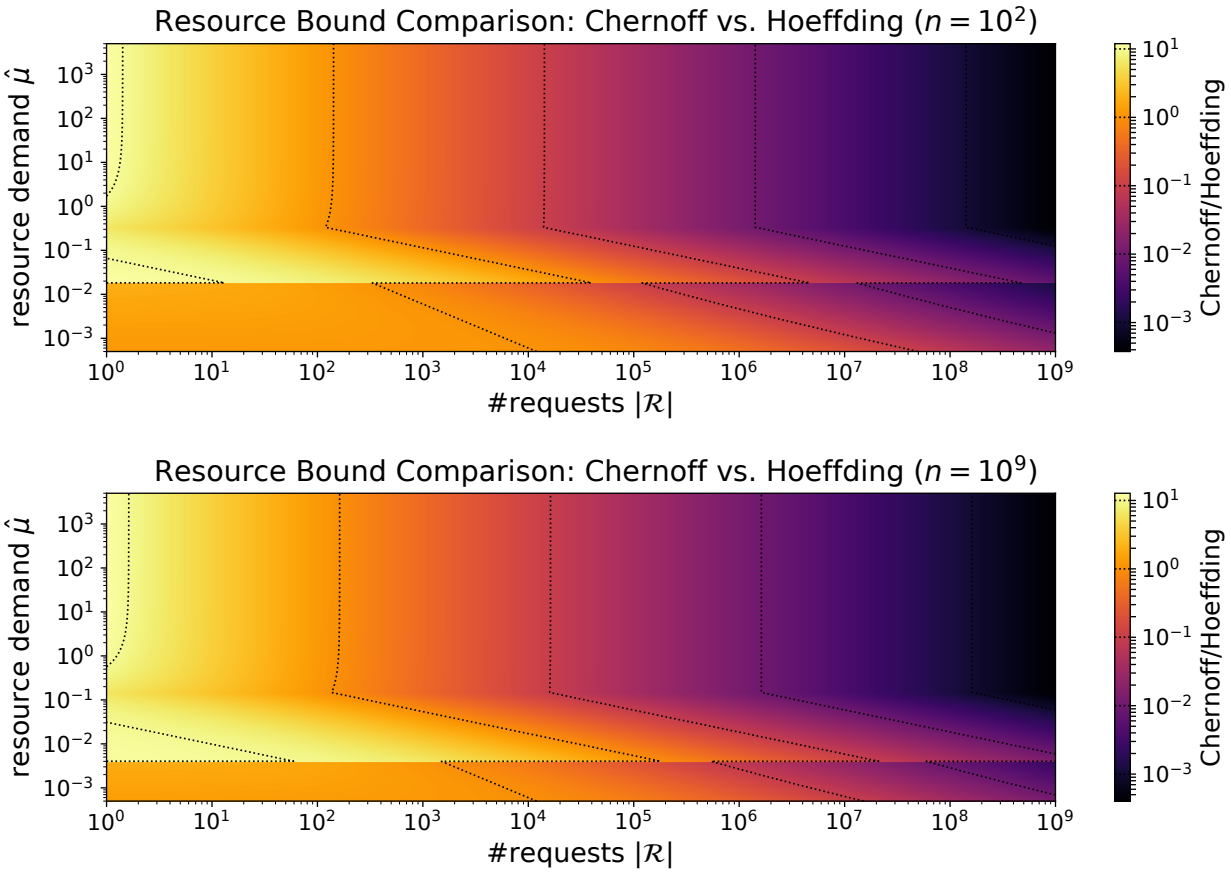


Figure B.1: Comparison of the Chernoff and Hoeffding resource allocation bounds for the profit approximation without latencies, i.e., $\Delta_V = \Delta_E = 1$, in dependence of the number of requests and the resource demand $\hat{\mu} = \delta_{\max} \cdot \kappa_{\max}$ for $n = 10^2$ and $n = 10^9$. Depicted is the quotient of the resource bounds based on the Chernoff bound and the ones based on the Hoeffding bound. Depicted are also the contour lines for the values $10^1, 10^0, \ldots, 10^{-3}$. Note that the three distinct horizontal bands are due to the three branches of the function Λ_{prof} (cf. Corollary 7.10). Given the definition of Λ_{prof}, the size of the middle bands slightly increases when increasing the number of nodes.

List of Algorithms

List of Figures

List of Tables

Bibliography

[AB09] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[Ach09] T. Achterberg. "SCIP: solving constraint integer programs". In: *Mathematical Programming Computation* 1.1 (July 2009), pp. 1–41. ISSN: 1867-2957. DOI: 10.1007/s12532-008-0001-1.

[AKK12] V. Abhishek, I. A. Kash, and P. Key. "Fixed and market pricing for cloud services". In: *2012 Proceedings IEEE INFOCOM Workshops*. Mar. 2012, pp. 157–162. DOI: 10.1109/INFCOMW.2012.6193479.

[Alt+07] A. Altın, E. Amaldi, P. Belotti, and M. Ç. Pınar. "Provisioning virtual private networks under traffic uncertainty". In: *Networks* 49.1 (Jan. 2007), pp. 100–115. DOI: 10.1002/net.20145.

[ALV08] M. Al-Fares, A. Loukissas, and A. Vahdat. "A Scalable, Commodity Data Center Network Architecture". In: *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*. SIGCOMM '08. Seattle, WA, USA: ACM, 2008, pp. 63–74. ISBN: 978-1-60558-175-0. DOI: 10.1145/1402958.1402967.

[Ama+16] E. Amaldi, S. Coniglio, A. M. Koster, and M. Tieves. "On the computational complexity of the virtual network embedding problem". In: *Electronic Notes in Discrete Mathematics* 52 (2016), pp. 213–220.

[Amo+13] A. Amokrane, M. F. Zhani, R. Langar, R. Boutaba, and G. Pujolle. "Greenhead: Virtual Data Center Embedding across Distributed Infrastructures". In: *IEEE Transactions on Cloud Computing* 1.1 (Jan. 2013), pp. 36–49. ISSN: 2168-7161. DOI: 10.1109/TCC.2013.5.

[And+05] T. Anderson, L. Peterson, S. Shenker, and J. Turner. "Overcoming the Internet impasse through virtualization". In: *Computer* 38.4 (Apr. 2005), pp. 34–41. ISSN: 0018-9162. DOI: 10.1109/MC.2005.136.

[And02] D. G. Andersen. "Theoretical Approaches to Node Assignment". [Online]. Available: http://repository.cmu.edu/compsci/86/. Dec. 2002.

[Ans99] K. M. Anstreicher. "Linear programming in O ([n3/ln n] L) operations". In: *SIAM Journal on Optimization* 9.4 (1999), pp. 803–812.

[AS16] N. Alon and J. H. Spencer. *The Probabilistic Method*. 4th. Wiley Publishing, 2016. ISBN: 9781119061953. DOI: 10.1002/0471722154.

[Bal+11] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. "Towards Predictable Datacenter Networks". In: *Proceedings of the ACM SIGCOMM 2011 Conference*. SIGCOMM '11. Toronto, Ontario, Canada: ACM, 2011, pp. 242–253. ISBN: 978-1-4503-0797-0. DOI: 10.1145/2018436.2018465.

[Ban+11] N. Bansal, K.-W. Lee, V. Nagarajan, and M. Zafer. "Minimum Congestion Mapping in a Cloud". In: *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*. PODC '11. San Jose, California, USA: ACM, 2011, pp. 267–276. ISBN: 978-1-4503-0719-2. DOI: 10.1145/1993806.1993854.

[Ban+15] N. Bansal, K. Lee, V. Nagarajan, and M. Zafer. "Minimum Congestion Mapping in a Cloud". In: *SIAM Journal on Computing* 44.3 (2015), pp. 819–843. DOI: 10.1137/110845239.

[Bas+17] A. Basta, A. Blenk, K. Hoffmann, H. J. Morper, M. Hoffmann, and W. Kellerer. "Towards a Cost Optimal Design for a 5G Mobile Core Network Based on SDN and NFV". In: *IEEE Transactions on Network and Service Management* 14.4 (Dec. 2017), pp. 1061–1075. ISSN: 1932-4537. DOI: 10.1109/TNSM.2017.2732505.

[Bay+13] L. R. Bays, R. R. Oliveira, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary. "Security-aware Optimal Resource Allocation for Virtual Network Embedding". In: *Proceedings of the 8th International Conference on Network and Service Management*. Las Vegas, Nevada: IFIP, 2013.

[BE05] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 2005. ISBN: 9780521619462.

[Beg+17] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis, and X. Costa-Perez. "Optimising 5G infrastructure markets: The business of network slicing". In: *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. May 2017. DOI: 10.1109/INFOCOM.2017.8057045.

[Bha+16] D. Bhamare, R. Jain, M. Samaka, and A. Erbad. "A survey on service function chaining". In: *Journal of Network and Computer Applications* 75 (2016), pp. 138–155. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2016.09.001.

[Bie+14] M. Bienkowski, A. Feldmann, J. Grassler, G. Schaffrath, and S. Schmid. "The Wide-Area Virtual Service Migration Problem: A Competitive Analysis Approach". In: *IEEE/ACM Trans. Netw.* 22.1 (Feb. 2014), pp. 165–178. ISSN: 1063-6692. DOI: 10.1109/TNET.2013.2245676.

[BN09] N. Buchbinder and J. (Naor. "The Design of Competitive Online Algorithms via a Primal–Dual Approach". In: *Foundations and Trends® in Theoretical Computer Science* 3.2–3 (2009), pp. 93–263. ISSN: 1551-305X. DOI: 10.1561/0400000024.

[Bod96] H. L. Bodlaender. "A linear-time algorithm for finding tree-decompositions of small treewidth". In: *SIAM Journal on computing* 25.6 (1996), pp. 1305–1317.

[Bod97] H. L. Bodlaender. "Treewidth: Algorithmic techniques and results". In: *Proceedings of Mathematical Foundations of Computer Science 1997*. Ed. by I. Prívara and P. Ružička. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 19–36. ISBN: 978-3-540-69547-9.

[Bod98] H. L. Bodlaender. "A partial k-arboretum of graphs with bounded treewidth". In: *Theoretical Computer Science* 209.1 (1998), pp. 1–45. ISSN: 0304-3975. DOI: https://doi.org/10.1016/S0304-3975(97)00228-4.

[Bou+15] M. Bouet, J. Leguay, T. Combe, and V. Conan. "Cost-based placement of vDPI functions in NFV infrastructures". In: *International Journal of Network Management* 25.6 (2015), pp. 490–506. DOI: 10.1002/nem.1920.

[CB09] N. M. M. K. Chowdhury and R. Boutaba. "Network virtualization: state of the art and research challenges". In: *IEEE Communications Magazine* 47.7 (July 2009), pp. 20–26. ISSN: 0163-6804. DOI: 10.1109/MCOM.2009.5183468.

[Cha+01] P. Chandra, Y.-h. Chu, A. Fisher, J. Gao, C. Kosak, T. S. E. Ng, P. Steenkiste, E. Takahashi, and H. Zhang. "Darwin: customizable resource management for value-added network services". In: *IEEE Network* 15.1 (Jan. 2001), pp. 22–35. ISSN: 0890-8044. DOI: 10.1109/65.898819.

[Che+11] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang. "Virtual Network Embedding Through Topology-aware Node Ranking". In: *SIGCOMM Comput. Commun. Rev.* 41.2 (Apr. 2011), pp. 38–47. ISSN: 0146-4833. DOI: 10.1145/1971162.1971168.

[Chi+12] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Deng, J. Benitez, U. Michel, H. Damker, K. Ogaki, T. Matsuzaki, M. Fukui, K. Shimano, D. Delisle, Q. Loudier, C. Kolias, I. Guardini, E. Demaria, R. Minerva, A. Manzalini, D. López, F. J. R. Salguero, F. Ruhl, and P. Sen. "Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action". In: *SDN and OpenFlow World Congress*. Oct. 2012. URL: https://portal.etsi.org/NFV/NFV_White_Paper.pdf (visited on Sept. 19, 2019).

[Cho+11] N. M. K. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica. "Managing Data Transfers in Computer Clusters with Orchestra". In: *Proceedings of the ACM SIGCOMM 2011 Conference*. SIGCOMM '11. Toronto, Ontario, Canada: ACM, 2011, pp. 98–109. ISBN: 978-1-4503-0797-0. DOI: 10.1145/2018436.2018448.

[Chu+07] J. Chuzhoy, V. Guruswami, S. Khanna, and K. Talwar. "Hardness of Routing with Congestion in Directed Graphs". In: *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*. STOC '07. San Diego, California, USA: ACM, 2007, pp. 165–178. ISBN: 978-1-59593-631-8. DOI: 10.1145/1250790.1250816.

[CLN04] Y. Cui, B. Li, and K. Nahrstedt. "On Achieving Optimized Capacity Utilization in Application Overlay Networks with Multiple Competing Sessions". In: *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*. SPAA '04. Barcelona, Spain: ACM, 2004, pp. 160–169. ISBN: 1-58113-840-7. DOI: 10.1145/1007912.1007937.

[Coh+15] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz. "Near optimal placement of virtual network functions". In: *2015 IEEE Conference on Computer Communications (INFOCOM)*. Apr. 2015, pp. 1346–1354. DOI: 10.1109/INFOCOM.2015.7218511.

[Cor+09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009. ISBN: 9780262033848.

[Cor08] G. Cornuéjols. "Valid inequalities for mixed integer linear programs". In: *Mathematical Programming* 112.1 (Mar. 2008), pp. 3–44. ISSN: 1436-4646. DOI: 10.1007/s10107-006-0086-0.

[CRB09] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba. "Virtual Network Embedding with Coordinated Node and Link Mapping". In: *IEEE INFOCOM 2009*. Apr. 2009, pp. 783–791. DOI: 10.1109/INFCOM.2009.5061987.

[CRB12] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba. "ViNEYard: Virtual Network Embedding Algorithms with Coordinated Node and Link Mapping". In: *IEEE/ACM Trans. Netw.* 20.1 (Feb. 2012), pp. 206–219. ISSN: 1063-6692. DOI: 10.1109/TNET.2011.2159308.

[Dai+15] X. Dai, Y. Wang, J. M. Wang, and B. Bensaou. "Energy Efficient Virtual Cluster Embedding in Public Data Centers". In: *2015 IEEE Global Communications Conference (GLOBECOM)*. Dec. 2015, pp. 1–6. DOI: 10.1109/GLOCOM.2015.7416982.

[DF13] R. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer-Verlag London, 2013. ISBN: 978-1-4471-5558-4. DOI: 10.1007/978-1-4471-5559-1.

[DKM18] S. Dräxler, H. Karl, and Z. Á. Mann. "JASPER: Joint Optimization of Scaling, Placement, and Routing of Virtual Network Services". In: *IEEE Transactions on Network and Service Management* 15.3 (Sept. 2018), pp. 946–960. ISSN: 1932-4537. DOI: 10.1109/TNSM.2018.2846572.

[DL05] J. Desrosiers and M. E. Lübbecke. "A Primer in Column Generation". In: *Column Generation*. Ed. by G. Desaulniers, J. Desrosiers, and M. M. Solomon. Boston, MA: Springer US, 2005, pp. 1–32. ISBN: 978-0-387-25486-9. DOI: 10.1007/0-387-25486-2_1.

[Döh+19] E. Döhne, A. Elvers, T. Koch, and M. Rost. *GitHub Organization containing the Projects for the Evaluation of the Randomized Rounding Algorithms*. https://github.com/vnep-approx/. 2019.

[Döh18] E. Döhne. "Virtual Network Embedding via Decomposable LP Formulations: Orientations of Small Extraction Width and Beyond". MA thesis. Technische Universität Berlin, Oct. 2018. URL: https://arxiv.org/abs/1810.11280 (visited on Sept. 19, 2019).

[DP09] D. P. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009. DOI: 10.1017/CBO9780511581274.

[DPS02] J. Diaz, J. Petit, and M. Serna. "A Survey of Graph Layout Problems". In: *ACM Comput. Surv.* 34.3 (Sept. 2002), pp. 313–356. ISSN: 0360-0300. DOI: 10.1145/568522.568523.

[DS18] Y. Disser and M. Skutella. "The Simplex Algorithm Is NP-Mighty". In: *ACM Trans. Algorithms* 15.1 (Nov. 2018). ISSN: 1549-6325. DOI: 10.1145/3280847.

[Duf+99] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merive. "A Flexible Model for Resource Management in Virtual Private Networks". In: *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. SIGCOMM '99. Cambridge, Massachusetts, USA: ACM, 1999, pp. 95–108. ISBN: 1-58113-135-6. DOI: 10.1145/316188.316209.

[EMP16] G. Even, M. Medina, and B. Patt-Shamir. "On-Line Path Computation and Function Placement in SDNs". In: *Stabilization, Safety, and Security of Distributed Systems*. Ed. by B. Bonakdarpour and F. Petit. Cham: Springer International Publishing, 2016, pp. 131–147. ISBN: 978-3-319-49259-9.

[Epp02] D. Eppstein. "Subgraph Isomorphism in Planar Graphs and Related Problems". In: *Graph Algorithms and Applications I*. 2002, pp. 283–309. DOI: 10.1142/9789812777638_0014.

[ERS16a] G. Even, M. Rost, and S. Schmid. "An Approximation Algorithm for Path Computation and Function Placement in SDNs". In: *Structural Information and Communication Complexity*. Ed. by J. Suomela. Cham: Springer International Publishing, 2016, pp. 374–390. ISBN: 978-3-319-48314-6. DOI: 10.1007/978-3-319-48314-6_24.

[ERS16b] G. Even, M. Rost, and S. Schmid. "An Approximation Algorithm for Path Computation and Function Placement in SDNs". In: *CoRR* abs/1603.09158 (2016). URL: http://arxiv.org/abs/1603.09158 (visited on Sept. 19, 2019).

[Eve+13] G. Even, M. Medina, G. Schaffrath, and S. Schmid. "Competitive and deterministic embeddings of virtual networks". In: *Theoretical Computer Science* 496 (2013). Distributed Computing and Networking (ICDCN 2012), pp. 184–194. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2012.10.036.

[Fat18] F. Fattohi. "Competitive Online Virtual Cluster Embedding Algorithms". MA thesis. Technische Universität Berlin, Oct. 2018. URL: https://arxiv.org/abs/1810.03162 (visited on Sept. 19, 2019).

[Fel+95] M. R. Fellows, J. Kratochvil, M. Middendorf, and F. Pfeiffer. "The complexity of induced minors and related problems". In: *Algorithmica* 13.3 (Mar. 1995), pp. 266–282. ISSN: 1432-0541. DOI: 10.1007/BF01190507.

[FF10] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton, NJ, USA: Princeton University Press, 2010. ISBN: 9780691146676.

[FG06] J. Flum and M. Grohe. *Parameterized complexity theory*. Springer-Verlag Berlin Heidelberg, 2006. ISBN: 978-3-540-29952-3. DOI: 10.1007/3-540-29953-X.

[FIC19] FICO. *Xpress Optimization*. 2019. URL: https://www.fico.com/en/products/fico-xpress-optimization (visited on Sept. 19, 2019).

[Fis+13] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach. "Virtual Network Embedding: A Survey". In: *IEEE Communications Surveys Tutorials* 15.4 (Apr. 2013), pp. 1888–1906. ISSN: 1553-877X. DOI: 10.1109/SURV.2013.013013.00155.

[FL04] C. A. Floudas and X. Lin. "Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review". In: *Computers & Chemical Engineering* 28.11 (2004), pp. 2109–2129. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2004.05.002.

[Fou+17] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina. "Network Slicing in 5G: Survey and Challenges". In: *IEEE Communications Magazine* 55.5 (May 2017), pp. 94–100. ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1600951.

[Gle+18] A. Gleixner, M. Bastubbe, L. Eifler, T. Gally, G. Gamrath, R. L. Gottwald, G. Hendel, C. Hojny, T. Koch, M. E. Lübbecke, S. J. Maher, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, F. Schlösser, C. Schubert, F. Serrano, Y. Shinano, J. M. Viernickel, M. Walter, F. Wegscheider, J. T. Witt, and J. Witzig. *The SCIP Optimization Suite 6.0*. eng. Tech. rep. 18-26. Takustr. 7, 14195 Berlin: ZIB, 2018.

[GLS81] M. Grötschel, L. Lovász, and A. Schrijver. "The ellipsoid method and its consequences in combinatorial optimization". In: *Combinatorica* 1.2 (1981), pp. 169–197.

[GLS88] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*. Springer-Verlag Berlin Heidelberg, 1988. ISBN: 978-3-642-97883-8. DOI: 10.1007/978-3-642-97881-4.

[Goo19] Google. *OR-Tools*. 2019. URL: https://developers.google.com/optimization/ (visited on Sept. 19, 2019).

[GOS08] N. Goyal, N. Olver, and F. B. Shepherd. "The VPN Conjecture is True". In: *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*. STOC '08. Victoria, British Columbia, Canada: ACM, 2008, pp. 443–450. ISBN: 978-1-60558-047-0. DOI: 10.1145/1374376.1374440.

[Gre+09] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. "VL2: A Scalable and Flexible Data Center Network". In: *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*. SIGCOMM '09. Barcelona, Spain: ACM, 2009, pp. 51–62. ISBN: 978-1-60558-594-9. DOI: 10.1145/1592568.1592576.

[Guo+09] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. "BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers". In: *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*. SIGCOMM '09. Barcelona, Spain: ACM, 2009, pp. 63–74. ISBN: 978-1-60558-594-9. DOI: 10.1145/1592568.1592577.

[Gup+01] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener. "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. Hersonissos, Greece: ACM, 2001, pp. 389–398. ISBN: 1-58113-349-9. DOI: 10.1145/380752.380830.

[Gur19] Gurobi Optimization. *Gurobi Solver*. 2019. URL: https://www.gurobi.com (visited on Sept. 19, 2019).

[HB16] J. G. Herrera and J. F. Botero. "Resource Allocation in NFV: A Comprehensive Survey". In: *IEEE Transactions on Network and Service Management* 13.3 (Sept. 2016), pp. 518–532. ISSN: 1932-4537. DOI: 10.1109/TNSM.2016.2598420.

[Hen+10] T. A. Henzinger, A. V. Singh, V. Singh, T. Wies, and D. Zufferey. "A Marketplace for Cloud Resources". In: *Proceedings of the Tenth ACM International Conference on Embedded Software*. EMSOFT '10. Scottsdale, Arizona, USA: ACM, 2010, pp. 1–8. ISBN: 978-1-60558-904-6. DOI: 10.1145/1879021.1879022.

[HLZ08] I. Houidi, W. Louati, and D. Zeghlache. "A Distributed Virtual Network Mapping Algorithm". In: *2008 IEEE International Conference on Communications*. May 2008, pp. 5634–5640. DOI: 10.1109/ICC.2008.1056.

[HP15] J. M. Halpern and C. Pignataro. *Service Function Chaining (SFC) Architecture*. RFC 7665. Oct. 2015. DOI: 10.17487/RFC7665.

[IBM19] IBM. *ILOG CPLEX Optimization Studio*. 2019. URL: https://www.ibm.com/en-us/products/ilog-cplex-optimization-studio (visited on Sept. 19, 2019).

[IR11] J. Inführ and G. R. Raidl. "Introducing the Virtual Network Mapping Problem with Delay, Routing and Location Constraints". In: *Network Optimization*. Ed. by J. Pahl, T. Reiners, and S. Voß. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 105–117. ISBN: 978-3-642-21527-8.

[Jai+13] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. "B4: Experience with a Globally-deployed Software Defined Wan". In: *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*. SIGCOMM '13. Hong Kong, China: ACM, 2013, pp. 3–14. ISBN: 978-1-4503-2056-6. DOI: 10.1145/2486001.2486019.

[JK15] A. Jarray and A. Karmouch. "Decomposition Approaches for Virtual Network Embedding with One-shot Node and Link Mapping". In: *IEEE/ACM Trans. Netw.* 23.3 (June 2015), pp. 1012–1025. ISSN: 1063-6692. DOI: 10.1109/TNET.2014.2312928.

[JP13] R. Jain and S. Paul. "Network virtualization and software defined networking for cloud computing: a survey". In: *IEEE Communications Magazine* 51.11 (Nov. 2013), pp. 24–31. ISSN: 0163-6804. DOI: 10.1109/MCOM.2013.6658648.

[Kar72] R. M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations*. Ed. by R. E. Miller, J. W. Thatcher, and J. D. Bohlinger. Boston, MA: Springer US, 1972, pp. 85–103. ISBN: 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2_9.

[KM98] T. Koch and A. Martin. "Solving Steiner tree problems in graphs to optimality". In: *Networks* 32.3 (Dec. 1998), pp. 207–232. DOI: 10.1002/(SICI)1097-0037(199810)32:3<207::AID-NET5>3.0.CO;2-O.

[Kni+11] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. "The Internet Topology Zoo". In: *IEEE Journal on Selected Areas in Communications* 29.9 (Oct. 2011), pp. 1765–1775. ISSN: 0733-8716. DOI: 10.1109/JSAC.2011.111002.

[Kot+15a] V. Kotronis, R. Klöti, M. Rost, P. Georgopoulos, B. Ager, S. Schmid, and X. Dimitropoulos. "Investigating the Potential of the Inter-IXP Multigraph for the Provisioning of Guaranteed End-to-End Services". In: *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. SIGMETRICS '15. Portland, Oregon, USA: ACM, 2015, pp. 429–430. ISBN: 978-1-4503-3486-0. DOI: 10.1145/2745844.2745877.

[Kot+15b] V. Kotronis, R. Klöti, M. Rost, P. Georgopoulos, B. Ager, S. Schmid, and X. Dimitropoulos. *Investigating the Potential of the Inter-IXP Multigraph for the Provisioning of Guaranteed End-to-End Services*. Tech. rep. 360. ETH Zurich, Laboratory TIK, Feb. 2015. URL: ftp://ftp.tik.ee.ethz.ch/pub/publications/TIK-Report-360.pdf (visited on Sept. 19, 2019).

[Kot+16a] V. Kotronis, R. Klöti, M. Rost, P. Georgopoulos, B. Ager, S. Schmid, and X. Dimitropoulos. "Stitching Inter-Domain Paths over IXPs". In: *Proceedings of the Symposium on SDN Research*. SOSR '16. Santa Clara, CA, USA: ACM, 2016, 17:1–17:12. ISBN: 978-1-4503-4211-7. DOI: 10.1145/2890955.2890960.

[Kot+16b] V. Kotronis, R. Klöti, M. Rost, P. Georgopoulos, B. Ager, S. Schmid, and X. A. Dimitropoulos. "Investigating the Potential of the Inter-IXP Multigraph for the Provisioning of Guaranteed End-to-End Services". In: *CoRR* abs/1611.03407 (2016). URL: http://arxiv.org/abs/1611.03407 (visited on Sept. 19, 2019).

[Kot+16c] V. Kotronis, R. Klöti, M. Rost, P. Georgopoulos, B. Ager, S. Schmid, and X. A. Dimitropoulos. "Stitching Inter-Domain Paths over IXPs". In: *CoRR* abs/1611.02642 (2016). URL: http://arxiv.org/abs/1611.02642 (visited on Sept. 19, 2019).

[KS97] S. G. Kolliopoulos and C. Stein. "Improved approximation algorithms for unsplittable flow problems". In: *Proceedings 38th Annual Symposium on Foundations of Computer Science*. Oct. 1997, pp. 426–436. DOI: 10.1109/SFCS.1997.646131.

[Kum+02] A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener. "Algorithms for Provisioning Virtual Private Networks in the Hose Model". In: *IEEE/ACM Trans. Netw.* 10.4 (Aug. 2002), pp. 565–578. ISSN: 1063-6692. DOI: 10.1109/TNET.2002.802141.

[KV18] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer-Verlag Berlin Heidelberg, 2018. ISBN: 978-3-662-56038-9. DOI: 10.1007/978-3-662-56039-6.

[Lao+09] N. Laoutaris, G. Smaragdakis, P. Rodriguez, and R. Sundaram. "Delay Tolerant Bulk Data Transfers on the Internet". In: *SIGMETRICS Perform. Eval. Rev.* 37.1 (June 2009), pp. 229–238. ISSN: 0163-5999. DOI: 10.1145/2492101.1555376.

[LF01] X. Lin and C. Floudas. "Design, synthesis and scheduling of multipurpose batch plants via an effective continuous-time formulation". In: *Computers & Chemical Engineering* 25.4 (2001), pp. 665–674. ISSN: 0098-1354. DOI: https://doi.org/10.1016/S0098-1354(01)00663-9.

[Liu+19] M. Liu, A. W. Richa, M. Rost, and S. Schmid. "A Constant Approximation for Maximum Throughput Multicommodity Routing And Its Application to Delay-Tolerant Network Scheduling". In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. Apr. 2019, pp. 46–54. DOI: 10.1109/INFOCOM.2019.8737402.

[LK09] J. Lischka and H. Karl. "A Virtual Network Mapping Algorithm Based on Subgraph Isomorphism Detection". In: *Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures*. VISA '09. Barcelona, Spain: ACM, 2009, pp. 81–88. ISBN: 978-1-60558-595-6. DOI: 10.1145/1592648.1592662.

[LR01] D. H. Lorenz and D. Raz. "A simple efficient approximation scheme for the restricted shortest path problem". In: *Operations Research Letters* 28.5 (2001). ISSN: 0167-6377. DOI: https://doi.org/10.1016/S0167-6377(01)00069-4.

[LRS16] T. Lukovszki, M. Rost, and S. Schmid. "It's a Match!: Near-Optimal and Incremental Middlebox Deployment". In: *SIGCOMM Computer Communication Review (CCR)* 46.1 (Jan. 2016), pp. 30–36. ISSN: 0146-4833. DOI: 10.1145/2875951.2875956.

[LRS17] T. Lukovszki, M. Rost, and S. Schmid. "Approximate and Incremental Network Function Placement". In: *CoRR* abs/1706.06496 (2017). URL: http://arxiv.org/abs/1706.06496 (visited on Sept. 19, 2019).

[LRS18] T. Lukovszki, M. Rost, and S. Schmid. "Approximate and incremental network function placement". In: *Journal of Parallel and Distributed Computing* 120 (2018), pp. 159–169. ISSN: 0743-7315. DOI: https://doi.org/10.1016/j.jpdc.2018.06.006.

[Lud+14] A. Ludwig, M. Rost, D. Foucard, and S. Schmid. "Good Network Updates for Bad Packets: Waypoint Enforcement Beyond Destination-Based Routing Policies". In: *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*. HotNets-XIII. Los Angeles, CA, USA: ACM, 2014, 15:1–15:7. ISBN: 978-1-4503-3256-9. DOI: 10.1145/2670518.2673873.

[Lud+16] A. Ludwig, S. Dudycz, M. Rost, and S. Schmid. "Transiently Secure Network Updates". In: *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*. SIGMETRICS '16. Antibes Juan-les-Pins, France: ACM, 2016, pp. 273–284. ISBN: 978-1-4503-4266-7. DOI: 10.1145/2896377.2901476.

[Lud+18] A. Ludwig, S. Dudycz, M. Rost, and S. Schmid. "Transiently Policy-Compliant Network Updates". In: *IEEE/ACM Transactions on Networking* 26.6 (Dec. 2018), pp. 2569–2582. ISSN: 1063-6692. DOI: 10.1109/TNET.2018.2871023.

[Men27] K. Menger. "Zur allgemeinen Kurventheorie". ger. In: *Fundamenta Mathematicae* 10.1 (1927), pp. 96–115. URL: http://eudml.org/doc/211191 (visited on Sept. 19, 2019).

[MG07] J. Matousek and B. Gärtner. *Understanding and Using Linear Programming*. Springer-Verlag Berlin Heidelberg, 2007. ISBN: 978-3-540-30697-9. DOI: 10.1007/978-3-540-30717-4.

[Mij+16] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba. "Network Function Virtualization: State-of-the-Art and Research Challenges". In: *IEEE Communications Surveys Tutorials* 18.1 (2016), pp. 236–262. ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2477041.

[MKC13] L. Mai, E. Kalyvianaki, and P. Costa. "Exploiting Time-Malleability in Cloud-based Batch Processing Systems". In: *Workshop on Large-Scale Distributed Systems and Middleware (LADIS'13)*. ACM, Nov. 2013. URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/mai13exploiting.pdf (visited on Sept. 19, 2019).

[MKK14] S. Mehraghdam, M. Keller, and H. Karl. "Specifying and placing chains of virtual network functions". In: *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*. Oct. 2014, pp. 7–13. DOI: 10.1109/CloudNet.2014.6968961.

[MP12] J. C. Mogul and L. Popa. "What We Talk About when We Talk About Cloud Network Performance". In: *SIGCOMM Comput. Commun. Rev.* 42.5 (Sept. 2012), pp. 44–48. ISSN: 0146-4833. DOI: 10.1145/2378956.2378964.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. New York, NY, USA: Cambridge University Press, 1995. ISBN: 9780521474658.

[MT14] H. Moens and F. D. Turck. "VNF-P: A model for efficient placement of virtualized network functions". In: *10th International Conference on Network and Service Management (CNSM) and Workshop*. Nov. 2014, pp. 418–423. DOI: 10.1109/CNSM.2014.7014205.

[MU17] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. 2nd. New York, NY, USA: Cambridge University Press, 2017. ISBN: 9781107154889. DOI: 10.1017/cbo9780511813603.

[Nap+16] J. Napper, W. Haeffner, M. Stiemerling, D. R. Lopez, and J. Uttaro. *Service Function Chaining Use Cases in Mobile Networks*. Internet-Draft. Apr. 2016. 26 pp. URL: https://tools.ietf.org/html/draft-ietf-sfc-use-case-mobility-06 (visited on Sept. 19, 2019).

[Ném+16] B. Németh, B. Sonkoly, M. Rost, and S. Schmid. "Efficient service graph embedding: A practical approach". In: *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. Nov. 2016, pp. 19–25. DOI: 10.1109/NFV-SDN.2016.7919470.

[Ném+18] B. Németh, M. Szalay, J. Dóka, M. Rost, S. Schmid, L. Toka, and B. Sonkoly. "Fast and efficient network service embedding method with adaptive offloading to the edge". In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. Apr. 2018, pp. 178–183. DOI: 10.1109/INFCOMW.2018.8406882.

[Pap81] C. H. Papadimitriou. "On the Complexity of Integer Programming". In: *J. ACM* 28.4 (Oct. 1981), pp. 765–768. ISSN: 0004-5411. DOI: 10.1145/322276.322287.

[PH13] R. Pal and P. Hui. "Economic models for cloud service markets: Pricing and Capacity planning". In: *Theoretical Computer Science* 496 (2013). Distributed Computing and Networking (ICDCN 2012), pp. 113–124. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2012.11.001.

[Pop+13] L. Popa, P. Yalagandula, S. Banerjee, J. C. Mogul, Y. Turner, and J. R. Santos. "ElasticSwitch: Practical Work-conserving Bandwidth Guarantees for Cloud Computing". In: *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*. SIGCOMM '13. Hong Kong, China: ACM, 2013, pp. 351–362. ISBN: 978-1-4503-2056-6. DOI: 10.1145/2486001.2486027.

[QN15] P. Quinn and T. Nadeau. *Problem Statement for Service Function Chaining*. RFC 7498. Apr. 2015. DOI: 10.17487/RFC7498.

[Rab+13] M. G. Rabbani, R. P. Esteves, M. Podlesny, G. Simon, L. Z. Granville, and R. Boutaba. "On tackling virtual data center embedding problem". In: *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. May 2013, pp. 177–184.

[Räc08] H. Räcke. "Optimal Hierarchical Decompositions for Congestion Minimization in Networks". In: *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*. STOC '08. Victoria, British Columbia, Canada: ACM, 2008, pp. 255–264. ISBN: 978-1-60558-047-0. DOI: 10.1145/1374376.1374415.

[Rag86] P. Raghavan. "Randomized Rounding And Discrete Ham-Sandwich Theorems". dissertation. Berkeley, CA, USA: University of California at Berkeley, 1986. URL: http://digitalassets.lib.berkeley.edu/techreports/ucb/text/CSD-87-312.pdf (visited on Sept. 19, 2019).

[Rag88] P. Raghavan. "Probabilistic construction of deterministic algorithms: Approximating packing integer programs". In: *Journal of Computer and System Sciences* 37.2 (1988), pp. 130–143. ISSN: 0022-0000. DOI: https://doi.org/10.1016/0022-0000(88)90003-7.

[RAL03] R. Ricci, C. Alfeld, and J. Lepreau. "A Solver for the Network Testbed Mapping Problem". In: *SIGCOMM Comput. Commun. Rev.* 33.2 (Apr. 2003), pp. 65–81. ISSN: 0146-4833. DOI: 10.1145/956981.956988.

[RDS19] M. Rost, E. Döhne, and S. Schmid. "Parametrized Complexity of Virtual Network Embeddings: Dynamic & Linear Programming Approximations". In: *SIGCOMM Comput. Commun. Rev.* 49.1 (Feb. 2019), pp. 3–10. ISSN: 0146-4833. DOI: 10.1145/3314212.3314214.

[RFS15] M. Rost, C. Fuerst, and S. Schmid. "Beyond the Stars: Revisiting Virtual Cluster Embeddings". In: *SIGCOMM Computer Communication Review (CCR)* 45.3 (July 2015), pp. 12–18. ISSN: 0146-4833. DOI: 10.1145/2805789.2805792.

[RS13] M. Rost and S. Schmid. *Technical Report and Simulation Results for the Paper "It's About Time: On Optimal Virtual Network Embeddings under Temporal Flexibilities"*. Oct. 2013. DOI: 10.5281/zenodo.1490172.

[RS16] M. Rost and S. Schmid. "Service Chain and Virtual Network Embeddings: Approximations using Randomized Rounding". In: *CoRR* abs/1604.02180 (2016). URL: http://arxiv.org/abs/1604.02180 (visited on Sept. 19, 2019).

[RS18a] M. Rost and S. Schmid. "Charting the Complexity Landscape of Virtual Network Embeddings". In: *2018 IFIP Networking Conference (IFIP Networking)*. May 2018, pp. 1–9. DOI: 10.23919/IFIPNetworking.2018.8696604.

[RS18b] M. Rost and S. Schmid. "NP-Completeness and Inapproximability of the Virtual Network Embedding Problem and Its Variants". In: *CoRR* abs/1801.03162 (2018). URL: http://arxiv.org/abs/1801.03162 (visited on Sept. 19, 2019).

[RS18c] M. Rost and S. Schmid. "Virtual Network Embedding Approximations: Leveraging Randomized Rounding". In: *2018 IFIP Networking Conference (IFIP Networking)*. May 2018, pp. 1–9. DOI: 10.23919/IFIPNetworking.2018.8696623.

[RS18d] M. Rost and S. Schmid. "Virtual Network Embedding Approximations: Leveraging Randomized Rounding". In: *CoRR* abs/1803.03622 (2018). URL: http://arxiv.org/abs/1803.03622 (visited on Sept. 19, 2019).

[RS19a] M. Rost and S. Schmid. "On the Hardness and Inapproximability of Virtual Network Embeddings". In: *IEEE/ACM Transactions on Networking (under submission)* (2019).

[RS19b] M. Rost and S. Schmid. "Virtual Network Embedding Approximations: Leveraging Randomized Rounding". In: *IEEE/ACM Transactions on Networking (to appear)* (2019).

[RSF14] M. Rost, S. Schmid, and A. Feldmann. "It's About Time: On Optimal Virtual Network Embeddings under Temporal Flexibilities". In: *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. May 2014, pp. 17–26. DOI: 10.1109/IPDPS.2014.14.

[RT85] P. Raghavan and C. D. Thompson. "Provably Good Routing in Graphs: Regular Arrays". In: *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*. STOC '85. Providence, Rhode Island, USA: ACM, 1985, pp. 79–87. ISBN: 0-89791-151-2. DOI: 10.1145/22145.22154. (Visited on Sept. 19, 2019).

[RT87] P. Raghavan and C. D. Tompson. "Randomized rounding: A technique for provably good algorithms and algorithmic proofs". In: *Combinatorica* 7.4 (Dec. 1987), pp. 365–374. ISSN: 1439-6912. DOI: 10.1007/BF02579324.

[SA90] H. Sherali and W. Adams. "A Hierarchy of Relaxations between the Continuous and Convex Hull Representations for Zero-One Programming Problems". In: *SIAM Journal on Discrete Mathematics* 3.3 (1990), pp. 411–430. DOI: 10.1137/0403036.

[Sah+15] S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet, and P. Demeester. "Network service chaining with optimized network function embedding supporting service decompositions". In: *Computer Networks* 93 (2015). Cloud Networking and Communications II, pp. 492–505. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2015.09.035.

[See10] S. Seetharaman. "Energy Conservation in Multi-tenant Networks Through Power Virtualization". In: *Proceedings of the 2010 International Conference on Power Aware Computing and Systems*. HotPower'10. Vancouver, BC, Canada: USENIX Association, 2010, pp. 1–8.

[She+12] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar. "Making Middleboxes Someone else's Problem: Network Processing As a Cloud Service". In: *SIGCOMM Comput. Commun. Rev.* 42.4 (Aug. 2012), pp. 13–24. ISSN: 0146-4833. DOI: 10.1145/2377677.2377680.

[Shi+98] T. K. Shih, A. Y. Chang, H.-J. Lin, S.-H. Yen, and C.-F. Chiu. "Interval algebra for spatio-temporal composition of distributed multimedia objects". In: *Proceedings 1998 International Conference on Parallel and Distributed Systems (Cat. No.98TB100250)*. Dec. 1998, pp. 308–315. DOI: 10.1109/ICPADS.1998.741088.

[SIB03] W. Szeto, Y. Iraqi, and R. Boutaba. "A multi-commodity flow based approach to virtual network resource allocation". In: *GLOBECOM '03. IEEE Global Telecommunications Conference (IEEE Cat. No.03CH37489)*. Vol. 6. Dec. 2003, 3004–3008 vol.6. DOI: 10.1109/GLOCOM.2003.1258787.

[Sin+12] A. Singla, C. Hong, L. Popa, and P. B. Godfrey. "Jellyfish: Networking Data Centers Randomly". In: *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2012, San Jose, CA, USA, April 25-27, 2012*. 2012, pp. 225–238. URL: https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final82.pdf (visited on Sept. 19, 2019).

[Skö+14] P. Sköldström, B. Sonkoly, A. Gulyás, F. Németh, M. Kind, F.-J. Westphal, W. John, J. Garay, E. Jacob, D. Jocha, J. Elek, R. Szabó, W. Tavernier, G. Agapiou, A. Manzalini, M. Rost, N. Sarrar, and S. Schmid. "Towards Unified Programmability of Cloud and Carrier Infrastructure". In: *2014 Third European Workshop on Software Defined Networks*. Sept. 2014, pp. 55–60. DOI: 10.1109/EWSDN.2014.18.

[Soa+15] J. Soares, C. Gonçalves, B. Parreira, P. Tavares, J. Carapinha, J. P. Barraca, R. L. Aguiar, and S. Sargento. "Toward a telco cloud environment for service functions". In: *IEEE Communications Magazine* 53.2 (Feb. 2015), pp. 98–106. ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7045397.

[Sou+14] R. Soulé, S. Basu, P. J. Marandi, F. Pedone, R. Kleinberg, E. G. Sirer, and N. Foster. "Merlin: A Language for Provisioning Network Resources". In: *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*. CoNEXT '14. Sydney, Australia: ACM, 2014, pp. 213–226. ISBN: 978-1-4503-3279-8. DOI: 10.1145/2674005.2674989.

[SSF12] G. Schaffrath, S. Schmid, and A. Feldmann. "Optimizing Long-Lived CloudNets with Migrations". In: *2012 IEEE Fifth International Conference on Utility and Cloud Computing*. Nov. 2012, pp. 99–106. DOI: 10.1109/UCC.2012.7.

[Tam17] H. Tamaki. "Positive-Instance Driven Dynamic Programming for Treewidth". In: *25th Annual European Symposium on Algorithms (ESA 2017)*. Ed. by K. Pruhs and C. Sohler. Vol. 87. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 68:1–68:13. ISBN: 978-3-95977-049-1. DOI: 10.4230/LIPIcs.ESA.2017.68.

[WS11] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. 1st. New York, NY, USA: Cambridge University Press, 2011. ISBN: 9780521195270.

[Wu+09] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang. "MDCube: A High Performance Network Structure for Modular Data Center Interconnection". In: *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*. CoNEXT '09. Rome, Italy: ACM, 2009, pp. 25–36. ISBN: 978-1-60558-636-6. DOI: 10.1145/1658939.1658943.

[Xie+12] D. Xie, N. Ding, Y. C. Hu, and R. Kompella. "The Only Constant is Change: Incorporating Time-varying Network Reservations in Data Centers". In: *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. SIGCOMM '12. Helsinki, Finland: ACM, 2012, pp. 199–210. ISBN: 978-1-4503-1419-0. DOI: 10.1145/2342356.2342397.

[YLH17] F. Yan, T. T. Lee, and W. Hu. "Congestion-Aware Embedding of Heterogeneous Bandwidth Virtual Data Centers With Hose Model Abstraction". In: *IEEE/ACM Trans. Netw.* 25.2 (Apr. 2017), pp. 806–819. ISSN: 1063-6692. DOI: 10.1109/TNET.2016.2606480.

[Yu+08] M. Yu, Y. Yi, J. Rexford, and M. Chiang. "Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration". In: *SIGCOMM Comput. Commun. Rev.* 38.2 (Mar. 2008), pp. 17–29. ISSN: 0146-4833. DOI: 10.1145/1355734.1355737.

[Zha+14] Q. Zhang, M. F. Zhani, M. Jabri, and R. Boutaba. "Venice: Reliable virtual data center embedding in clouds". In: *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. Apr. 2014, pp. 289–297. DOI: 10.1109/INFOCOM.2014.6847950.